MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Sistem de afișare al datelor meteorologice pe hartă digitală

LUCRARE DE LICENŢĂ

Absolvent: **Miklós Erik**

Coordonator ştiinţific: **Asist.Ing. Cosmina IVAN**

2017

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Miklós ERIK**

TITLUL LUCRĂRII DE LICENŢĂ

1. **Enunţul temei:** *Scurtă descriere a temei lucrării de licenţă şi datele iniţiale*

2. **Conţinutul lucrării:** Cuprins, Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză şi fundamentare teoretică, Proiectare de detaliu şi implementare, Testare şi validare, Manual de instalare şi utilizare, Concluzii, Bibliografie, Anexe.

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**: Asist.Ing. Cosmina Ivan

5. **Data emiterii temei:** 1 martie 2016

6. **Data predării:** 17 Februarie 2017

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a)_____
_____,
legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării
_____
_____
_____elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

_____

Nume, Prenume

_____

Semnătura

Cuprins

Capitolul 1. Introducere

În zilele noastre toate instrumentele care ne folosim au început să se digitalizeze pe rând. Procesul asta de digitalizare acopera atât și viața personală, dar și uneltele folosite la lucru. Uneltele digitale sunt mai simple, mai simplu de folosit și mult mai flexibile, astfel simplificând viața omului modern. În ultimii douăzeci de ani și hărțile tipărite au fost digitalizate și prin acest proces s-au extins posibilitățile de utilizare al acestor instrumente software în diverse sisteme și aplicații.

Digitalizarea globală s-a început cu digitalizarea datelor. Datele digitalizate au înlocuit datele tipărite pe hârtie, și în zilele noastre sunt mai stabile și sigure decât hârtia. Datele digitale permit stocarea și gestionarea mult mai multe date și cu o eficiență mult mai bună. Cu timpul au apărut seturi de date imense care pentru exemplu pot stoca și gestiona datele meteorologice începînd din anii 80 până azi.

1.1. Contextul proiectului

Vremea este o informație vitală care definește o mare parte al vieții noastre și această proprietate e în schimbare constantă. Astfel este foarte important să conoaștem starea și viitorul acestor informații.

Aplicația creată poate să ne informă despre mai multe informații meteorologice și ne ajută la evidențierea acestora oriunde pe pământ. Aplicația oferă un set de funcționalității care simplifică găsirea datelor meteorologice interesate. Are o interfață familiară asemănătoare diverselor aplicații care folosesc hărții, astfel navigarea și afișarea informațiilor este deja cunoscută pentru majoritatea utilizatorilor. În afară de utilizări clasice, aplicația salvează locațiile căutate de către utilizatori, astfel utilizatorii pot primi datele de interes, în timp relativ scurt după accesarea aplicației.

Aplicația oferă tot timpul informații proaspete, prin actualizarea automată al datelor în baza de date. Aplicația descarcă datele meteorologice de pe un set de date foarte cunoscută și recunoscută.

Gruparea punctelor pe hartă ajută la vizibilitatea datelor pe hartă, este o problemă comună, la hărți că la o scară specifică este limitat volumul datelor și detaliilor afisate pe hartă. Din această cauza s-a folosit un algoritm de clusterizare, care grupează datele pe hartă, prin schimbarea scării putem ascunde sau afișa diverse niveluri de detaliere a datelor.

1.2. Motivația

Proiectul curent realizează implementarea unui sistem de afișare al dateleor metorologice pe hărți digitale pentru a crea o aplicație utilă, cu scopul că un utilizator să poată găsi informațiile meteorologice rapid și ușor.

Aplicația permite afișarea datelor meteorologice pentru întregul glob pământesc, putând constitui o sursă de informare utilă și cu bună aplicabilitate.

1.3. Conținutul lucrării

Acest document conține următoarele opt capitole:

Capitolul 1 (Introducere)

În acest capitol este prezentat contextul general al proiectului și tematica propusă, împreună cu motivarea alegerii și structura acestui document.

Capitolul 2 (Obiectivele Proiectului)

În acest capitol sunt prezentate obiectivele propriu-zise ale proiectului. Capitolul prezintă și obiectivele specifice ale sistemului.

Capitolul 3 (Studiu Bibliografic)

În acest capitol sunt prezentate technologiile pe care se bazează proiectul, sunt prezentate hărțile digitale, evoluția lor și o scurtă comparație între technologii existente. Sunt prezentate seturile de date meteorologice existente, evoluția lor și structura acestora. Capitolul conține descrierea sistemelor similare.

Capitolul 4 (Analiză și Fundamentare Teoretică)

În acest capitol este prezentată arhitectura conceptuală a sistemului, descrierea detaliată al funcționalităților aplicației, technologiile folosite la dezvoltarea sistemului și protocoalele pe care folosește aplicația.

Capitolul 5 (Proiectare de Detaliu și Implementare)

În acest capitol este prezentată structura și arhitectura sistemului și descrierea de implementare a componentelor. Pe lângă acestea, sunt prezentate structura bazei de date, diagrama de deployment și fluxul de execuție al sistemului.

Capitolul 6 (Testare și Validare)

În acest capitol sunt prezentate metodele de testare folosite la testarea sistemului. Sunt prezentate metode manuale și automate de testare și validare a splicației.

Capitolul 7 (Manual de Instalare și Utilizare)

În acest capitol sunt prezentați pașii de instalare al sistemului și manualul de utilizarea a aplicației.

Capitolul 8 (Concluzii)

Acest capitol conține un set de concluzii legate la sistem și recomandări pentru înbunătățirea și extinderea sistemului.

Capitolul 2. Obiectivele Proiectului

În acest capitol este prezentat obiectivul general al proiectului, precum și obiectivele specifice, definite de cerințele funcționale și non-funcționale. Obiectivul general în constituie afișarea datelor meteorologice pe hărți digitale.

2.1. Obiective generale

În principiu proiectul este alcătuit din două module, sistemul trebuie să conțină o aplicație care poate descărca datele meteorologice dintr-un set de date meteorologice, această aplicație va descărca mai multe tipuri de date meteorologice ca și temperatura, presiunea atmosferică, radiația UV și precipitația. După descărcarea acestor date aplicația trebuie să salvează datele într-o bază de date care este rapidă și usor accesibilă de către cealaltă componentă a sistemului. Cea de a două componentă a sistemului este un sistem de afișare al datelor meteorologice. Această aplicație trebuie să folosească un modul cu o hartă digitală, pe care sunt afișate datele meteorologice.

Aplicația oferă informații meteorologice generalej, ca și temperatura, pentru toți vizitatorii aplicației, temperatura este cea mai importantă informație generală.

Aplicația oferă un set de informații meteorologice specifice pentru utilizatorii înregistrați în sistem, astfel sistemul trebuie să gestioneze și datele utilizatorilor.

De asemenea aplicația oferă utilizatorilor posibilitatea de a se înregistra în sistem, utilizatorul poate accesa mai multe informații meteorologice prin autentificarea în sistem.

Pentru utilizatorii autentificații sistemul oferă opțiunea de a alege informația meteorologică afișată pe harta, dar și extinderea punctelor de pe hartă pentru aflarea informațiilor detaliate despre punctul respectiv.

Aplicația oferă utilizatorilor autentificate posibilitatea să caută locații pe hartă, și aflarea detaliilor meteorologice apropiate la locația căutată de către utilizator.

Harta digitală trebuie să ofere posibilitatea de navigare pa hartă, navigarea trebuie să fie ușor de înțeles și rapidă.

2.2. Obiective specifice

Obiectivele specifice se referă atât la funcționalitățiile necesare sistemului prezentate mai sus, cât și funcționalitățile care sunt necesare pentru simplificarea utilizării sistemului.

Obiectivul specific cel mai important este folosirea unui algoritm de clusterizare al punctelor apărute pa hartă. Sistemul operează cu foarte multe date, sistemul acoperă întregul pământ oferind informațiile meteorologice aproape în fiecare colț al hărții. Asta înseamnă o cantitate enormă de date și puncte afișate pe hartă, astfel ca rezultat datele sunt greu de citit, markerele acoperă toată harta. Din acest motiv sistemul folosește un algoritm de clusterizare, care grupează punctele afișate pe hartă, și afișează o valoare medie a datelor meteorologice, astfel un punct indică informațiile meteorologice pentru o arie mai largă. Metoda de clusterizare asigură ca la orice stare a hărții densitatea punctelor afișate să fie optimă și utilizatorii să poată citi informațiile ușor.

Un alt obiectiv specific al aplicației este salvarea locațiilor căutate da către utilizatori, astfel utilizatorul poate regăsi locațiile interesate instant după fiecare accesare al aplicației.

Capitolul 3. Studiu Bibliografic

În acest capitol sunt prezentate pe scurt rezultatele studiului bibliografic şi analiza sistemelor similare pe care este bazată ideea din spatele proiectului. În acest capitol sunt prezentate informaţii generale despre hărţi digitale, seturi de date meteorologice şi alte aplicaţii de monitorizare meteorologică pe hărţi online.

3.1. Hărţi digitale

O hartă digitală pe internet este şi consumată şi furnizată, distribuitorii de hărţi digitale oferă servicii prin care utilizatorii aleg informaţiile care doresc să apară pe hartă. Funcţionalitatea cea mai importantă al acestor distribuitori este producerea hărţţilor care oferă o reprezentare precisă a ariei respective [4].

3.1.1. Tipuri de hărţi digitale

Primele clasificări ale hărţilor digitale au fost specificate de Kraak Brown în anul 2001, el a distins hărţii statice şi dinamice şi mai de vreme hărţii interactive şi numai afişabile. În ziua de azi există următoarele tipuri de hărţi digitale. [5]

- Hărţi digitale analitice
- Hărţi digitale animate şi de timp real
- Hărţi digitale colaborative
- Atlase online
- Hărţii digitale statice

OpenStreetMap şi Google Maps sunt hărţii colaborative, utilizatorii sunt colaboratori în fluxul de creare şi înbunătăţire al acestor hărţi [8].

3.1.2. Istoria hărţilor digitale

Până acum câţiva ani, sintetizarea, manipularea şi reprezentarea geografică a informaţiei a fost restricţionată la hărţi pe hârtie şi aceste cerinţe au fost limitate la procese manuale, neinteractive. Îmbunătăţirea exponenţială a performanţei bazată pe tehnologii diverse de procesare computerizată şi cercetarea, încă în creştere, de manipulare interactivă şi analiză a informaţiei geografice a creat nevoia de sisteme geografice de informare GIS.

Cu apariţia sistemelor geografice au apărut mai multe posibilităţi de utilizare al acestor sisteme, iar mulţimea datelor afişate pe aceste sisteme a crescut direct proporţional.

Un avantaj al harţii digitale este că putem schimba scara geografică, iar acest avantaj atrage o mare problema: datele care arată bine şi sunt vizibile la o scară mica, sunt aglomerate la o scară mai mare. Pentru rezolvarea acestei probleme există două abordări, una este ascunderea parţială sau totală a datelor, a doua este folosirea un algoritm de clustering. Pentru marii furnizorii de date geografice deja sunt dezvoltate funcţionalităţi de grupare.

Astfel, Static Maps API are funcţionalitatea de a grupa informatiile pe GoogleMaps, iar de la altă parte plugin-ul Mapbox şi OpenLayers se grupeaza datele pe setul OpenStreetMaps.

Hărți tradiționale tipărite oferă o privire similară ca și cele digitale, dar de multe ori sunt greu utilizabile, acoperă numai o arie specifică și nu poate conține informații des actualizate. Însumând nu există nici o soluție pentru actualizarea hărților tipărite. În schimb hărți digitale pot fi actualizate. Evoluția hărților digitale este prezentată cronologic [5]:

- 1993: Xerox PARC Map Viewer, primul server de hărți bazat pe CGI/Perl, permite proiecția, stilizarea și definirea hărții.
- 1994: The Word Wide Earthquake Locator, primul hartă digitală interactivă, bazat pe Xerox PARC Map Viewer.
- 1994: The National Atlas of Canada, prima versiune de National Atlas of Canada a fost lansat. Poate fi considerat ca și primul atlas online.
- 1995: The Gazetteer for Scotland, a fost lansat prototipul al Gazetteer pentru Scoția. Prima bază de date geografică cu mapare interactivă.
- 1995: MapGuide, inițial introdus ca Argus MapGuide
- 1996: Mapquest, primul hartă digitală populare care are funcționalitatea de Address Matching și Pouting Service.
- 1996: MultiMap, unul dintre cele mai populare hărti online în Marea Britanie.
- 1996: Geomedia WebMap 1.0, prima versiune de Geomedia WebMap, deja suportă grafica vectorială prin folosirea ActiveCGM.
- 1997: US Online National Atlas Initiative, USGS coordinează și crează atlasul național și online al Statelor Unite.
- 1997: UMN MapServer 1.0, dezvoltat la University of Minnesota o parte al proiectului NASA ForNet Project.
- 1997: GeoInfoMapper, a fost dezvoltat primul aplet bazat pe Java și GIS numit JavaMap. Apicația suportă exportarea și conversia datelor geografice.
- 1998: Terraserver USA,un servicui care furnizează imagini aeriale de la Microsoft și HP.
- 1998: MapObjects Inernet Map Server, organizația ESRI a intrat în business-ul hărți online.
- 2000: ESRI Geography Network, ESRI a fundat Geography Network pentru distribuirea servicilor de date și hărți online.
- 2000: UMN MapServer 3.0, suportul de raster adaugat, primul UMN MapServer cu sursă deschisă.
- 2001: GeoServer, a început projectul GeoServer.
- 2002: UMN MapServer 3.5, suport pentru PostGIS și ARCSDE adaugat.
- 2003: Nasa World Wind, un glob online virtual care încarcă datele din resurse distribuite. Utilizatorii pot adauga conținuturi personale.
- 2004: OpenStreetMap, o hartă digitală cu sursă și conținut deschis fundat de Steve Coast.
- 2005: Google Maps, prima versiune de Google Maps, bazat pe rastere organizate în arbori, datele încărcate asincron. Permite încorporarea hărți digitale în orice altă pagină web.
- 2005: Google Earth, un glob virtual cu obiecte tri-dimensionale, permite utilizatorilor sa integreze conținuturile personale.
- 2005: OpenLayers, prima versiune al librariei cu sursă deschisă JavaScript.
- 2006: WikiMapia

- 2009: Compania Nokia a lansat hărții Ovi Maps care a fost accesibil pe dispozitivele Nokia
- 2010: A fost fundat MapBox-ul
- 2012: Compania Apple elimină Google Maps ca și aplicație de mapare implicită, și crează propriul aplicație de mapare.

3.1.3. OpenStreetMap și GoogleMaps

OpenStreetMap născut in 2004, în acest timp hărțile digitale au fost controlate de către organizații guvernamentale și private, au fost costisitoare și foarte restricționate astfel au fost accesibile numai de către companii mari. Ideea din spatele OSM era rezolvarea acestei probleme, prin crearea unor hărții digital editabile și extensibile.[23]

Această abordare a fost recunoscută și de către Google, și a deschis în direcția utilizatorilor. În anul 2008 compania a întrodus aplicația Google Map Maker care a avut abordări și interfață similară ca și OSM.

Diferența majoră dintre aceste două servicii este că dacă un utilizator editează harta OSM, proprietarul modificărilor va fi utilizatorul și comunitatea, în schimb la Google Maps proprietarul modificărilor va fi compania Google.[8]

Proiectul OpenStreetMap este deschisă, cu o licență deschisă. Datele care construiesc harta sunt date de tip vector. Punctele de coordonate ale străzii sunt clasificate linear alcătuit din puncte cu legături între ele. Google Maps nu e deschis în aceași nivel de date de construire al hărților. Aceste date nu sunt accesibile pentru dezvoltatori. Astfel Google Maps poate oferi produse și servicii care folosesc date geografice deja randate. Google de asemenea protejează drepturile cu mai multe variații de copyright și restricții. Google maps este o platformă foarte populară și recunoscută, dar există realizarea puterii datelor georgafice accesibile și posibilităților al Planet.osm de unde se pot descărca datele pentru întregul pământ. Serviciile oferite de Google Maps sunt foarte importante, există servicii similare și pentru OpenStreetMap.[4]

3.2. Seturi de date meteorologice

Seturi de date meteorologice de obicei sunt arhive al organizațiilor mari care se ocupă cu meteorologie. Aceste arhive sunt construite din colectarea datelor continuă pe întregul glob. Cu timpul aceste seturi de date au crescut foarte mari, unele dintre marile organizații au creat propria arhitectură de stocare și limbaj pentru interogarea și gestionarea datelor și arhivelor meteorologice. Datele arhivate sunt des analizate și reanalizate pentru înțelegerea evenimentelor climatice. Analizele datelor meteorologice culese crează baza prognozelor meteorologice.

Seturi de date meteorologice globale de temperatură sunt surse esențiale pentru monitorizarea și înțelegerea schimbărilor climatice. Seturi de date cel mai des folosite combină observațiile istorice, temperatura de aer și datele stațiilor pe pământ cu seturi de date globale de temperatură și de suprafață al mării. Conceptul al seturi de date meteorologice este simplu dar obținerea datelor e mai grea. Seturi de date cel mai utilizate: MLOST de la organizația NOAA, setul de date GISTEMP de la NASA și HadCRUT din Marea Britanie. [25]

Setul de date Global Historical Climatology Network (GHCN) este o bază de date al rezumatelor climatice din stații meteorologice pe suprafața pământului. Datele sunt obținute din mai mult de 20 de surse. Niste date sunt mai vechi decăt 175 de ani în

schimb altele sunt obținute de câteva ore. GHCN este setul de date de arhive oficială și înlocuieste seturi de date mai vechi de tip NCEI [26].

Organizația European Centre for Medium-Range Weather Forecast (ECMWF), este o organizație interguvernamentală suportată de cele mai multe țări din Europa și acționează setul de date MARS. Setul de date Meteorological Archival and Retrieval System colectează datele începând din anul 1985. Setul de date la început a crescut cu 70 Mbytes în fiecare zi. În anul 1996 cresterea setului de date a ajuns la 125 Gbytes pentru date operaționale și 7 Gbyte pentru date de cercetare în fiecare zi, arhiva totală a fost 22 Tbytes. Din aceste motive ECMWF a creat o arhitectură și un model de date foarte avansată pentru stocarea acestor date [13].

3.3. Sisteme similare

3.3.1. Netatmo weathermap

Site-ul weathermap [27] este o hartă online care afisează datele colectate din stațiile portabile fabricate de firma Netatmo.

Aplicația are la bază o platformă de GoogleMaps și folosește setul de date GeoBasis-De/BKG, care este setul de bază al hărții de la Google.

Applicația acoperă toată harta, iar acoperirea depinde de locația utilizatorilor. Se integrează funcționalitățile de baza al GooleMaps-ului ca și zoom in, zoom out și funcționalitatea de drag cu ajutorul mouse-ului.

Sistemul ne oferă două tipuri de informații despre vreme: prima este temperatura, temperatura este afișată numeric și marcată cu coduri de culoare, sunt șapte culori diferite pentru reprezentarea temperaturii, se mai poate vizualiza umiditatea, cantitatea precipității este afișată cu ajutorul a șapte tonuri diferite ale culorii albastru.

Umiditatea poate fi vizualizată pe zi, pe oră, sau live, iar temperatura poate fi vizualizată direct sau filtrat.

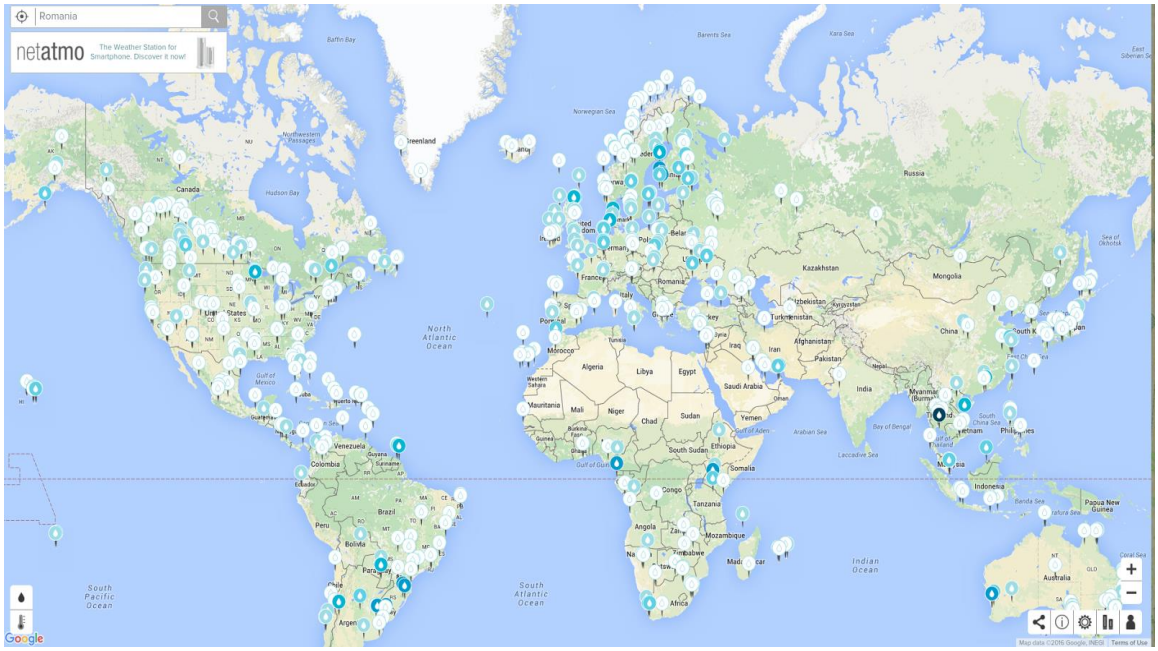

Figura 3.1 Interfața aplicației Weathermap

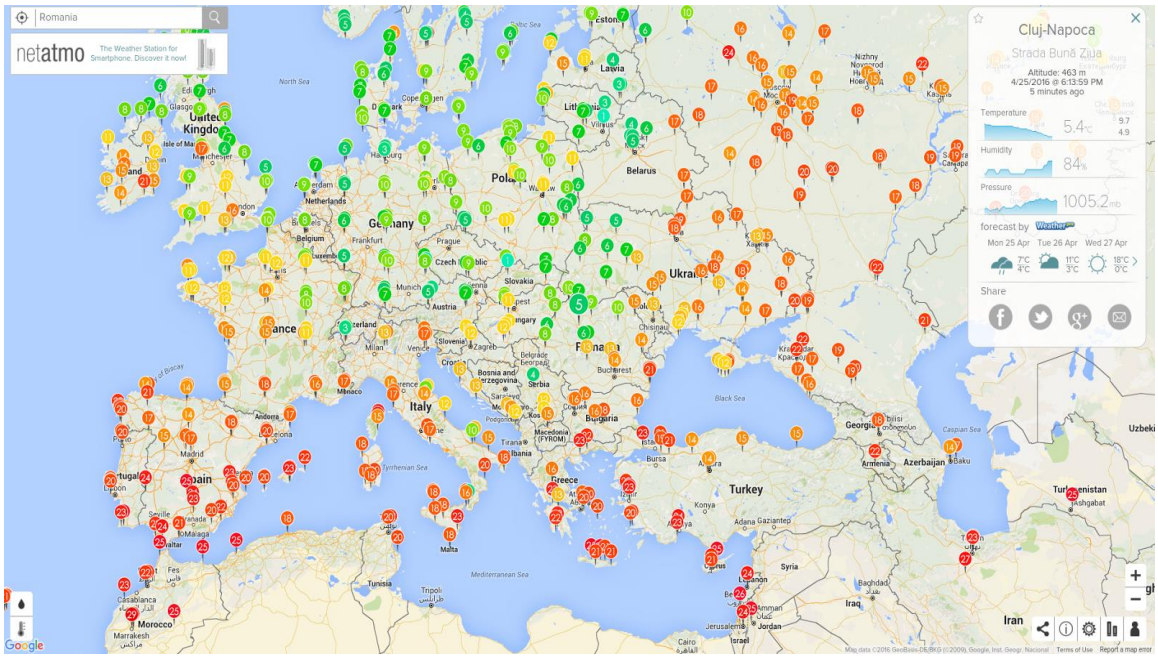

Figura 3.2 Interfața aplicației Weathermap de la firma Netatmo.

Pentru fiecare marker se pot afișa informațiile detaliate într-o fereastră pop-up unde lângă temperatură și umiditate se mai afișează și presiunea și locația senzorului.

Pe acestă pagină se găsește o bară de căutare, unde putem căuta locația dorită.

Dezavantajul acestui sistem este că datele se suprapun de foarte multe ori și datele sunt greu de vizualizate.

3.3.2. Davis WeatherLink

Firma Davis vinde stații meteorologice profesionale pentru folosire publică sau privată. Site-ul WeatherLink [28] afișează datele colectate din aceste stații.

Applicația are la bază o platformă de Google Maps și acopera toată harta, și în acest sistem acoperirea depinde de utilizatori, dar firma fabrica și stații oficiale, deci acoperirea și calitatea datelor este mult mai bună. Se integrează funcționalitățiile de baza al GooleMaps-ului ca și zoom in, zoom out și funcționalitatea de drag cu ajutorul mouse-ului, lânga aceste fucționalității aplicația permite schimbarea între harta regulară și hartă cu imagini de satelit.

Aplicatia folosește un algoritm de grupare bazat pe grid, care ne dă o solutie nu chiar optima. Pe hartă sunt afișate numărul stațiilor în grupuri, iar datele pot fi citite numai dacă afișăm detaliile despre o singură statie, Marker-ele grupurilor sunt colorate in funcție de temperatură, dar date concrete nu sunt afișate

Pentru fiecare marker se pot afișa informațiile detaliate într-o fereastră pop-up unde sunt afișate informațiile: temperatura, umiditatea, presiunea atmosferică și locația senzorului.

Pe această pagină se găsește o bară de căutare, unde putem căuta locația dorită.

Avantajul sistemului față de cel precedent este că reacționeaza mult mai rapid la diverse comenzi, și se incarcă datele foarte repede.

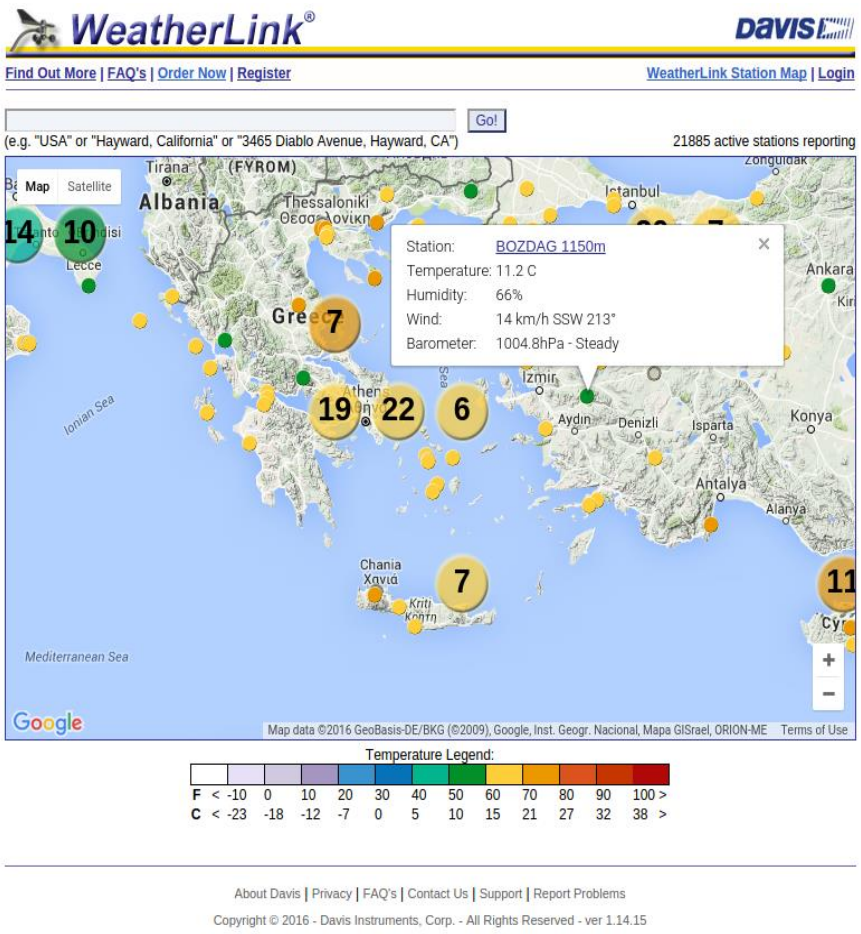

Figura 3.3 Interfața aplicației WeatherLink de la firma Davis

3.3.3. WunderMap

Wundemap [29] este o applicație similară ca și cele detaliate mai sus, iar acest sistem folosește stații meteorologice reale și de nivel mare, care colectează și analizează mult mai multe date, ca: direcția și puterea vântului, puterea soarelui, radiația UV, punctele extreme, focuri de pădure, etc. Au senzori și pe părți acoperite cu apă, afișează datele in timp real, si se reactualizează automat, dar pot fi vizualizate datele in orice moment trecut, sau evoluția datelor intr-un interval de timp. Pe lângă funcționalitățile prezentate la sistemele de mai sus, acest sistem ne permite planificarea rutelor. Sistemul are o pagină unde putem vizualiza diverse statistici ale datelor meteorologice.

Acest sistem folosește algoritmi de grupare bazate pe distanțe între marker-e, și informațiile sunt reprezentate pe mai multe layer-e, care îmbunătățesc performanța aplicației.

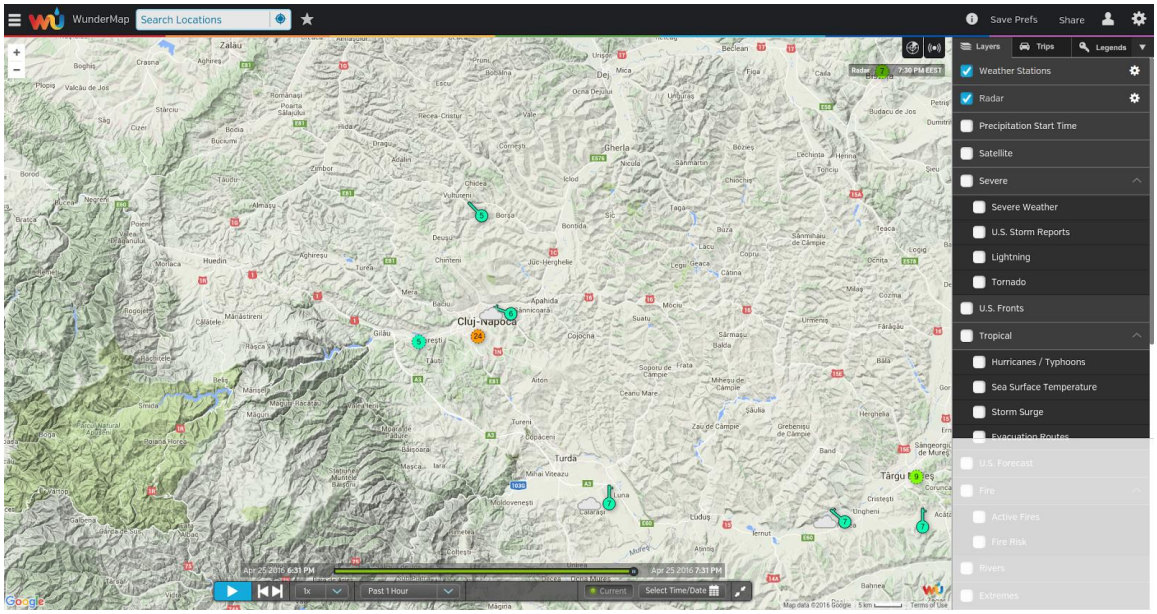

Figura 3.4 Interfața aplicației WunderMap

3.3.4. Concluzii comparative între sisteme similare

În urma studiului realizat au fost analizate sistemele de mai sus, se poate observa că, aplicațiile descrise oferă printre cele mai importante funcționalități, posibilitatea de a grupa punctele de interes și de vizualizare a datelor meteorologice pe hartă. În opinia noastră, cel mai important în cadrul unei astfel de aplicații este citirea usoară a datelor in orice scara geografică și permiterea unei navigări ușoare și intuitive în cadrul sistemului. Putem observa că acest lucru este îndeplinit de toate aplicațiile similare prezentate, în același timp sistemul dezvoltat încercând să se implementează cât mai multe funcționalități pe care utilizatorul le poate accesa.

Am ales să descriu toate exemplele de mai sus, care reprezintă aplicații asemănătoare cu aplicația dezvoltată, deoarece prezintă în mare parte aceleași funcționalități și au fost o sursă de documentare pentru interfața și pentru operațiile care au fost implementate.

Capitolul 4. Analiză şi Fundamentare Teoretică

Acest capitol cuprinde cerințele sustemului, sunt prezentate lănga cerințele funcționale și cele nonfuncționale, arhitectura conceptuală a sistemului, cazurile de utilizare, precum și detaliile tehnologiilor care alcătuiesc baza și tehnologiile care au fost utilizate la realizarea aplicației.

4.1. Cerințele sistemului

Cerințele sistemului pot fi clasificate în două categorii de cerințe, prima este categoria cerințelor funcționale şi a doua este categoria cerințelor non-funcționale. Cerințele sistemului sunt prezentate conform acestor categorii.

4.1.1. Cerințe funcționale

Cerințele funcțional sunt cerințele care trebuie să fie îndeplinite de către sistem. Aceste cerințe cuprinde funcționalitățile aplicației. Cerințele funcționale ale sistemului au fost stabilite în urma analizării nevoilor al unui sistem minimal de monitorizare meteorologică atât și în urma analizei sistemelor similare. Aplicația are următoarele cerințe funcționale:

Tabel 4.1 Cerințele funcționale al aplicației

Identificator	Descrierea cerinței funcționale
CF1	Aplicația trebuie să afișeze o hartă care prezintă datele meteorologice.
CF2	Aplicația trebuie să permită utilizatorilor să modifice poziția pe hartă în direcțiile stângă, dreapta, sus și jos
CF3	Aplicația trebuie să permită utilizatorului să mărească sau să micșoreze bucata de hartă afișată, să modifice poziția pe axa z.
CF4	Aplicația trebuie să se afișeze datele meteorologice pe hartă în funcție de coordonatele date.
CF5	Aplicația trebuie să grupează informația pe hartă
CF6	Utilizatorul trebuie să caute o locație pe hartă
CF7	Aplicația trebuie să afișeze informațiile meteorologice pentru locația căutată de către utilizator.
CF8	Aplicația trebuie să permită utilizatorului să selecteze dintre diferite informații meteorologice care dorește să fie afișat pe hartă.
CF9	Aplicația trebuie să salveze locațiile căutate de către utilizator.
CF10	Aplicația trebuie să afișeze pe hartă locațiile căutate de către utilizator.

4.1.2. Cerințe non-funcționale

Cerințele non-funcționale sunt cerințele sistemului care nu reglementează calitativ funcționalitățile aplicației, design-ul au scopul de a oferi funcționalități cât mai bune în contextul dat de execuție.

În următorul tabel sunt expuse cerințele non-funcționale al aplicației.

Tabel 4.2 Cerințele non-funcționale al aplicației

CNF1	**Extensibilitate**	Este posibilă extinderea aplicației, se pot adăuga noi funcționalității la codul de JavaScript, parte de PHP este responsabil numai la traficul de date intre JavaScript și baza de date.
CNF2	**Performanța**	Aplicația este rapidă, fiind concepută similar aplicațiilor studiate, clusterizarea este operația cea mai costisitoară. Timpul de răspuns depinde de performanța calculatorului client, majoritatea calculelor se rulează în partea de client.
CNF3	**Securitate**	Autentificarea și înregistrarea sunt securizate.
CNF4	**Utilizabilitatea**	Interfața utilizator este simplă, familiar, daja văzut la alte aplicații populare.
CNF5	**Accesibilitatea**	Fiind o aplicație web, poate fi accesat oriunde. Utilizatorul are nevoie doar de conexiune la internet.
CNF6	**Disponibilitate**	Fiind o aplicație web este disponibil 24 de ore. Serverul pe care e instalat sistemul, funcționează constant.
CNF7	**Scalabilitate**	Performanța sistemului se menține și în condițile creșterii numărului de utilizatori..

Extensibilitatea este proprietatea sistemului care măsoară cât de ușor se pot adăuga funcționalități noi la aplicație. Deoarece aplicația este construită modular, se pot adăuga funcționalități fără a modifica structura existentă.

Performanța se măsoară în timpul de răspuns al aplicației. Aplicația conține două operațiuni mai costisitoare, încărcarea datelor meteorologice, care se face numai odată, după încărcarea paginii web și clusterizarea, care se execută la fiecare modificare asupra hărții, această operație trebuie să prelucreze foarte multe informații instant, răspunsul are un delay, dar nu este semnificativ.

Securitate: Aplicația nu operează cu informații secrete. Autentificarea și înregistrarea utilizatorilor este securizat, nu se poate atacat pagina prin cross side scripting. Accesul la resursele meteorologice sunt securizate cu cheie de API primit de la ECMWF, care nu este accesibil din afara serverului.

Utilizabilitatea sistemului este dată de ușurința cu care acesta poate fi învățat și utilizat. Din punctul de vedere al utilizatorului aplicația este foarte simplă, conține o singură pagină, toate operațiile pot fi făcute în aceași pagină. Interfata de utilizator este

simplă și ușor de înțeles, este familiară și deja văzut la mai multe aplicații populare. Aranjarea elementelor pe pagină permită utilizatorului să vadă ce dorește dar și să se găsească toate informațiile ușor.

Accesibilitatea reprezintă simplicitatea de a accesa aplicația. Fiind o aplicație web se poate fi accesat foarte ușor prin intermediul unui browser web, așa utilizatorul are nevoie doar conexiune la internet și un browser web pentru a putea folosi aplicația.

Disponibilitate de obicei este masurată în procente și reprezintă timpul în care aplicația este funcțională. Apliccația este disponibil 24 de ore. Actualizarea datelor meteorologice se rulează odată pe zi, iar frecvența actualizării poate fi modificat foarte usor prin reconfigurarea cron-ului.

Scalabilitate este abilitatea al unui sistem ca să se evolueză în funcție de numărul de utilizatori crescute. Aplicația execută foarte puține operații pe server, deci numărul de utilizatorii nu poate fi o problema, iar datele trimise pot fi pregătite în memorie deoarece sunt actualizate o dată pe zi, iar cache-ul poate fi actualizat dupa actualizarea datelor meteorologice.

4.2. Arhitectura sistemului

Arhitectura sistemului este bazată pe modelul client-server. Acest tip de arhitectură implică o aplicație distribuită, aplicația conține resurse de servicii care se numesc servere, și părți care folosesc aceste servicii, numit clienți. De obicei clienții si serverele sunt alocate pe hardware-le separate și se comunică între ele printr-o rețea de calculatoare.

Arhitectura conceptuală a aplicației este următoarea:

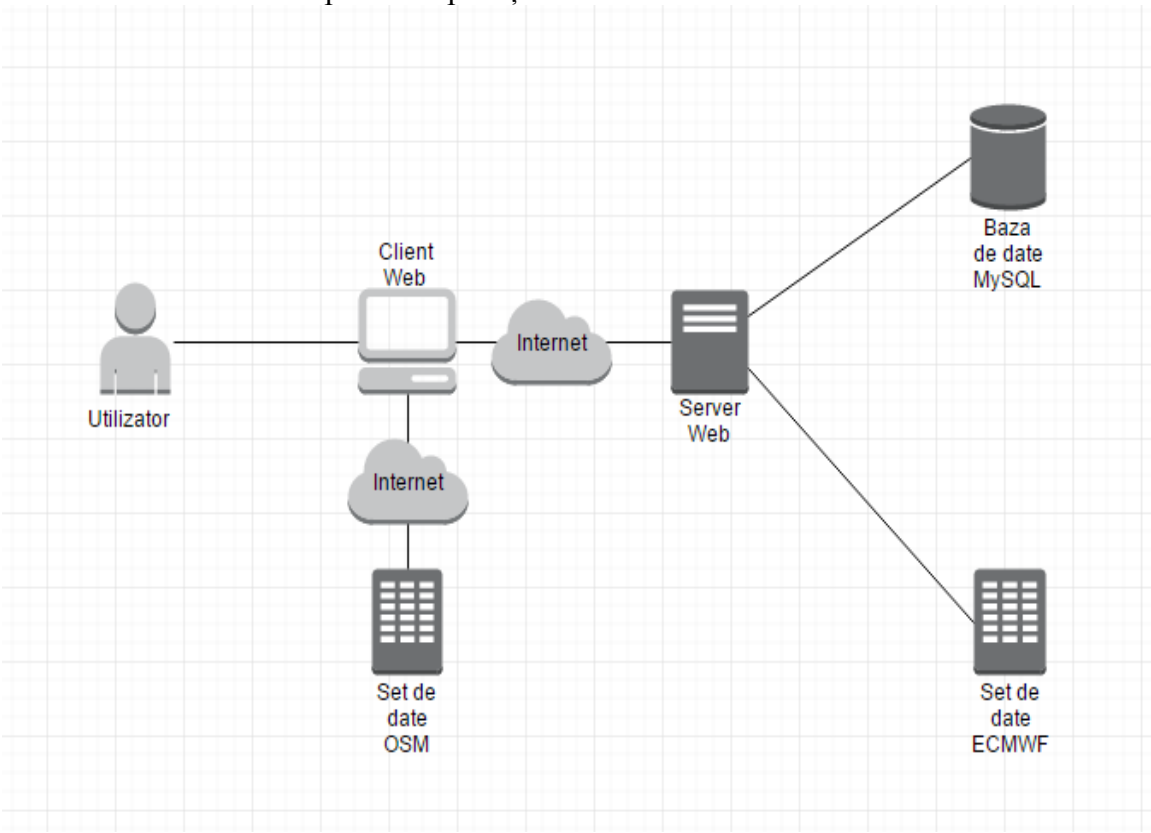

Figura 4.1: Arhitectura conceptuală a sistemului

În figura de mai sus se vede că în acest sistem clientul este browser-ul ales de către utilizatorul aplicației pe calculatorul de client. În direcția clientului informațiile sunt furnizate de către aplicația de server și de către un set de date extern care trimite țiglele, bucățile de hară, care sunt afițate în browserul clientului. Clientul comunică cu serverele prin internet utilizând protocolul de HTTP [2].

Set-ul de date OSM îndeplinește rolul de server în acest sistem, set-ul de date OSM furnizează imaginile care construiesc harta în bucăți, layer-ul de comunicare între OSM și browse-ul client este librăria Leaflet, care ajută și în afițarea hatrei. Setul de date OSM poate furnizează datele geografice și în format GeoJson. La orice modificare al clientului pe harta OSM se face o înterogare către setul de date OSM, și cu ajutorul librăriei Leaflet se redesenează harta [16].

Aplicația de pe server are două părți, una furnizează datele meteorologice către browser-ul client în format Json prin protocolul HTTP. A doua parte al aplicației server se apdatează și salvează datele meteorologice în baza de date relațională. Applicația trimite cereri către furnizorul ECMWF prin ănterogării MARS. În această situație applicația de pe server pot fi considerat ca un client, al serverului ECMWF.

Set-ul de date ECMWF furnizează datele meteorologice în format Grib sau Grib2 prin Grib API, acest intermediu pot fi înterogat folosind înterogări MARS, care poate converti datele în formatul NetCDF. Din punct de vedere arhitectural setul de date ECMWF este un furnizor de date meteorologice deci un server. Serviciul ECMWF furnizează și coordinate împreună cu datele meteorologice, prin coordinate, sunt conectate datele OSM și informațiile meteorologice, în nivelului aplicației de client [14].

Pentru salvarea datelor meteorologice în baza de date relaționale MySQL aplicația de pe server folosește librăria pygrib cu care reușește să se decodeze datele din fișierul Grib și să se trimite la baza de date MySQL, după acest pas baza de date poate furniza datele meteorologice actualizate către procesul care trimite datele la aplicație client.

4.3. Cazuri de utilizare

În acest sub-capitol sunt prezentate toate cazurile de utilizare ale utilizatorul sistemului. O reprezentare grafică al cazurilor de utilizare este diagrama cazurilor de urtilizare pe care sunt prezentați actorii sistemului și acțiunile prin care utilizatorul poate interacționa cu sistemul.

În aplicația prezentată avem utilizatori autentificați și neautentificați, utilizatorii autentificați au toate drepturile de utilizare pe care le au utilizatorii neautentificați. Deci din punct de vedere al cazurilor de utilizare, cazurile de utilizare cu actorul autentificat include cazurile de utilizare cu actorul neautentificat.

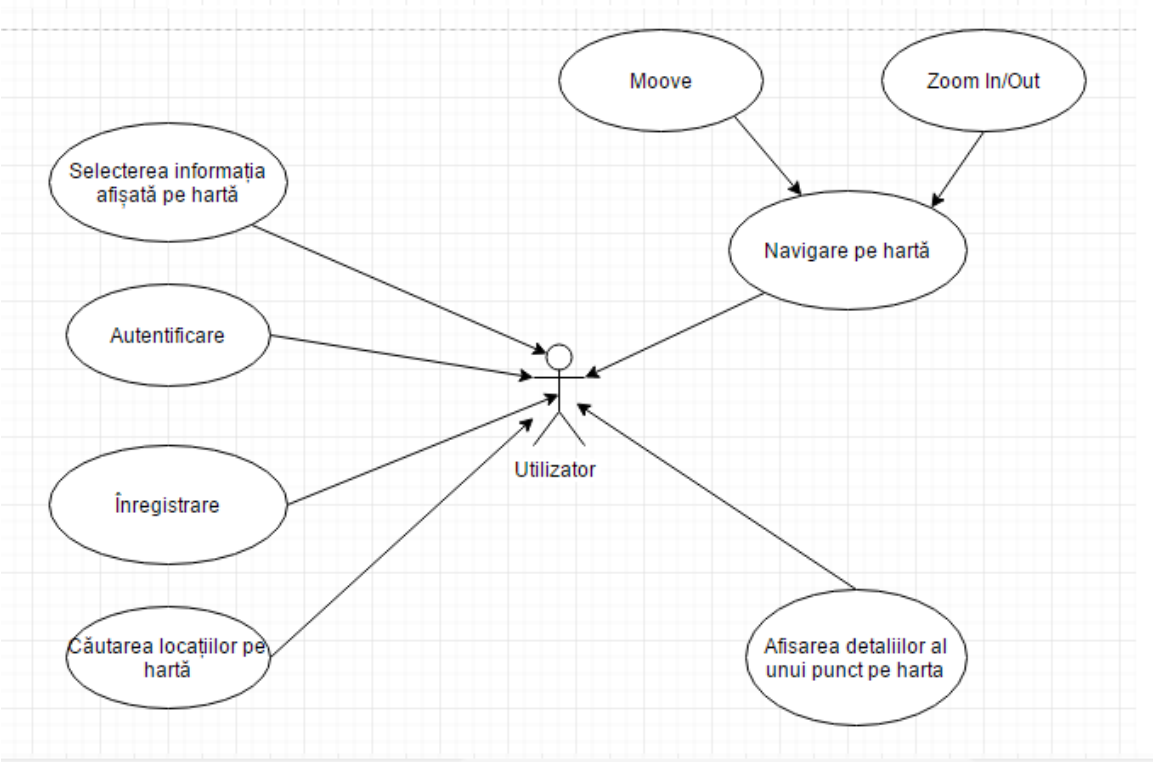

Figura 4.2 Diagrama cazurilor de utilizare

În continuare urmează prezentarea detaliată al cazurilor de utilizare ale sistemului, pentru applicația de client.

CU1

Numele cazului de utilizare: Înregistrare în sistem
Actor Principal: Utilizator neautentificat.
Părți interesate: Utilizatorul neautentificat doreste să înregistreze în sistem pentru a avea acces la mai multe detalii meteorologice.
Precondiții: Aplicația trebuie sa fie încărcată în browser.
Post condiții: Utilizatorul va fi înregistrat în applicație.
Scenariu de succes:

1. Utilizatorul apasă butonul de login.
2. Utilizatorul trece la secțiunea de Register în fereastra de pop-up.
3. Utilizatorul introduce datele necesare pentru înregistrare, numele, prenumele, numele de utilizator, parola și adresa de email.
4. Utilizatorul apasa butonul de Sign Up și datele vor fi salvate în baza de date.
5. Utilizatorul are acum posibilitatea de a se autentifica în aplicație.

Eșecuri:

1. Utilizatorul nu are permisiuni de conectare la Internet.
2. Datele introduse sunt eronate
    - Sistemul nu lasă utilizatorul sa se înregistreze fără să completeze toate câmpurile.
    - Utilizatorul trebuie să introduce un nume de utilizator care nu există deja în baza de date.

- Email-ul introdus este verificat dacă este o adresă de email validă.
- Numele și prenumele introduse pot conține numai litere.

3. Serverul nu este accesibil dintr-un oarecare motiv.

CU2

Numele cazului de utilizare: Autentificare în sistem
Actor Principal: Utilizator neautentificat.
Părți interesate: Utilizatorul neautentificat dorește să acceseze contul său, ca să vadă detaliile meteorologoce.
Precondiții: Aplicația trebuie să fie încărcată în browser.
Utilizatorul trebuie să fie înregistrat în sistem.
Post condiții: Utilizatorul este autentificat în aplicație.
Scenariu de succes:

1. Utilizatorul apasa butonul de LogIn.
2. Apare o fereastră pop-up în mijlocul ecranului.
3. Utilizatorul întroduce numele de utilizator și parola.
4. Utilizatorul apasă butonul de Log In.
5. Sistemul validează datele introduse de către utilizator.
6. Sistemul salvează cookie-ul in browser și reține autentificarea.
7. Sistemul reîncarcă pagina automat.

Eșecuri:

1. Utilizatorul nu au permisiuni de conectare la Internet.
2. Datele eronate sunt eronate
    - Sistemul nu lasă utilizatorul sa se înregistreze fără să completeze toate câmpurile.
    - Numele de utilizator și parola introdusă trebuie sa corespundă cu datele din baza de date.
3. Serverul nu este accesibil dintr-un oarecare motiv.

CU3

Numele cazului de utilizare: Selectarea informației afișată pe harta
Actor Principal: Utilizator autentificat
Părți interesate: Utilizatorul autentificat dorește să se selecteze proprietatea meteorologică afișată pe hartă.
Precondiții: Utilizatorul trebuie să fie autentificat în sistem.
Post condiții: Utilizatorul va schimba layer-ul de markere afișat pe hartă.
Scenariu de succes:

1. Utilizatorul apasă butonul de layer control din colțul dreapta în sus.
2. Controlerul de layere se extinde, opțiunile devin vizibile.
3. Utilizatourl selectează opțiunea dorită de pe lista informațiilor meteorologice.
4. Proprietatea meteorologica selectată va fi afișată instant pe hartă.

Extensii:

1. Unitatea de măsură este afișată pe lista proprietăților.
4. Fiecare proprietatea meteorologică este marcat cu markere cu culori diferite.

Eșecuri:

1. Utilizatorul nu are permisiuni de conectare la Internet.

2. Datele meteorologice nu au fost încărcate în aplicație dintr-un oarecare motiv.
3. Serverul nu este accesibil dintr-un oarecare motiv.

CU4

Numele cazului de utilizare: Căutarea locațiilor pe hartă
Actor Principal: Utilizator autentificat
Părți interesate: Utilizatorul autentificat dorește să verifice starea meteorologică la o locație dorită.
Precondiții: Utilizatorul trebuie să fie autentificat în sistem.
Post condiții: Utilizatorul va primi informațiile meteorologice la locația căutată.
Scenariu de succes:

1. Utilizatorul începe să introducă numele locației dorite.
2. Motorul de căutare va oferi o listă de sugestii.
3. Utilizatorul selectează locația dorită.
4. Aplicația plasează un marker nou, pe hartă.
5. Aplicația asignează detaliile meteorologice al marker-ului cel mai apropiat.

Extensii:

4.Aplicația focusează la locația căutată.
5.Fiecare locație căutată rămâne pe hartă până reâncărcarea pagini.
4.Locația căutată va fi salvată în baza de date.

Eșecuri:

1. Utilizatorul nu are permisiunea de conectare la Internet.
2. Aplicația nu poate conecta la motorul de căutare.
3. Aplicația nu reușește să salveze locația căutată.
4. Utilizatorul nu mai este autentificat în sistem.
5. Serverul nu este accesibil dintr-un oarecare motiv.

CU5

Numele cazului de utilizare: Afișarea detaliilor unui punct de pe hartă
Actor Principal: Utilizator autentificat
Părți interesate: Utilizatorul autentificat dorește să află mai multe informații despre un punct specific de pe hartă.
Precondiții: Utilizatorul trebuie să fie autentificat în sistem.
Post condiții: Vor fi afișate detaliile al punctului specific într-o fereastră pop-up.
Scenariu de succes:

1. Utilizatorul selectează punctul de pe hartă
2. Apare o fereastră pop-up, în care vor fi afișate informațiile detaliate.

Eșecuri:

1. Utilizatorul nu are permisiuni de conectare la Internet.
2. Datele meteorologice nu sunt încărcate corect în aplicație
3. Serverul nu este accesibil dintr-un oarecare motiv.

CU6

Numele cazului de utilizare: Navigare pe hartă
Actor Principal: Orice utilizator al aplicației
Părți interesate: Utilizatorul dorește să schimbă partea vizibilă din hărtă în fereastra de browser.
Precondiții: Aplicația trebuie să fie încărcată în browser.
Post condiții: Utilizatorul va vedea bucata dorită de pe hartă.
Scenariu de succes:

1. Utlizatorul efectuează scroll sau click and drag cu mouse-ul pe hartă
2. Partea afișată pe hartă va fi schimbată conform acțiunile utilizatorului.

Extensii:

1.Markerele afișate pe hartă vor fi re-clusterizate automat.

Eșecuri:

1. Utilizatorul nu are permisiuni de conectare la Internet.
2. Aplicația nu reușește să descarcă tile-urile de hartă.
3. Serverul nu este accesibil dintr-un oarecare motiv.

4.4. Fundamentare teoretică

4.4.1. OpenStreetMap

Open Street Map (OSM) [23] este un project colaborativ pentru a crea o hartă editabilă și gratuită. Creșterea projectului OSM a fost motivat de restricțiile de folosire al informațiilor de hartă în marea parte al lumii. Datele de hartă al Open Street Map au fost colectate cu ajutorul voluntariilor folosind sisteme de GPS personale, calculatoare portabile, camere digitale și dispozitive de înregistrare voce. Datele au fost trimise la baza de date OpenStreetMap.

Proiectul OpenStrretMaps a fost lansat in iulie al anului 2004, iar de la 2006 januarie era posibil editarea publica a datelor OSM. Dupa sase ani, numarul contribuitorilor a crescut peste 1 milioane de utilizatori, iar numărul punctelor de interes au crescut peste 3 miliarde cu 2 miliarde de noduri, relații și căi, marimea datelor este peste 300 de GB.

4.4.1.1.Istoria OSM

OpenStreetMaps a fost creat în iulie 2004 de către Steve Coast. Steve nu a înteles că Ordnance Survey-ul de ce a creat seturi de date massive dar nu a diztribuit gratuit cu cei care au platit pentru crearea acestora. Steve a început un proiect de mapare care ne oferă datele gratuit. OpenStreetMap-ul nu este o hartă, este o bază de date, harta online este reprezentarea datelor, nu contine imagini de satelit [8].

În ianuarie 2006 a apărut prima versiune a editorului offline JOSM care a ajutat contribuitorii să editeze datele de OSM, majoritatea contribuitorilor sunt utilizatori noncomerciali.

In 2007 a apărut prima versiune a editorului Potlach, care are o interfață online, avantajul este că utilizatorul nu trebuie să retină baza de date totală locală.

In 2009 aprilie OpenStreetMap s-a schimbat la technologia API cu versiunea 0.6.

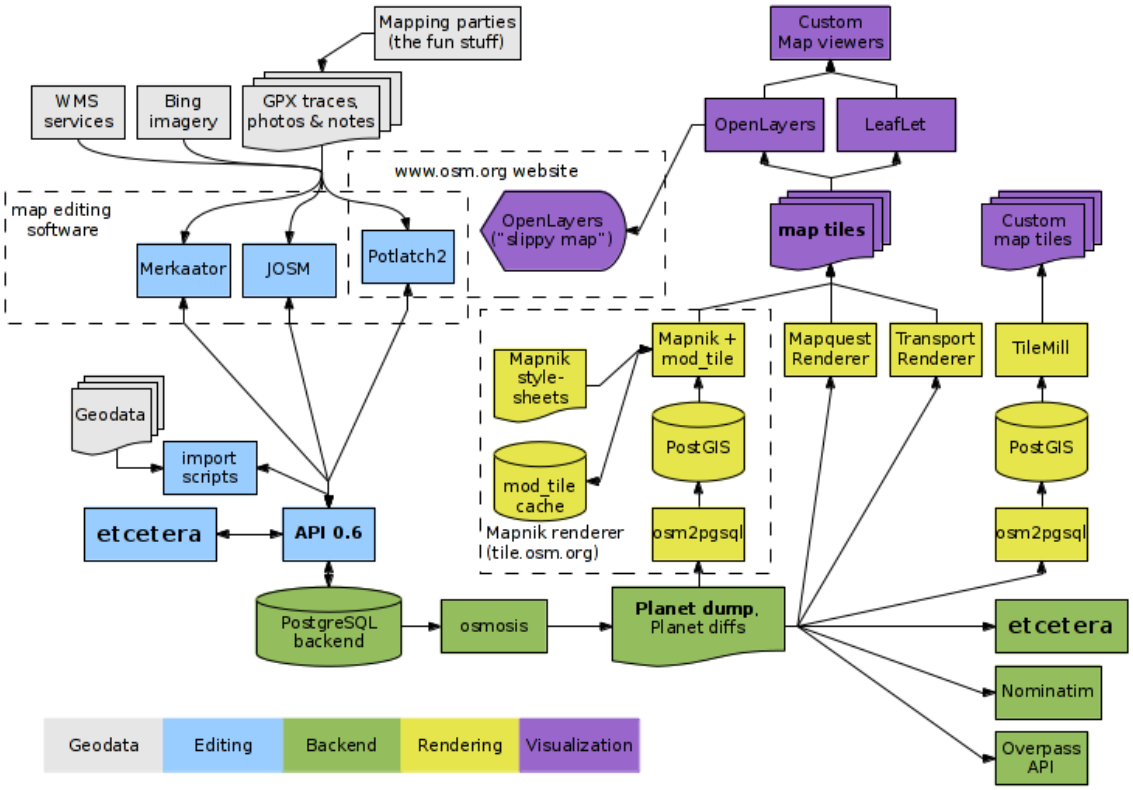

Figura 4.3 Componentele OSM [22]

4.4.1.2. Modelul de date

Componentele setului de date OSM sunt noduri, căi, relatii și taguri.

Nodurile, elementele de baza, sunt definite prin latitudinea și longitudinea geografică. Nodurile descriu punctele de interes (POIs). În afară de utilizarea lor la drumuri, mai sunt utilizate pentru reprezentarea obiectelor fără mărime ca și vârfuri de munte.

Căile sunt conexiuni ordonate de noduri. Caile pot fi deschise: obiecte lineare ca si drumuri, căi ferate și inchise: arii. Sunt folosite pentru reprezentarea obiectelor lineare ca și drumuri, râuri și arii ca și păduri, lacuri sau parcări.

Relatiile sunt grupuri de căi și noduri cu roluri asociate. Attribute generice: uid, user, timestamp, visible, version, changeset; un set de tag-uri:

Un tag este o pereche de cheie și valoare. Sunt folosite pentru stocarea datelor meta despre obiectele de pe hartă ca și tipul, numele și alte proprietăți. Sunt tot timpul atașate la un obiect nod, cale sau relație, un tag nu poate exista în sine.

Primitivele OSM sunt stocate si procesate în formate diferite. Datele principale de OSM sunt stocate in baza de date principala al OSM. Baza de Date principală este o bază de date PrstgreSQL, care conține o tabelă pentru fiecare tip de primitive, cu obiecte individuale stocate pe rânduri. Fiecare editare se întâmplă în această bază de date și toate celelalte formate ale datelor OSM sunt create din această bază de date.

Pentru transferul de date, sunt create mai multe exportări ale acestei baze de date, care sunt valabile pentru descărcare. Exportul complet se numeste planet.osm. Există doua tipuri de extensii, în format XML sau Protocol Buffered Binary Format (PBF) [22].

4.4.1.3. Licența deschisă

Ideea în spatele OpenStreetMap este sa ne ofere date geospațiale gratuite și deschise [23].
Cele mai multe seturi de date geografice sunt eliberate sub licența proprietară, și în cazul în care este gratuită (cost-less), nu poate fi exploatată gratuit.

Licența gratuită care este identic folosită la entități intelectuale (muzică, cărți, poze, informatii). Licența Creative Commons adaptată la arte, are patru optiuni: Attribution(BY), Non Commercial(Nc), No Derivatives(ND), Share Alike(SA) și șase combinații posibile: CC-by, CC-by-sa, CC-by-nc, CC-by-nc-nd, CC-by-nc-sa, CC-by-nd.

Licența în care OpenStreetMap este publicată este Open Data Commons Open Data Base License (ODbL). Acesta prevede că, indiferent de manipulările făcute în baza de date OSM originală, versiunea manipulată, de asemenea, să fie publicată în conformitate cu ODbl. Produsul rezultat poate fi tratat în mod convenabil de creator (chiar ca un produs comercial) atâta timp cât nu seamănă cu o bază de date și baza de date părinte al acestui produs este publicată. Cartografia bazată pe bucăți de hartă încărcate independent, și documentația sunt licențiate sub licența Creative Commons Attribution-ShareAlike 2.0 (CC by-sa).

4.4.1.4. Instrumentele de editare

Există diferite tool-uri care permit contribuitorilor să editeze datele OSM Potlach este un editor online, scris in Flash, se poate rula din browser, este simplu de utilizat, recomandat pentru începători. JOSM - Java OpenStreetMap Editor este o aplicație desktop scrisă în Java, cu arhitectura plugin, este recomandat petru utilizatori mai avansați, oferă un set mare de funcționalități [22].

4.4.2. ECMWF și MARS

Europen Centre pentru Medium-Range Weather Forecasts (ECMWF) [13], este o organizație interguvernamentală suportată de cele mai multe națiuni ale Europei care e bazat în Shinfield Park, Reading, Regatul Unit al Marii Britanii. Se operează unul dintre cele mai mari complexe de supercomputer în Europa și cea mai mare arhivă a datelor meteorologice în format numeric.

ECMWF a fost stabilit în anul 1975, după recunoașterea nevoii unui set de resurse tehnice și stințifice a servicilor meteorologice în Europa. ECMWF a fost început cu 22 de țări, după anul 2010 încă 4 țări au intrat.

ECMWF furnizează o prognoză globală înainte la 15 zile și progzonă sezonală 12 de luni. Produsele lor sunt furnizate la servicile naționale de vreme. Missiunea lui ECMWF este producerea prognozelor meteorologice și monitorizarea Earth-system, furnizarea cercetăriilor stiințifice și tehnice ca să îmbunătățească prognozele, arhivarea datelor meteorologice.

Predicțiile numerice de vreme culeg datele meteorologice de la sateliți și alte sisteme de observare ca și stații meteorologice, baloane meteorologice. Aceste date sunt folosite la producerea uei stări inițiale al modelului atmosferii, din care un model atmosferic este folosit la prognozele meteorologice.

4.4.2.1. MARS

Meteorological Archival and Retrieval System (MARS) [14] al ECMWF. Sistemul MARS este bazat pe o architectură de client-server. Wqb-MARS este un set de aplicații web prin care putem accesa datele din MARS cu ajutorul unui browser. Catalogul MARS reprezintă starea contentului din MARS, fiecare câmp meteorologic poate fi accesat și descărcat de aici.

Cand cautăm în catalogul MARS când am ajuns la pasul final unde alegem datele, trebuie să alegem din opțiunile verificarea valabilității datelor, vizualizarea înterogării MARS, estimarea volumul de descărcare sau, descărcarea datelor selectate in format GRIB sau NetCDF.

Sistemul MARS răspunde la mii de înterogări în fiecare zi. Durata descărcării datelor depinde de numarul înterogărilor concurente.

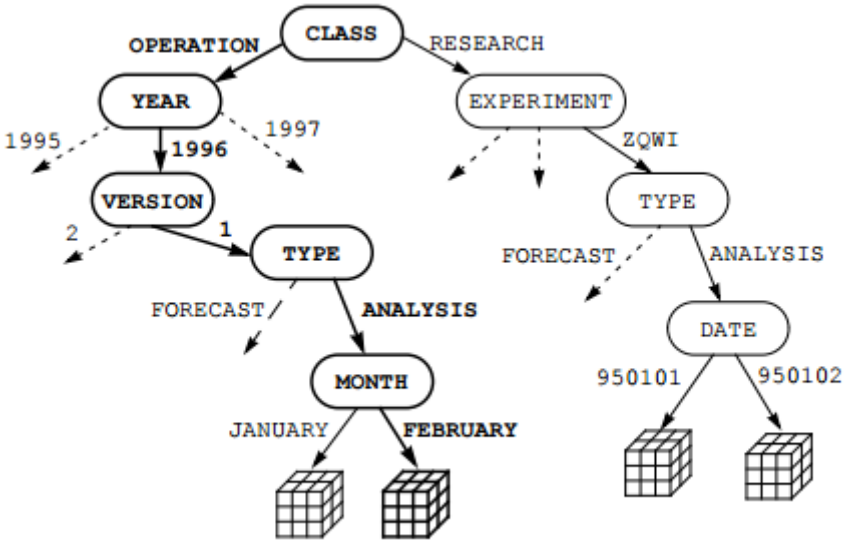

Figura 4.4 Arborele înterogării MARS [17]

4.4.2.2. Produse ECMWF

Lângă datele disponibile de la MARS, utilizatorii trebuie să urmărească activitățiile lui ECMWF. În acest sub-capitol sunt prezentate produsele cele mai populare al lui ECMWF.

Datele operaționale produse zilnic de către ECMWF cuprind modele atmosferice, modele de vale, sisteme de ansamblu, condiții de limită, ansamblu de Multy-Analysis, prognoză sezonală, prognoză lunară, analiza oceaniilor.

Seturile de date climatologice lunare este o archivă al datelor din modelele de vale și atmosferă lunară. Reprezentarea arhivei poate varia în funcție de practica operațională al ECMWF.

ECMWF reanalyses este reanalizarea prognozelor utilizat pentru educație, cercetare și aplicații comerciale. Seturile de date reanalizate disponibile conțin analize, prognoze și acumlări de prognoze ca ieșire de la modele atmosferice. Experimentele de cercetare sunt generate zilnic conținând IFS (Integrated Forecast System) de către ECMWF.

Datele de input la modele ECMWF sunt următoarele câmpuri care conțin o selecție de produse, observații. Observațiile sunt folosite ca intrări la sistemul de asimilare și sunt arhivate în MARS, la fel sunt construite si feedback-urile. [10]

4.4.3. Algoritmi de clusterizare

Clusterizarea poate fi considerată una dintre cea mai importantă problemă de învățare, deci ca și orice altă problemă de acest tip, se ocupă cu găsirea unei structuri într-o colecție de date nemarcate. O definiție a clusterizării poate fi ”procesul organizării obiectelor în grupuri al căror participanți sunt într-un fel similari”. Un cluster este o colecție de obiecte care sunt similare înre ele și nu sunt similare cu obiectele din alte clustere sau grupuri.

Scopul clusterizării este să se determine gruparea intrinsecă al unui set de date nemarcate. Nu există o criterie optimă care pot fi independent de scopul final al clusterizării, acest criteriu trebuie definit de utilizator ca rezultatul clusterizării îndeplinească nevoile [3].

Algoritmii de clusterizare pot fi aplicați în mai multe arii, ca de exemplu: marketing, biologie, librării, asigurări, city-planning, studierea cutremurelor sau aplicații web.

Algoritmii de clusterizare pot fi clasificați următoarele tipuri: clusterizare exclusivă, clusterizare cu suprapunere, clusterizare ierarhică și clusterizare probabilistică. În primul caz datele sunt grupate într-un fel exclusiv, deci dacă un obiect poate fi asociat numai la un singur cluster. Al doilea tip de clusterizare este opusul primului tip. Algoritmii ierarhici sunt bazați la uniunea între două grupuri vecine, la început fiecare obiect este un cluster, după câteva iterații ajunge la densitatea dorită [24].

Algoritmul folosit în proiect este algoritmul K-Means [6], pentru că grupează datele în funcție de distanța între ele, care este cel mai relevant în cazul hărților. Algoritmul acceptă două intrări, datele însuși și ”k” numărul clusterelor. Ieșirea algoritmului este k clusters. Algoritmul folosește distanța pentru calcularea similarității. Fiecare cluster are un centroid, este un punct care reprezintă clusterul cel mai semnificativ.

Algoritmul de clusterizare K-Means are următorii pași:

1. Se alege k elemente aleator și definește ca centroide inițiale.
2. Pentru fiecare punct se caută centroid-ul cel mai apropiat și asociază punctul la clusterul cel mai apropiat.
3. Apdatează centroid-ul pentru fiecare cluster. Metoda populară este media punctelor din cluster.
4. Se repetă punctele 2. și 3., până ce nici un punct nu își schimbă cluster.

4.5. Tehnologii și protocoale utilizate

Aplicația va fi dezvoltată in limbajele de programare JavaScript și PHP, aplicația se bazează pe setul de date Open Street Map, ce este accesat prin librăria Leaflet pentu JavaScript bazat pe HTML5 si CSS3 care este similar cu librariile OpenLayers sau Google Maps API, este unul dintre cea mai populare librării de mapare pentru limbajul JavaScript. Pentru accesarea datelor meteorologice este folosit un server de Apache MySQL, și se folosește Ajax pentru comunicarea cu aplicația client. Ca și mediu de

dezvoltare este utilizat editorul WebStorm care este un IDE foarte puternic pentru html, javascript, css și o gama largă de tehnologii web moderne. Aplicația de actualizare al datelor meteorologice este dezvoltată în limbajul de programare Python, aceasta aplicație folosește extensiile ECMWF.

4.5.1. Limbajul JavaScript

JavaScript [30] este un limbaj dinamic, slab tipizat și este bazat pe prototipuri. În cazul acestui limbaj funcțiile implemenatate în cod sunt tratate ca și simple obiecte. Acest limbaj de programare suportă mai multe paradigme de programare, precum: paradigma funcțională, imperativă, dar și cea orientată pe obiecte. O proprietate importantă pentru acest limbaj este tratarea funcțiilor ca și obiecte de primă clasă. Orice variabilă care nu prezintă un tip de date primitiv este un obiect sau o funcție. Mai mult, un obiect poate fi schimbat inclusiv după ce a fost definit sau instanțiat.

JavaScript suportă tipuri de date primitive, ca și: number, string, boolean, undefined și null. Spre deosebire de alte limbaje de programare putem observa că JavaScript folosește un singur tip pentru reprezentarea numerelor. Tipul undefined este atribuit unei variabile care nu a fost definită, iar tipul null se folosește atunci când o variabilă a fost definită dar nu a fost încă inițializată. Pentru orice variabilă definită în cod JavaScript dacă aceasta nu reprezintă nici unul dintre tipurile de date primitive amintite mai sus, aceasta va fi tratată ca și obiect. Din cauza caracterului slab tipizat, orice variabilă de tip primitiv își poate schimba tipul.

Funcțiile din JavaScript sunt considerate obiecte de primă clasă, tipul lor fiind function. Limbajul permite și specificarea funcțiilor anonime, care nu au nume și pot fi referențiate printr-o variabilă, sintaxa acestora fiind function() {...}. În JavaScript tehnica apelurilor callback este folosită prin transmiterea unei funcții anonime ca parametru unei alte funcții.

JavaScript este un limbaj multi-paradigme: scripting, orientat pe obiecte, imperativ, dar și funcțional. Pentru aplicația prezentă am folosit mai mult paradigmele de scripting, dar și cea funcțională, întrucât prezintă funcții de înaltă clasă. Paradigma imperativă este îndeplinită prin faptul că avem o sintaxă asemănătoare Java pentru unele funcționalități. În acest limbaj funcțiile sunt evaluate la rulare, moment în care ne bazăm pe posibilitatea de care dispune de a importa și de a include anumite script-uri.

JavaScript este un limbaj de scriptare, publicat in anul 1995 pentru laici ca să extindă paginile de HTML cu efecte. De atunci JavaScript a devenit un limbaj foarte puternic și des folosit.

*Argument technologic.*Am ales această technologie pentru că este limbajul cel mai comun folosit pentru aplicațiile de client.

4.5.2. Protocolul HTTP

HTTP (HyperText Transfer Protocol) [2] este un protocol de tip text folosit cu precădență pentru a accesa resursele din Internet păstrate în rețeaua www . HTTP este un protocol de aplicație folosit pentru sisteme informatice distribuite și colaborative. Mai mult, acest protocol a fost fondatorul comunicării din rețeaua World Wide Web. Dacă în zona de adrese a browserului nu specificăm nimic, atunci protocolul implicit va fi considerat cel de HTTP. Cu ajutorul tehnicii de comunicare pe care o pune la bază protocolul putem transmite pagini web între două dispozitive.

Atunci când accesăm un link și protocolul principal este cel de HTTP sau este considerat ca și implicit, în zona de adresă a browserului putem împărți acea adresă după cum urmează: prefixul www, care este non-funcțional, folosit doar ca o convenție pentru a accentua folosirea protocolului HTTP, porțiunea dintre www și până la denumirea nivelului de nivel superior (de ex .com), care este adresa dispozitivului gazdă (host) și domeniul de nivel superior (.com). Ceea ce urmează după aceste entități reprezintă o cale locală pentru acel dipozitiv constituită din parametrii cererii HTTP. Această cale poate conține diferiți marcatori, precum '?' care specifică un string query folosit pentru transmiterea informațiilor dintr-o pagină web în alta, sau '&' pentru delimitarea parametrilor. În funcție de metoda HTTP folosită acești paramentrii pot să apară sau nu în link.

Metodele folosite pentru transmiterea unei cereri HTTP sunt:

GET – aceasta este metoda cea mai des utilizată și este apelată atunci când utilizatorul îi cere serverului o anumită resursă sau informație. Pentru această metodă parametri amintiți mai sus sunt atașați în link, începând cu marcatorul '?' și avem perechi de parametri sub forma "nume_parametru=valoare", separate prin marcatorul '&'.

PUT – metodă folodită pentru a salva anumite documente pe server;

HEAD – metodă asemănătoare cu metoda GET, însă în cazul acestei metode serverul returnează doar antetul resursei;

POST – este o metodă care se folosește pentru a trimite parametrii în corpul mesajului și nu în link, trimițând date către server. Această metodă este mai des întâlnită atunci când folosim formulare, pentru datele utilizatorului de exemplu, întrucât aceste informații confidențiale, precum parolele, nu se pot trimite în link;

DELETE – este o altă metodă care se folosește pentru stergerea resurselor de pe server, acest lucru definind metoda opusă metodei PUT.

Pentru cererile pe care o face protocolul HTTP serverul se trimite un răspuns. Aceste răspunsuri au coduri dupa care se clasifică răspunsurile:

100 - codurile care incep cu cu cifra 1 se reprezintă cereri informaționale prin care anunță utilizatorul că poate să continue cererea sau că e în curs de schimbare a protocolului.

200 - orice răspuns al serverului începând cu cifra 2 inseamnă că cererea s-a efectuat cu succes.

300 - codurile care se începe cu cifra 3 indică faptul că va fi nevoie de o redirectare pentru a putea duce cererea la final.

400 - sunt coduri care indică mesaje de eroare ale utilizatorului, indică utilizatorului că va trebui să reia operațiile anterioare.

500 - categoria acesta indică o eroare de server, unde acesta nu se poate rezolva cererea, din cauza unei erori interne.

4.5.3. Ajax

Ajax (Asyncronous JavaScript and XML) [2] este un limbaj de programare care este folosit pentru schimbul de date cu un server și actualizarea unor părți din paginile HTML, fără reâncărcarea toată pagina. Ajax este o technica care gestionează dinamic și rapid paginile de web, și se bazează pe standardele de internet, precum obiecte de tip XMLHttpRequest, JavaScript, CSS, XML și combinația acestora.

În jurul anilor 1990, cele mai multe pagini de web au fost bazate pe put HTML, fiecare acțiune al utilizatorilor au incărcat o pagină complet de pe server. Aceasta procedură nu era eficientă. În 1996 a fost întrodus tag-ul iframe de către Internet Explorer. În anul 1998 grupul Microsoft Outlook Web App a inmplementat prima componentă XMLHTTP. În 1999, Microsoft a folosit technologia lor de iframe pentru a apdate conținutul paginilor in Internet Explorer pentru schimbarea datelor în mod dinamic.

Termenul Ajax a fost folosit începând de 18 februarie al anului 2005 de către Jesse James Garrett in articolul ”Ajax: A New Approach to Web Applications” bazat pe technici folosite pe paginile Google.

În 5 aprilie al anului 2006 asociația World Wide Web Consortium a lansat primul obiect XMLHttpRequest cu intenția de a crea un standard web.

*Argument technologic.*Am ales aceasta technologie pentru că prin apeluri Ajax nu trebuie încărcat toată pagina la fiecare modificare, și aplicația descrisă folosește foarte multe resurse.

4.5.4. MySQL

MySQL [12] este un sistem de management al bazelor de date relațională cu sursă deschisă. Numele sistemului vine din numele fondatorului Michael Widenius, și SQL este prescurtarea al Structured Query Language. Proiectul de dezvoltare MySQL a facut codul de sursa disponibil sub termenele licenței GNU General Public License. MySQL a fost spozorat și deținut de către o singură firmă suedeză numit MySQL AB, azi e deținut de către Oracle Corporation. Sunt disponibile mai multe versiuni cu plată pentru consum proprietar, aceste versiuni oferă extra funcționalități adiționale.

MySQL-ul este o componentă de baza al LAMP-ului. Aplicațiile care folosesc MySQL înclud: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB și Drupal. MySQL-ul este folosit ăn mai multe site-uri web de scală largă ca și Google, Facbook, Twitter, Flickr și ToyTube.

MySQL a fost creat de către compania suedesă MySQL AB, fundat de către David Axmark, Allan Larsson și Michael Widenius. Dezvoltarea originală de către Widenius și Axmark s-a început in anul 1994. Prima versiune a fost lansat la 23 mai al anului 1995. Inițial a fost creat pentru folosire personală.

MySQL este oferit sub două ediții diferite: MySQL Community Server și MySQL Enterprise Server. Versiunea de Enterprise se diferă printr-o serie de extensii specifice pentru această versiune.

Când se folosește un motor de indexare diferită decât InnoDB, MySQL nu suportă limbajul SQL standard pentru câteva funcționalități ca și chei străine și verificarea constrângerilor. La versiunile mai vechi ca MySQL 5.7, triggerele sunt limitate la o singură acțiune per cronometrare.

Mysqldump este un instrument logic de backup inclus și în ediția Community și în Enterprise. Suportă backup-ul din toate motoarele de depozitare in memorie.

*Argument technologic.*Aceasta technologie a fost ales pentru că este cel mai comun utilizat, și integrarea este simplă cu sistemul propriu, asigurăstocare și acces necesare sistemului și fără costuri.

4.5.5. Apache

Serverul Apache [31] de HTTP este web serverul cel mai popular in lume. Original era bazat pe serverul NCSA HTTTPd, dezvoltarea Apache-ului a inceput in 1995. Apache a jucat un rol de cheie la creșterea inițiala al World Wide Web-ului, a depășit foarte repede serverul NCSAHTTPd și a devenit serverul de HTTP dominant, este considerat ca serverul cel mai folosit de la 1996 aprilie. In 2009 a devenit primul software de web server care au servit mai mult decât 100 de millioane de site-uri.

Apache este dezvoltat de o comunitate deschisă de dezvoltatori sub institutul Apache Software Foundation. Software-ul este disponibil pentru toate sisteme de operare comune, ca și: Unix, Microsoft Windows, NetWare, OpenVMS, OS/2, etc. Apache este gratuit și cu sursă deschisă.

*Argument technologic.*A fost ales această server web pentru că este simplu și foarte ușor de configurat și utilizat pentru aplicația dezvoltată.

4.5.6. Libraria Leaflet

Leaflet [16] este o librarie de mapare pentru JavaScript, este disponibil și pentru hărți interactive și este mobile-friendly. Costă aproximativ 33 de KB de cod JavaScript arhivat, conține toate funcționalitățile de care majoritatea dezvoltatorilor au nevoie.

Leaflet a fost creat cu simplicitate, focusat la performanță și la cazurile de utilizare. Este compatibil cu majoritatea platformelor de desktop si mobil, folosește toate avantajele technologiilor HTML5 și CSS3 dar este accesibil și prin technologii mai vechi. Poate fi extins cu multe plugin-uri , are și un API bine documentat și simplu. Leaflet permite utilizatorii să afișeze hărți pe servere publice fără să folosească la bază un server GIS. Poate incărca datele in format GeoJSON și să crează leyer-e interactive, ca și exemplu: marker-e cu popup-uri.

Folosirea tipică a librăriei Leaflet este crearea un element de HTML ca și un div, unde pot fi adaugate markerele și layer-ele. Exemplu:

```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);
```

Figura 4.5 Adaugarea hărții Leaflet

Elementele significante de leaflet sunt:

- Tipuri de raster (TileLayer și ImageOverlay)
- Tipuri de vector (Path, Polygon și tipuri specifici ca si Circle)
- Tipuri grupate (LayerGroup, FeatureFroup si GeoJSON)
- Controale (Zoom, Layers).

*Argument technologic.*A fost ales librăria Leaflet pentru că este mai simplu și mai compact decât librăria OpenLayers, și îndeplinește cerințele aplicației.

4.5.7. MARS request

MARS-ul (Meteorological Archival and Retrieval System) [9] este sistemul de archivare și recuperare al datelor meteorologice al ECMWF-ului. Serviciul MARS asigură descărcarea și listarea datelor metorologice din afara facilității ECMWF.

În fiecare zi, ECMWF produce mai multe feluri de analize și prognoze globale care sunt arhivate în MARS. Comanda mars accepta un fișier de interogare ca parametru saupoate citi interogarea din inputul standard. Comanda mars emis in linia de comandă va afișa un mesaj de salutare in sistem. La acest punct putem întroduce liniile întrogării, sfârșitul fisierului este marcat cu <CTRL>-D. Interogarea datelor din linia de comanda nu este recomandat, numai pentru scopuri de test. Metoda recomandată pentru trimiterea întrogărilor este prin fișiere de interogare. Sintaxa întrogării mars este următoarea, trebuie specificat o acțiune peste unei set de câmp sau observații. Seturile de câmp sunt perechi de cuvânt cheie și valoare.

```
retrieve                              action
  class=od
  stream    = oper,                   identification of archive
  expver    = 1,
  date      = -1,                     date and time related
  type      = analysis,
  levtype   = model levels,
  levelist  = 1/to/91,
  param     = temperature,
  grid      = 2.5/2.5,                post-processing
  area      = Europe,
  use       = infrequent,             execution control
  target    = analysis.12             storage
```

Figura 4.6 Exemplu de interogare MARS [14]

Acțiunea din interogarea de mars poate fi retrive, compute, list, read și write. Seturile de câmp pot fi văzute ca variabile. Mars accepta întegogșri multiple într-un singur apel, couvintele de cheie sunt mostenite din prima înterogare.

Sistemul MARS este bazat pe arhirectura de client-server, prin care sistemul devine mai flexibil și adaptiv la schimbări într-us sistem complex ca ECMWF. Lângă formatul .grib sistemul mars poate salva datele în format NetCDF.

4.5.8. Limbajul PHP

Denumirea vine din acronimul recursiv al cuvintelor Hypertext și Preprocessor [18]. Este un limbaj open source des folosit pentru scopuri generale. Este un limbaj de scripting specific pentru dezvoltare web. Poate fi încorporat in HTML. In schimb cu multe comenzii care returnez HTML ca si C sau Perl, paginile php conțin HTML lângă codul PHP. Bucățile de cod PHP se găsesc între tagurile <?php și ?> care ne permite să intrăm sau ieșim dim modul PHP.

Diferența cea mai semnificativă intre limbajele de client side ca și JavaScript este că codul PHP se execută pe server, generând HTML care va fi trimis la client. Partea client va primi rezultatul scriptului rulat, dar nu cunoaște codul susținută. Serverele pot fi configurate ca să se proceseze toate HTML-urile utilizând PHP, în acest fel utilizatorii nu va știe nimic despre codul pe server.

Interpretorul standard de PHP este bazat pe Zend Engine, care este un software free elaborat sub licența de PHP. PHP-ul poate fi larg utilizat în cele mai multe servere web și aproape pe toate sistemele de operare.

Dezvoltarea limbajuli a inceput in anul 1995, cand Rasmus Lerdorf a folosit limbajul pentru proiecte personale. Dupa testarea puplica lansaea oficială al PHP 3 a avut loc ăn Junie 1998. În data de 22 Mai 2000 a fost lansat PHP 4 cu motorul Zend 1.0. Dezvoltarea PHP 4 s-a terminat la August anul 2008, limbajul a ajuns pâna la versiunea 4.4.9. În date de 13 Julie 2004 a fost puplicat PHP 5 care a adus nou proprietății ca și suport-ul de programare orientată de obiecte, obiectele de date in PHP (PDO) și numeroase înbunătățiri de performanță. Versiunea 6 al limbajului PHP nu a fost lansat din cauza lipsei suportului de Unicode nativ. Versiunea următoare, PHP 7 a fost lansat in 2015 care a fost o înbunătățire al limbajului prin refactorizarea motorului Zend.

Limbajul PHP a fost folosit pentru cele mai utilizate framework-uri ca și PRADO, CakePHP, Symfony, CodeIgniter, Lavarel, Yii Framework, Phalcon și Zend Framework. In Octombrie al anului 2010 limbajul PHP a fost limbajul cel mai utilizat în programe de server side 75% în procente. Cele mai mari Web content management sisteme scrise in PHP încud MediaWiki, Joomla, WordPress, Drupal, Moodle, partea de utilizator al Facebook-ului.

*Argument technologic.*Am ales acest limbaj pentru dezvoltare aplicației de server, pentru că am experiență în acească technologie și PHP-ul este limbajul care a fost creat pentru dezvoltarea aplicațiilor web.

4.5.9. Limbajul Python

Python [20] este un limbaj de programare de nivel înalt folosit pentru scopuri generale creat de către Guido van Rossum și prima dată publicat în anul 1991. Este un limbaj independent, Python urmăreste filozofia de design accentuat la lizibilitatea codului și o sintaxa care ajuta programatorii ca să exprime conceptele în cele mai puține linii de cod posibile într-un limbaj ca și C sau Java.

Limbajul Python are un sistem de tipuri de date dinamic și managemant de memorie automată, limbajul suporta paradigme de programare multiple înclus programarea orientete de obiecte, programare funcțională, imperativă și procedurală. Are o librarie standard mare si cuprinzătoare.

Limbajul Python este larg utilizat și are interpretoare pentru orice sistem de operare, astfel permite codului Python sa fie executabil în foarte multe variații de sisteme. CPython este o implementare de referința al limbajului Python, este un program open source și are un model de dezvoltare bazat pe comunitate. CPython este gestionat de către organizația non profit Python Software Foundation.

Limbajul Python fiind un limbaj de nivel înalt se citește similar ca și limba engleză, din această cauză este foarte greu de învățat sintaxa limbajului. Python este foarte flexibil fiind un limbaj scris dinamic. Asta înseamnă că nu este foarte regulat la

crearea aplicațiilor, și programatorul are mai multe flexibilitate la rezolvarea problemelor folosind metode diferite.

Din cauza că Python este un limbaj scris dinamic, aceeași bucată de cod poate fi translatat la ceva diferit în funcție de context foarte ușor. Limbajul Python este lent din cauza că este flexibil și calculatorul trebuie să facă o grămadă de referențiere la orice definiție, acest lucru scade performanța limbajului Python. [19]

Începând de anul 2003, Python este între primele zece cele mai populare limbaje de programare, de la august 2016 este al cincielea pa aceași listă. Organizații mari care folosesc Python include Wikipedia, Google, Yahoo!, CERN, NASA. Python pot fi folosit ca limbaj de scripting pentru crearea aplicațiilor web, sistemele de cadru care folosesc Python sunt Django, Pylons, Pyramid, TurboGears, Tornado și multe alte. Librăriile ca NumPy, SciPy și Matplotlib permite folosirea limbajului în calculi științifice.

Argument technologic. Am ales acest limbaj pentru că librăriile pentru descărcarea datelor de pe serverele ECMWF și librăria pentru citirea fișierelor .drib sunt scrise in Python.

4.5.10. Cron

Cron-ul [21] este un program rulat in spate pe un sistem de operare folosit pentru executarea tascurile dorite în timpurile specificate.

Fișierul crontab este un fițier text simplu care conține lista comenzilor configurate să se ruleze în timpurile specificate. Fișierul poate fi editat folosind comanda crontab. Comenzile configurate în filierul de crontab sunt verificate și executate de programul rulat în spate. Fiecare utilizator al sistemului de operare are un fișier crontab separat. Fiecare fișier este rulat în spate indiferent dacă utilizatorul respectiv este autentificat sau nu.

Pe sistemele Ubuntu bazate pe Gnome este disponibil o interfață grafică pentru monitorizarea cron-urilor. Interfața grafică poate fi instalată cu comanda:

sudo apt-get install gnome-schedule.

Pentru utilizarea cron-urilor trebuie adăugate rânduri în fișierul de crontab. Pentru editarea fișierului folosim comanda: crontab -e. Fiecare rând din fișierul crontab are cinci câmpuri de timp sau dată, după care se scrie comanda pe care execută cron-ul și se încheie cu un caracter de linie nouă. Câmpurile sunt separate cu spații, cele cinci câmpuri nu pot conține spații. Cele cinci câmpuri care descrie timpurile de rulare ale comenzii sunt următoarele, minute: ia valorile între 0 și 59, ore: valori între 0 și 23, zile: ia valori între 1 și 31, luni: ia valori între 1 și 12, ziua: valori între 0 și 6 unde 0 reprezintă ziua de duminică.

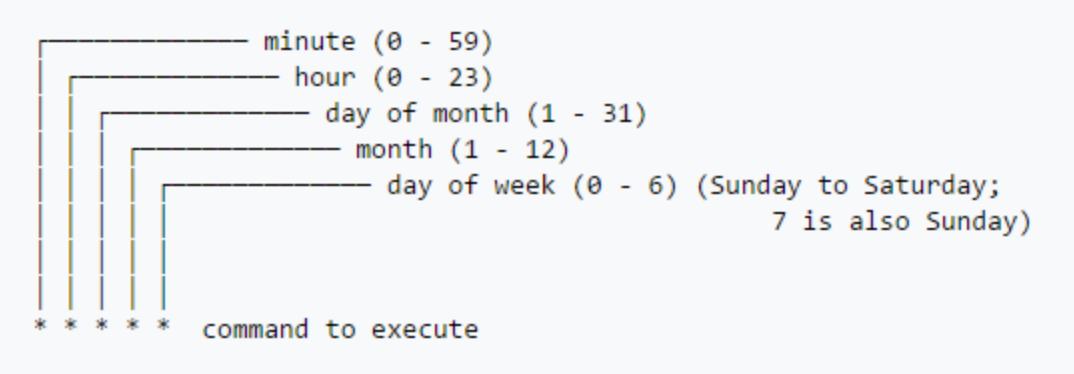

Figura 4.7 Explicarea crontabului [7]

Caracterul ”*” poate fi folosit ca o instanță care ea toate valorile posibile, mai multe valorii al unui câmp pot fi separate cu virgulă, valorii separate cu caracter-ul ”/” se folosește pentru rulare continuu, ca exemplu folosim ”/” dacă comanda trebuie rulat în fiecare trei ore. Niște implementări cron suportă următoarele prescurtări:

- @yearly - se rulează odată pe an la miezul nopții la 1 januarie
- @monthly - se rulează odată pe lună la miezul nopții, prima ziuă din luna respectivă
- @weekly - odată pe săptămâna la miezul noptii duminică
- @daily - odată pe zi la miezul nopții
- @hourly - in fiecare oră la primul minut
- @reboot - se rulează la pornirea sistemului [11]

Argument technologic. Cron-ul sistemului Linux este alegerea relevană pentru că nu există o altă aplicație care este încorporat în sistem.

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol este prezentat modul de proiectare al sistemului atât partea de afișarea datelor meteorologice cât și partea care apdatează datele meteorologice, cu ajutorul diagramelor de deployment, structura bazei de date, diagrama fluxului de execuție care ne arată funcționarea aplicației în ansamblu.

5.1. Structura generală al sistemului

Aplicația de monitorizare al vremii se bazează la arhitectura client-server, în care comunicarea între aplicația de client și aplicația de server se bazează pe protocolul HTTP.

Sistemul este alcătuit din două aplicații, una reprezintă datele meteorologice pentru utilizatori, și celălalt descarcă datele meteorologice de pe un set de date publice. Legătura între cele două aplicații este baza de date, ambele părți ale sistemului accesează aceasi bază de date locală.

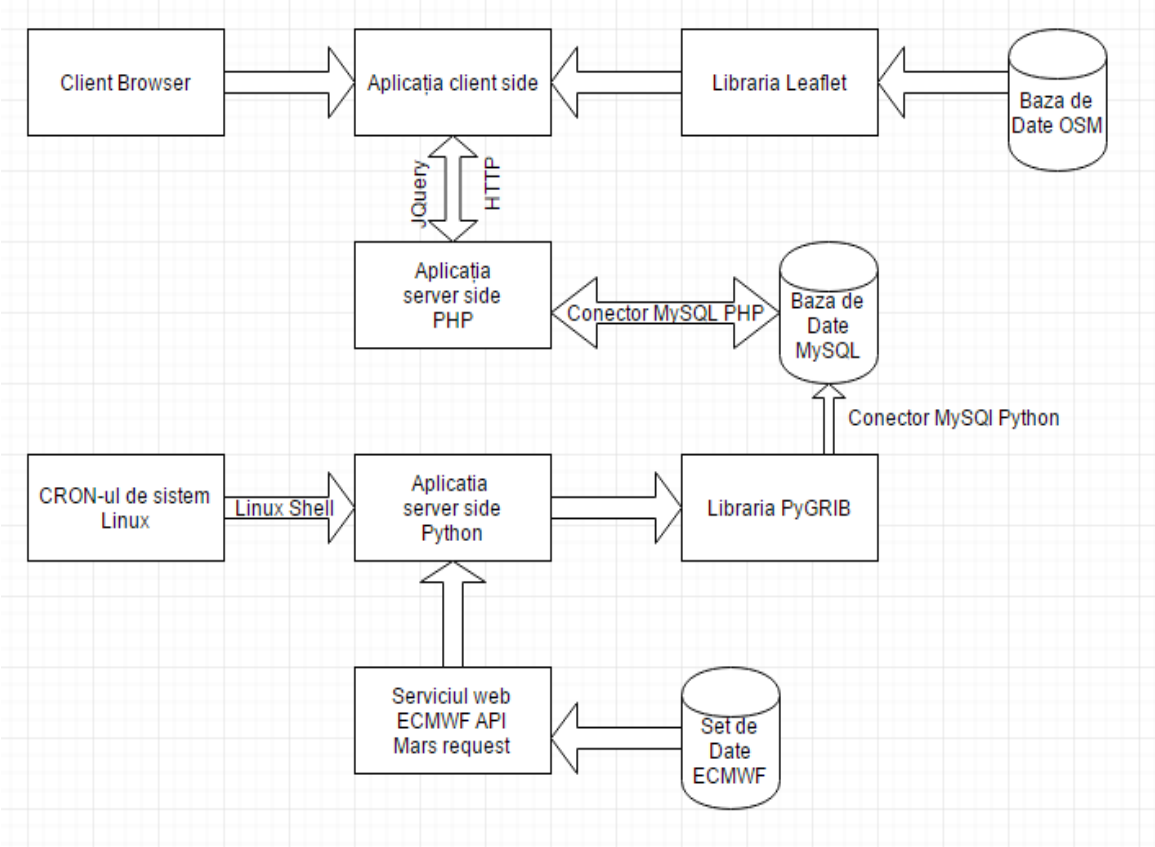

Figura 5.1 Schema generală a sistemului

În figura de mai sus este prezentată architectura de nivel înalt al aplicației. Sistemul este construit din trei module mari: aplicația de frontend, aplicația backend și aplicația de actualizare al datelor meteorologice. Fiecare modul este construit folosind tehnologii diferite, aplicația client-side co-operează cu aplicația de server-side iar aplicația de actualizare a datelor meteorologice se rulează automat și separat, se conectează la sistem prin baza de date locală care conține datele meteorologice.

Aplicația de client-side folosește technologiile HTML, JavaScript și Css. Librăria leaflet și OpenStreetMap-ul tot face parte de aplicația de client-side. Librăria Leaflet este mediatorul între aplicația client-side și OSM.

Aplicația de server-side este scrisă în limbajul de programae PHP, și este modulul care preia cererile aplicației de client-side și furnizează rezultatele. Acest modul se conectează la baza de dat MySQL unde sunt stocate informațiile meteorologice și datele utilizatorilor.

Aplicația de actualizare al datelor meteorologice este scrisă în limbajul de programare Python și este responsabil pentru actualitatea datelor meteorologice în baza de date. Folosește aceași bază de date ca și aplicația de backend.

Diagrama de pe figura 5.2 și 5.3 reprezintă fluxul de execuție al sistemului. Pe această diagramă se vede că aplicația de actualizare al datelor meteorologice se execută separat, și executia programului nu se concură cu execuția aplicaților celalaltă două. Din diagramele de flux de execuție putem înțelege logica de funcționare a sistemului. Fluxul de execuție ne arată componentele aplicaților când și cum sunt utilizate în cursul execuției al acțiunilor aplicațiilor.

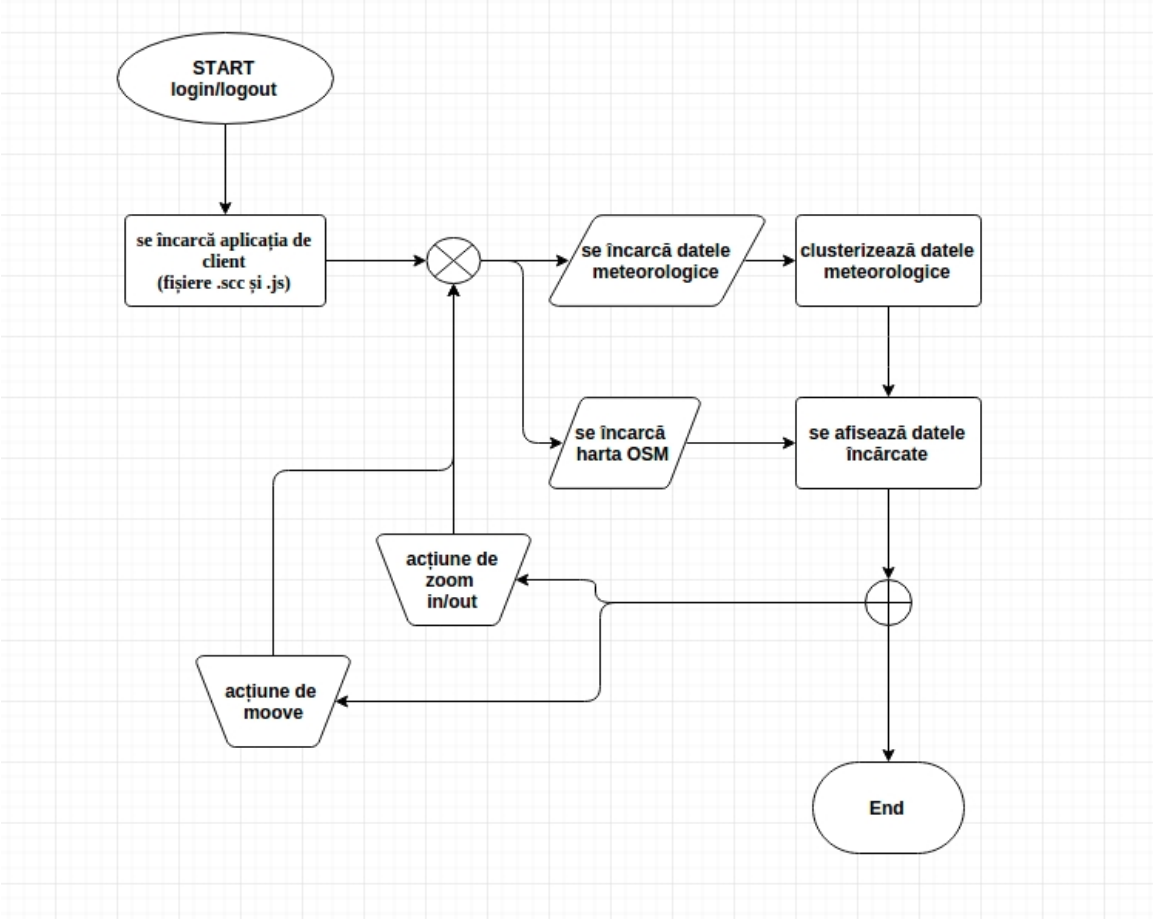

Figura 5.2 Flux de execuție al aplicației de reprezentare al datelor meteorologice.

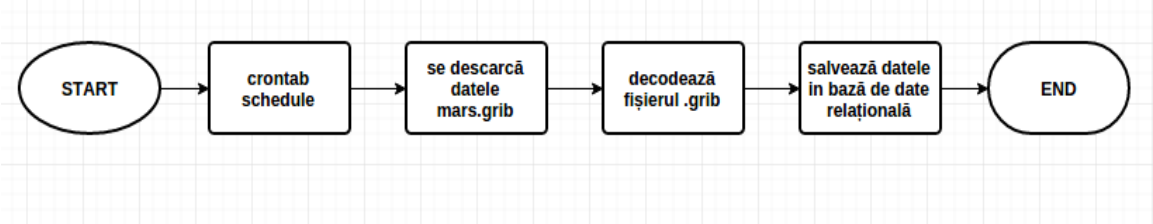

Figura 5.3 Flux de execuție al aplicației de actualizare al datelor meteorologice.

Fluxul de execuție al aplicației de actualizare al datelor meteorologice este mult mai simplu și linear, părțile aplicației sunt executate unul după altul, utilizatorul nu poate interacționa în execuția programului.

5.2. Diagrama de deployment

Diagrama de deployment este reprezentarea componentelor fizice ale sistemului. În diagrama din Finura 5.4 sunt reprezentate componentele de hardware al sistemului alocate în resursele utilizatorului sau proprietarul de sistem iar și componentele OSM și ECMWF.

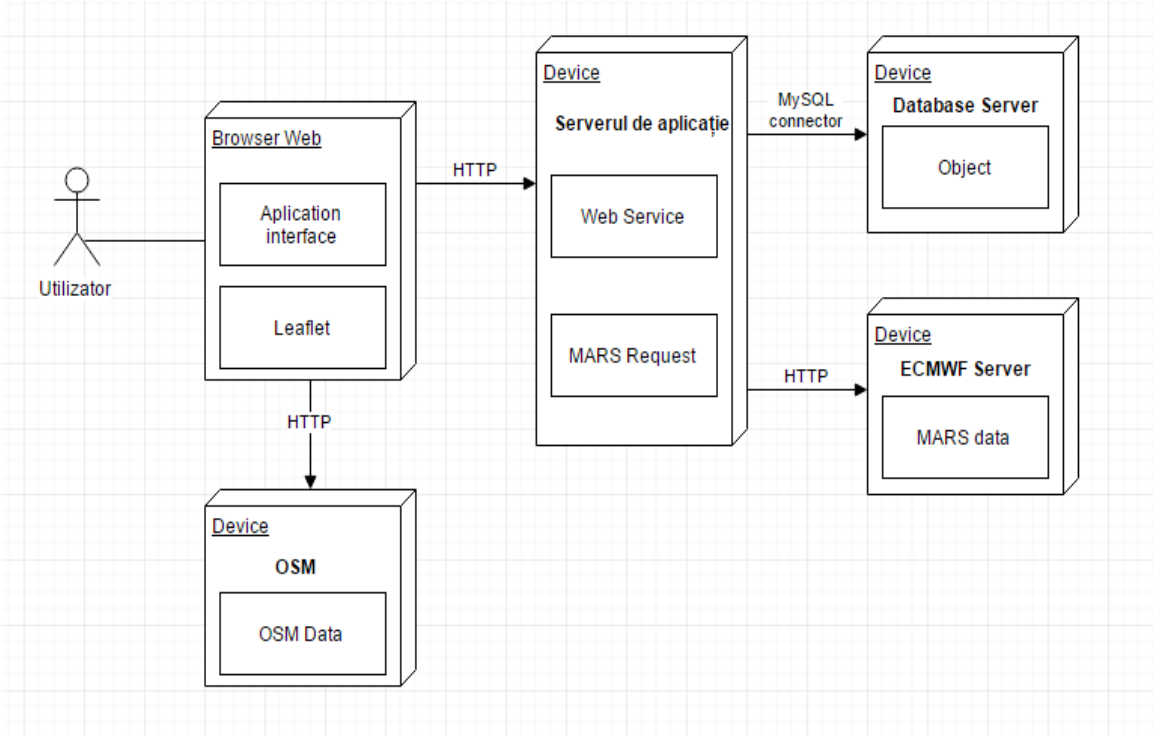

Figura 5.4 Diagrama de deployment

Componenta de Browser Web este calculatorul utilizatorului pe care, după acesarea site-ului, va fi descărcată aplicația de client-side și va fi rulat în browser. Browserull web comunică prin protocolul HTTP cu server-ul. Calculatorul de server conține aplicația de server-side și aplicația de actualizare a datelor meteorologice. Aceste programe sunt executate pe calculatorul de server și folosesc resursele acestora. Resursele server-ului nu sunt acesibile de către utilizator. Serverul de bază de date conține baza de

date și instalarea MySQL. Componentele OSM și ECMWF sunt componentele indirecte al sistemului, proprietarul nu are acces și poate gestiona aceste servere. Componentele de OSM și ECMWF furnizează datele la cererile aplicației.

5.3. Structura bazei de date

Sistemul folosește o bază de date construită pe platforma de MySQL. În diagrama următoare sunt prezentate tabelele din baza de date.

Baza de date este simplă și tabelele sunt destul de articulate. Aplicația lucrează cu un volum mare de date, cu puține conexiuni, relații între ele. Tabelele bazei de date pot fi grupate în tabele care rețin datele meteorologice, și tabelele care conțin datele utilizatorilor înregistrate.

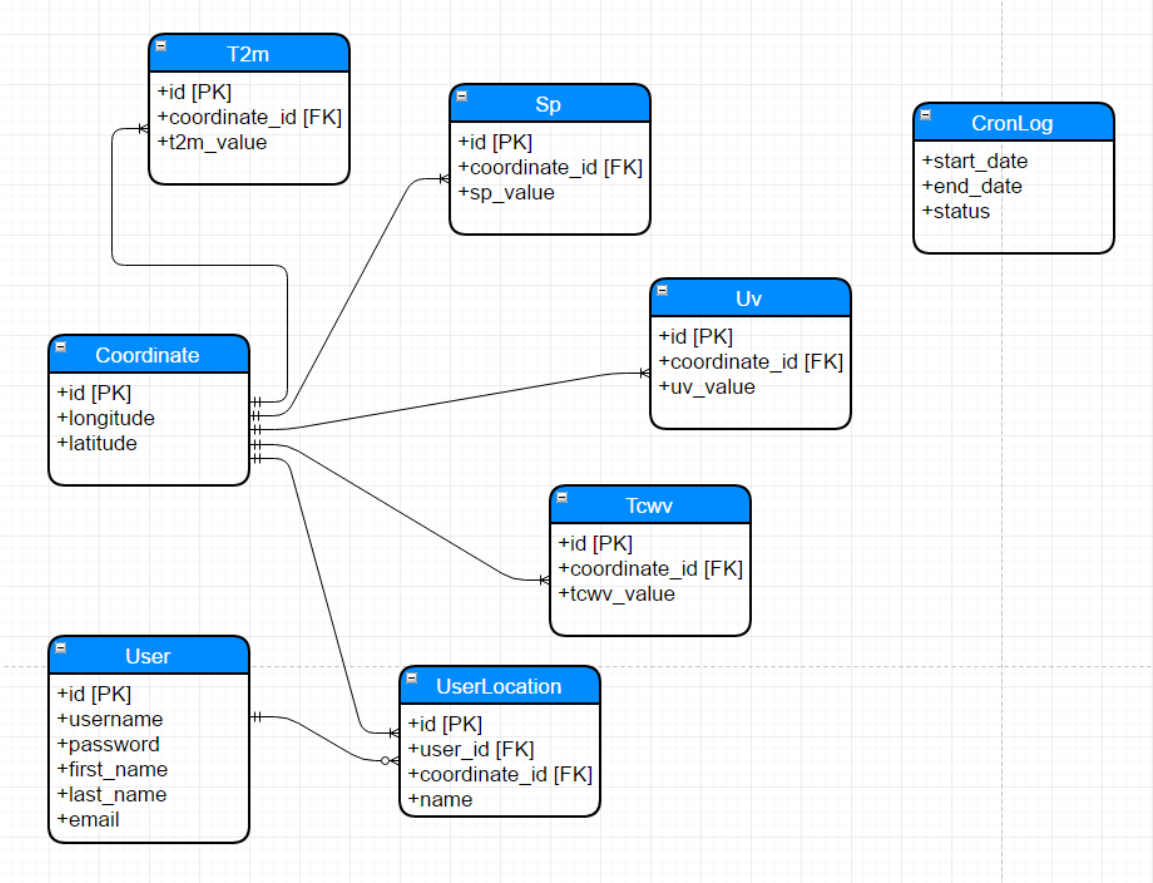

Figura 5.5 Diagrama bazei de date

Tabela **T2m** este un tabel care reține o proprietate meteorologică. Tabelul conține temperatura la înălțimea de doi metrii. Temperatura este legată la coordonate, deci tabelul conține informațiile de latitudine și longitudine.

Tabela **Sp** este un tabel care reține o proprietate meteorologică. Tabelul conține informația de presiune atmosferică. Structura tabelului e similar la toate tabelele care conțin date meteorologice.

Tabela **Uv** conține valoarea radiației al razelor uv. Acest tabel este o tabelă care reține date meteorologice.

Tabela **Tcwv** conține valoarea proprietății meteorologică de precipitație. Denumire tabelelor care conțin date meteorologice sunt alease conform denumirea acestor proprietății în sistemul ECMWF.

Tabela **Coordinate** conține perechile de longitudine și latitudine, acest tabel rezolvă repetarea datelor stocate în baza de date.

Tabela **User** conține datele utilizatorilor înregistrate în sistem. Tabela conține informațiile de nume, prenume, nume de utilizator, parola criptată și adresa de email al utilizatorilor. Cu ajutorul acestui tabel este realizată înregistrarea și autentificarea utilizatorilor.

Tabela **UserLocation** conține locațiile căutate de către utilizatorii înregistrați în sistem. Tabela conține coordonatele locației, identificatorul utilizatorului și numele locației căutate. Cu ajutorul acestui tabel utilizatorul poate vizualiza locațiile căutate după fiecare autentificare.

Tabelul **CronLog** conține datele rulării al aplicației de actualizare al datelor meteorologice.

5.4. Module principale al sistemului

5.4.1. Aplicația client-side

Aplicația client-side este dezvoltat în limbajul JavaScript folosind elemente de HTML și Css. Componentele aplicației sunt incluse în aplicație din fisierul index.html, acest fișier este centrul aplicației la care se conectează componentele aplicației, componentele comunică înre ele de asemenea. Aplicația de frontend are 5 mari module care se conectează la centrul aplicației, modului de map, componenta de cluster, componenta search, database și user.

Modului de Map se realizează conectarea la libraria Leaflet, și gestionează obiectul de map, creat de către Leaflet. Acest modul gestionează controlerul de layere pe hartă, se pune lista markerelor sau clusterelor pe hartă. În acest modul sunt ascultate acțiunile de zoom și moove pe hartă și în acest modul sunt create fereastrele de popup care conțin informații detaliate despre un cluster sau marker ales pe hartă. Punctele ce cluster sunt stocate in obiecte LayerGroup car apartine la librăria Leaflet.

Componenta de Cluster este responsabil pentru clusterizarea datelor meteorologice. În acesta componentă este implementat algoritmul K-Means de clusterizare, folosit pentru gruparea markerelor pe hartă. Fișierul este alcătuit din trei funcții, unul este funcția principala clusterKMeans(), în care sunt implementate pașile algoritmului. În afară de funcția principală mai sunt doua funcții auxiliare getNearestCentroid() care returnează clusterul cel mai apropiat la un marker, și funcția isMarkerInContainer() care verifică dacă un marker este vizibil în fereastra de browser.

Componenta de Search conține încorporarea funcționalității de căutare pe hartă care se bazează pe un modul de Google. Modulul de la Google Search returnează suggestiile de căutare și coordonatele locației căutată.

Modulul de Database interoghează datele din baza de date locală. Componenta acesta realizează comunicarea cu applicația de pe server prin apeluri Ajax. Acest modul este responsabil pentru încărcarea datelor meteorologice în aplicația de client-side. Aplicația încarcă datele odată la accesarea siteului, și reține datele în browser. Încărcarea

datelor mai multe ori este mai costisitoare decât reținerea acestora, la fel clusterizarea în browser e mai rapidă decât clusterizarea în backend.

Componenta de User este responsabil pentru gestionarea utilizatorilor, acest modul realizează posibilitatea înregistrării și autentificarea utilizatorilor siteului. Acest modul folosește Ajax ca să înterogheze datele utilizatorilor la fel ca și în modulul Database.

Aplicația client-side apelează scripturile din aplicația server-side prin apeluri Ajax, răzpunsul primit de la aplicația server side este prelucrat în partea de client.

5.4.2. Aplicația server-side

Aplicația de server-side este responsabil pentru furnizarea datelor meteorologice către aplicația de client-side și gestionarea datelor utilizatorilor. Aplicația este scrisă în limbajul de programare PHP și este implementată conform paradigmelor programării orientată pe obiecte.

Aplicația server-side este construită folosind patternul arhitectural Layers. Arhitectura Layers este cel mai popular pattern arhitectural folosit în proiectare software și esența este că clasele aplicației sunt grupate în layere, fiecare layer are o denumire și mai multe roluri, funcționalității specifice. O clasa poate aparține numai la un singur layer arhitectural, și trebuie sa respecte regulile de design și implementare al layerului respectiv. Două clase din aplicație pot comunica cu o alta clasă dacă clasa cealaltă aparține unui layer vecin al layerului clasei respective [1].

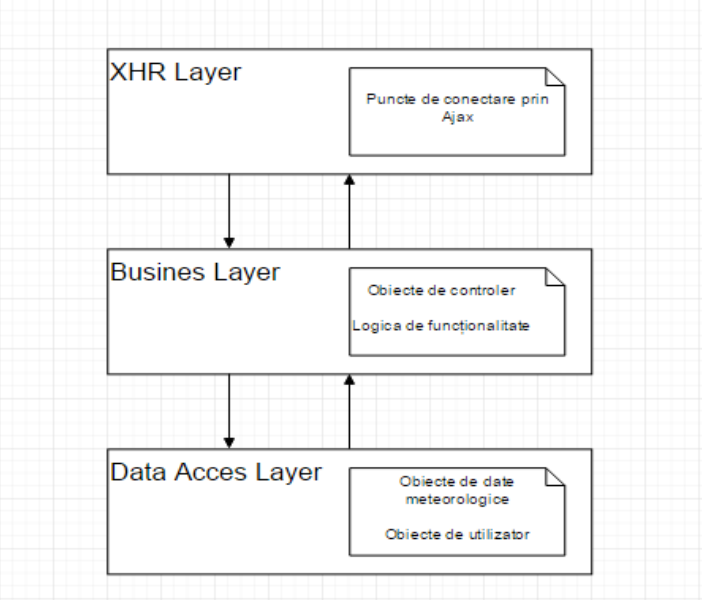

Figura 5.6 Diagrama layers

În aplicația server-side putem separa trei layere conform designului arhitecturii Layers. Layerul de acces date cu denumirea DataAccesLayer, acest layer conține clasele care reprezinta datele din baza de date și se încarca datele aflate în baza de date. Acest layer este defapt conectorul la baza de date, îl transforma datele relaționale din baza de date în obiecte folosite în aplicație ca și datele meteorologice și utilizatori înregistrați în sistem.

Layerul care descrie logica de implementare și al funcționalităților aplicației se numește BusinessLayer. Acest layer conține clase de tip controler, care controlează obiectele create în layerul de acces date. Clasele din acest layer se execută cererile aplicației de client-side și rezultatul lor este răspunsul care va fi trimis înapoi la aplicația client-side. Aceste clase gestionează obiectele care reprezină datele meteorologice și obiectele care reprezintă datele utilizatorilor.

Layerul care realizează conecțiunea la aplicația de client-side se numeste XHRLayer.

Acest layer conține fisierele php care sunt accesate de către apelurilor Ajax. Pentru fiecare apel Ajax din proiect aparține un fișier din layerul XHR. Fișierele din acest layer urmăresc următoarele pași. În primul rând se definesc include path-ul pentru aplicația de server-side, după asta urmează instanțierea controlerului necesar pentru executarea apelului și la final printează rezultatul primit la controlerul respectiv, acest text printat va fi trimis la aplicația de client-side și va fi răspunsul la apelul Ajax din frontend.

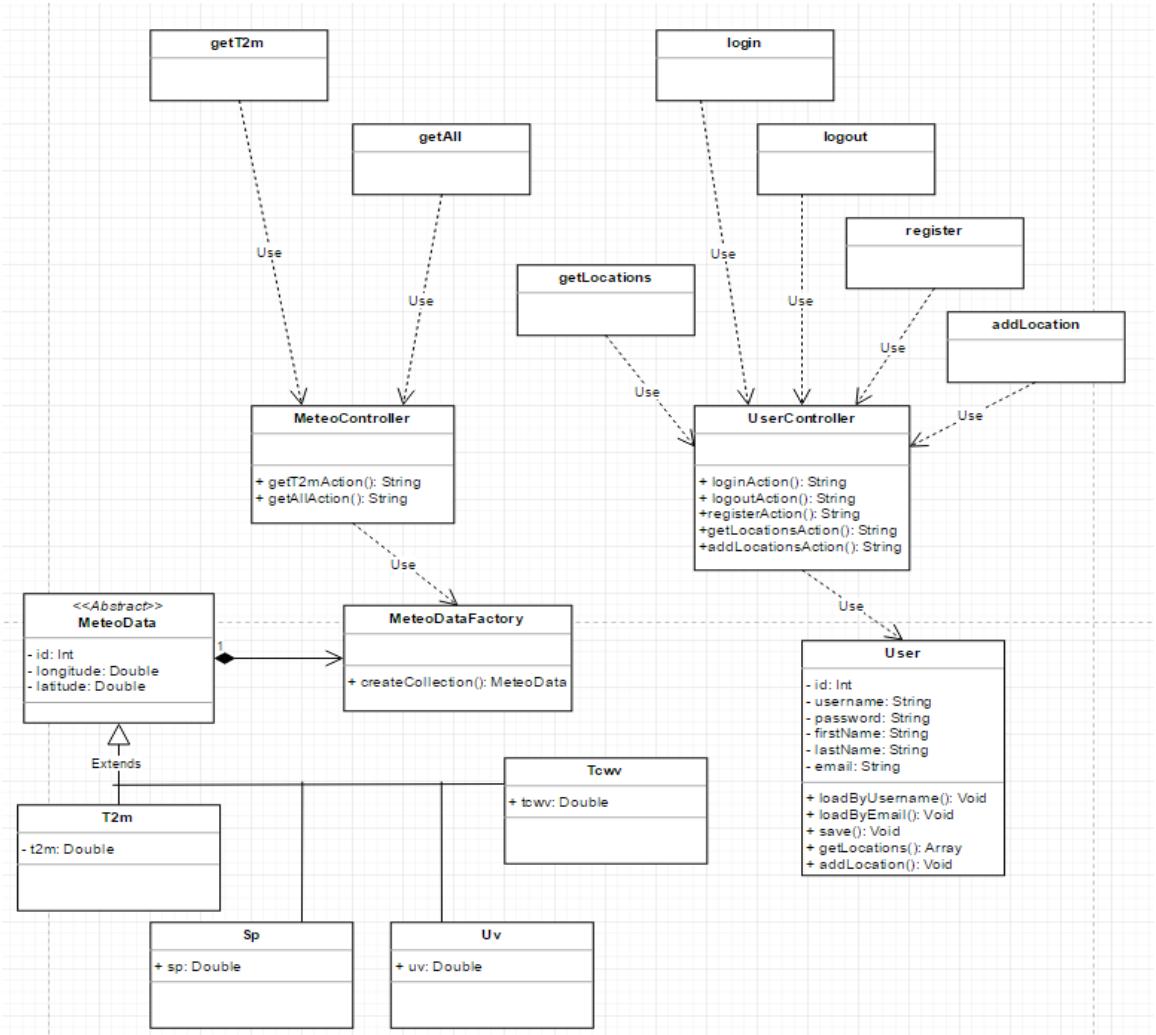

Figura 5.7 Diagrama claselor aplicației de server

5.4.2.1. Layerul XHR

Layerul XHR este layerul care conține fișierele de php care realizează comunicarea aplicației server-side cu aplicația de client-side.

Fiecare script PHP din acest layer reprezintă un apel Ajax din aplicația client-side. Acțiunea se începe în aplicația de client-side, o înterogare Ajax este declanșat și unul dintre fișierele din layerul XHR sunt executate prima dată pe server. Aceste scripturi au o schemă unitară și nu conțin nici o logică în cod, executarea scripturilor este lineară.

Regulile layerului de XHR sunt următoarele. Acest layer conține scripturi scrise in limbajul php, scriptul nu conține clase, acest script este apelat direct de către apelurile de Ajax lansate în aplicația de client-side. Acest script apelează o metoda a unei clase din layerul de business și printează rezultatul primit de la controller. Text-ul printat va fi răspunsul trimis la aplicația de client-side.

Scriptul **getT2m.php** este accesat de către aplicația client-side după fiecare încărcare al siteului web în browser. Scriptul incarcă controllerul MeteoController, după instanțierea controlerului apelează metoda getT2mAction. Răspunsul primit de la metodă este in formatul JSON pe care se printează. Răspunsul printat este trimis la aplicația client-side. Scriptul returnează informația meteorologică de temperatură, această înterogare din aplicația de client este cerută după încărcarea aplicației în browser și sessiunea de user nu sete setată, sau nu există.

Scriptul **getAll.php** este accesat de către aplicația client-side după fiecare încărcare al aplicației în browser dacă utilizatorul este autentificat. Scriptul încarcă controlerul MeteoController, după instanțierea controlerului, scriptul apelează metoda getAllAction. Răspunsul primit de la metoda getAllAction este în formatul JSON, acest răspuns este printat de către script și apoi trimis la aplicația client-side. Acest script returnează fiecare patru informație meteorologică folosită în sistem. Prin scriptul getAll din layer-ul XHR sunt trimise la aplicația de client datele de temperatură, de presiune atmosferică, radiația uv și precipitația.

Scriptul **login.php** este accesat când un utilizator al aplicației apasă butonul de login în aplicația de client. Scriptul încarcă controlerul UserController, după instanțierea controlerului, scriptul apelează metoda loginAction din controler. Răspunsul primit de al controler este un mesaj de acknowledge, care poate fi textul ”OK”, sau un mesaj de eroare. Mesajul este trimis la aplicația client-side ca răspuns la apelul Ajax. Prin executarea scriptului login.php utilizatorul devine autentificat în sistem, astfel are acces la toate funcționalitățiile aplicației.

Scriptul **logout.php** este accesat când un utilizator autentificat în sistem apasă butonul de logout în aplicația de client. Scriptul încarcă controlerul UserController, după instanțierea controlerului scriptul apelează metoda logoutAction din controler. Răspunsul primit de la controler este un mesaj de eroare sau un mesaj de acknowledge care recunoaște delogarea utilizatorului.

Scriptul **register.php** este accesat când un utilizator al sistemului încearcă să se înregisreze în sistem. Scriptul încarcă controlerul UserController, după instanțierea controlerului, scriptul apelează metoda de registerAction din controler. După executarea metodei răspunsul primit de la controler este tipărit astfel trimis la aplicația client-side.

Scriptul **addLocation.php** este accesat cand utilizatorul autentificat caută o locație cu ajutorul modulului de căutare. Scriptul instanțiază controllerul UserController,

și apelează metoda addLocationAction din controller. După executarea metodei răspunsul primit este tipărit de către script și trimis la aplicația client-side.

Scriptul **getLocations.php** este accesat la încărcarea aplicației în browser după încărcarea datelor meteorologice, și după fiecare adaugare de locație. Scriptul instanțiează controlerul UserController, și apelează metoda getLocationsAction din controller. După executarea metodei răspunsul primit in format Json este printat și trimis la aplicația de client.

5.4.2.2. Business Layer

În layerul de business se află clasele de tip controller, aceste clase conțin logica funcționalităților sistemului. Acest layer este situat între layerele de xhr și de acces de date. Layer-ul business răspunde la apelurile primite de la layer-ul xhr și foloseste clasele din layer-ul acces de date pentru încărcarea resurselor.

Layerul de business conține două clase de controler, controlerul pentru gestionarea utilizatorilor UserController și controllerul pentru gestionarea datelor meteorologice MeteoController, fiecare metodă din aceste clase sunt apelate de către un script specific din layer-ul de xhr, deci pentru fiecare script din layer-ul xhr este asociat o metodă din clasele de controler.

Regulile layerului de business sunt următoarele. Layer-ul business conține metode de tip acțiune, aici se află codul care operează, se schimbă sau transformă datele. Clasele din acest layer sunt conectate numai cu aplicația server side, nu fac conexiuni externe cu baza de date sau cu aplicația de client. Metodele sunt grupate în clase, clasele sunt de tip controler. Controlerul UserController conține acțiuni legate cu utilizatorii, si folosește resursele de tip utilizator iar colntrolerul MeteoController conține acțiuni legate cu datele meteorologice, și folosește resursele meteorologice.

Controlerul **MeteoController** este controlerul care gestionează datele meteorologice. Pornind de la funcționalitățiile sistemului, avem două cazuri ori avem un utilizator neautentificat care poate vizualiza informațiile de temperatură ori avem un utilizator autentificat care are acces la toate informațiile meteorologice. Clontrolerul MeteoController conține două metode pentru acoperirea aceste două cazuri.

- Metoda **getT2mAcion** transmite colecția punctelor cu informația meteorologică de temperatură folosind obiecte din layer-ul de acces date, către scriptul din layer-ul xhr cara a apelat metoda. Metoda getT2mAction transformă colecția de obiecte în format Json.
- Metoda **getAllAction** transmite colecția punctelor cu toate informațiile meteorologice folosind obiecte din layer-ul de acces date, către scriptul care a apelet metoda. Metoda încarcă patru colecții de obiecte pentru fiecare patru tip de date meteorologice. Colecțiile cu obiecte meteorologice sunt mergeuite și transformate în format Json.

Controlerul **UserController** este controlerul care gestionează datele utilizatoriilor. Acest controler conține metodele care sunt asociate cu scripturile din layer-ul xhr și două metoda care sunt componente private în clasă și ajută la validarea inputurilor utilizatoriilor. Controlerul UserController conține următoarele metode.

- Metoda **loginAction** autentifică utilizatorul în sistem. Înainte de setarea variabilelor de sessiune care este baza autentificării, metoda trece prin câteva pași.

Metoda validează datele de input de la utilizator din motive de securitate și verifică dacâ utilizatorul a completat campurile obligatorii în formularul de autentificare. Metoda verifică dacă datele întroduse de către utilizator corespund cu datele din baza de date, pentru acest lucru controlerul folosește obiecte de tip utilizator din layerul de acces date. Dacă informațiile primite de la utilizator nu trec la un punct de verificare, metoda returnează o serie de mesaje de eroare, în caz contrar metoda setează datele utilizatorului, id-ul și numele de utilizator, în variabila de sessiune, și returnează mesajul de succes.

- Metoda **validateLoginData** este o metoda ajutătoară, și este o componentă privată al clasei. Metoda trece pe un pas de securizare al datelor care ajută la prevenirea atacurilor de tip cross site scripting, apoi sunt verificate dacă câmpurile obligatorii sunt completate. Dacă metoda parcurge toți pași cu succes datele validate sunt returnare, în caz contrar returnează valoarea false.
- Metoda **logoutAction** ajută utilizatorul sa închide autentificarea în sistem. Metoda este foarte simplă, distruge variabilele de sesiune și returnează mesajul de succes.
- Metoda **registerAction** ajută la înregistrarea utilizatorilor în sistem. Metoda răspunde la apelul din leyer-ul xhr. Ca și la autentificare, metoda registerAction înainte de a salva utilizatorul în baza de date parcurge o serie de verificări. Metoda validează datele de input de la utilizator din motive de securitate și verifică dacâ utilizatorul a completat câmpurile obligatorii în formularul de înregistrare. La următorul pas metoda verifică dacă deja există în baza de date un utilizator care are același nume de utilizator sau aceeași adresă de email. Dacă informațiile utilizatorului trec la fiecare punct de validare, metoda crează un obiect nou de tip utilizator și asignează datele primite la obiect și apelează metoda de salvare pe obiect care va salva utilizatorul în baza de date. În caz contrar metoda returnează o serie de mesaje de eroare.
- Metoda **validateRegisterAction** este o metodă ajutătoare, este o componentă privată a clasei. Metoda trece pe un pas de securizare al datelor întroduse de către utilizator, care ajută la prevenirea atacurilor de tip cross site scripting. La următorii pași se verifică dacă câmpurile obligatorii sunt completate, dacă numele si prenumele utilizatoului nu conține alte caractere decât litere și dacă adresa de mail e validă. Dacă metoda parcurge toți pașii cu succes returnează datele validate, în caz contrar returnează valoarea false.
- Metoda **getLocationsAction** ajută la trimiterea locațiilor salvate al utilizatorului logat la aplicația de client. Metoda este apelată de către un script din layer-ul de xhr. În primul pas metoda încarcă userul conform variabilelor de sessiune, în următorul pas metoda încarcă locațiile utilizatorului și returnează locațiile în format Json.
- Metoda **addLocationAction** ajută la salvarea sau adaugarea unei locații la utilizatorul logat. Metoda este apelată de către un script din layer-ul xhr. Metoda transformă datele primite într-un array care reprezintă locația. În următorul pas metoda încarcă utiltizatorul conform variabilelor de sessiune, si adaugă locația la utilizator.

5.4.2.3. Data Acces Layer

Layerul de acces date conține clasele de tip model, aceste clase se modelează și reprezintă datele din baza de date. Acest layer realizează conexiunea între aplicația server-side și baza de date, clasele din aceste layer sunt folosite de către clasele din layer-ul de business.

Rolurile layerului de acces date sunt următoarele. Layerul da acces date contine modele de date, și se conectează la baza de date, folosește înterogării SQL. Clasele din acest layer se transformă datele relaționale din baza de date în obiecte. În aceste clase nu se află implementări al funcționalităților sistemului.

În acest layer se află clasele de model cere reprezintă obiectual datele meteorologice și utilizatorii. În afară de astea layerul mai conține o clasă care reprezintă conexiunea la bază de date. Clasele care reprezintă datele meteorologice, au perechi de colecții, o clasă de colecție operează cu mai multe obiecte de aceași tip.

Clasa **Connection** reprezintă conecțiune la baza de date. Clasa implemantează desing pattern-ul Singleton. Această clasă poate fi instanțiat numai odată la rularea aplicației. La fiecare apelare al clasei returnează același obiect creat astfel nu încarcă memoria serverului de fiecare dată când este folosit. Am folosit patternul singleton la această clasă pentru că este destul de general și des utilizat în sistem.[1]

Pentru reprezentarea datelor meteorologice sistemul foloseste o clasă abstractă care este o clasă generală pentru reprezentarea datelor meteorologice. Câte patru clase pentru cele patru tipuri de date meteorologice care extind clasa abstractă, pentru fiecare patru clasă meteorologică există o clasă de collecție care conține mai multe instanțe de date meteorologice intr-un singur obiect. Pentru clasele de colecții de date meteorologice este implementat design pattern-ul factory, care usurează folosirea acestor clase.

Clasa **MeteoData** este clasa abstractă pentru date meteorologice și conține proprietățile și metodele comune ale claselor care reprezintă datele meteorologice. Cele patru clase pentru reprezentarea datelor meteorologice sunt mostenite din această clasă.

Clasa **MeteoDataCollectionFactory** este clasa care crează obiectele, in clasa client numai un obiect de factory este instanțiat, și nu fiecare colecție de date meteorologice. Acestă clasă conține o metodă **createCollection** care returnează obiecte de tip colecție conform unui parametru care indică felul obiectului returnat. În cazul nostru metoda crează obiectele conform denumirea informației meteorologice care poate lua valorile: ”T2m”, ”Sp”, ”Uv” sau ”Tcwv”, dacă valoarea parametrului e altceva decât cele specificate, metoda returnează valoarea null.

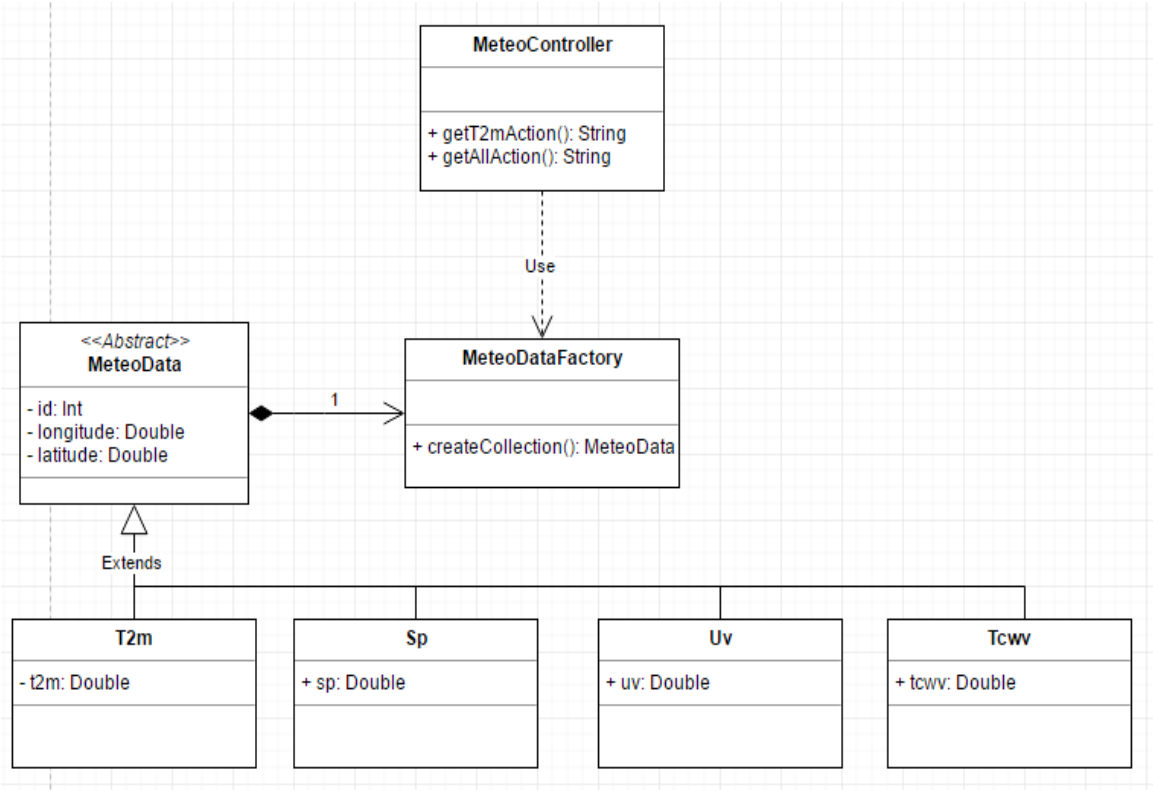

Figura 5.8 Diagrama pattern-ului factory.

Clasele **T2m**, **Sp**, **Uv** și **Tcwv** reprezintă datele meteorologice stocate în baza de date. Fiecare cele patru clase extind clasa abstractă MeteoData, și adaăgă proprietatea respectivă în plus. Acest clase pot stoca instanțele meteorologice, care sunt folosite în sistem.

Clases **T2mCollection**, **SpCollection**, **UvCollection** și **TcwvCollection** sunt perechile de colecție al claselor de mai sus. Aceste clase operează cu mai multe obiecte astfel simplifică folosirea și gestionarea acestor obiecte. Fiecare clasă de colecție conține o metodă de **loadAll** care încarcă toate rândurile din tabelul respectiv din baza de date, apoi returnează array-ul de obiecte de acelaș tip. Aceste metode sunt folosite pentru încărcarea datelor meteorologice.

Clasa care reprezintă utilizatorii sistemului este clasa **User**. Această clasă conține toate informațiile unui utilizator care se găsește în baza de date, pentru aceste date conține metode de seter și geter, aceste metode ajută la accesarea proprietăților private al clasei. În afară de metodele geter și seter clasa conține următoarele metode.

- Metoda **loadByUsername()** încarcă utilizatorul din baza de date care are numele de utilizator identic cu parametrul metodei, această metodă setează parametrii obiectului curent care este apelat.
- Metoda **loadByEmail()** încarcă utilizatorul din baza de date care are adresa de email identic cu valoarea primită ca paramatru, metoda setează parametrii obiectului curent.
- Metoda **save()** salvează utilizatorul în baza de date. Metoda salvează datele obiectului curent, pe care este apelat metoda.

- Metoda **getLoactions()** returnează lista locațiilor salvate ale utilizatorului curent.
- Metoda **addLocation()** adaugă o locație în baza de date cere e legal la utilizatorul curent.

5.4.3. Aplicația de actualizare a datelor meteorologice.

Aplicația de actualizare a datelor meteorologice este responsabilă pentru actualizarea datelor meteorologice în daza de date. Această aplicație este scrisă în limbajul de prrogramare python, am ales limbajul python, pentru că organizație ECMWF are biblioteci pentru acest limbaj care ajută descărcarea și citirea datelor meteorologice.

Aplicația de actualizare a datelor meteorologice este alcătuită din două părți, una descarcă datele meteorologice de pe serverele ECMWF și salvează într-un fișier .grib folosind technologia MARS request. Folosind librăria ECMWFDataServer putem conecta la setul de date foarte ușor.

Scriptul care descarcă datele meteorologice este fișierul **mars.py.** În acest fișier înainte de a descărca datelor meteorologice calculează data curentă folosind librăria datetime pentru limbajul Python. Înterogarea MARS salvează datele descărcate în fișierul **mars.grib** după salvarea acestui fișier se termină execuția scriptul de descărcare al datelor meteorologice.

```
server.retrieve({
    "class": "e2",
    "dataset": "era20c",
    "date": "2010-" + month + "-" + day,
    "expver": "1",
    "levtype": "sfc",
    "param": "15.128/134.128/137.128/167.128",
    "stream": "oper",
    "time": "12:00:00",
    "type": "an",
    "target": "mars.grib",
})
```

Figura 5.9 Interogarea MARS

A doua parte a aplicației de actualizare al datelor meteorologice este un script scris în limbajul de programare Python care citește datele din fișierul mars.grib și salvează datele meteorologice în baza de date relațională prin actualizarea rândurilor din baza de date. Pentru citirea fișierului mars.grib scriptul folosește librăria pygrib scris pentru limbajul de programare Python. În primul pas al executării scriptul adaugă în baza de date în tabelul cron_log un rând în care setează date de început ale scriptului și ca și pas final setează data terminării scriptului, astfel putem verifica dacă scriptul a rulat cu succes și datele meteorologice sunt actualizate.

Cele două componente al aplicației de actualizare al datelor meteorologice este executat de n linux shell script, un shel script conține comenzi de linie de comandă linux, care sunt executate în rând. Acest script execută scriptul mars.py apoi scriptul saveECMWFDData.py.

```
#!/bin/bash
###################################
#
# Update meteorological data cron
#
###################################

echo "Downloading ECMWF data..."
python mars.py
echo "Download complete."

echo "Save ECMWF data relational..."
python getECMWFData.py
```

Figura 5.10 Codul sursa al scriptului cron.sh

Pentru rularea scriptului cron.sh este definit un cronjob care rulează scriptul astfel aplicația de actualizare al datelor meteorologice în fiecare zi.

Capitolul 6. Testare şi Validare

În acest capitol sunt prezentate procesele de testare şi validare ale sistemului. Scopul testării al unui program este găsirea bug-urilor, funcționalităților care nu sunt implementate conform cererii sistemului și cazurilor extreme în care sistem-ul nu funcționează corect. Prin aceste procese se poate asigura corectitudinea unui program. Găsirea acestor probleme ajută în îmbunătățirea sistemului, dar și în definirea limitelor aplicațiilor. Testarea și validarea al unui sistem este prezent de la începutul implementări, nu numai produsul final este testat, testarea este prezent în fiecare stare al dezvoltării unui sistem. După fiecare problemă rezolvată procesul de testare al funcționalității corectate începe din nou. În lumea software-urilor fiecare sistem conține bug-uri, și procesul de testare și validare poate fi infinit.

În acest capitol sunt prezentate procesele de testare atăt pentru aplicația de client cât și pentru aplicația server-side. Procesele de testare și validare pot fi manuale și automatizate.

6.1. Testarea manuală a aplicației

Testarea manuală a aplicațiilor este efectuată după implementarea unor funcționalități noi, și după fiecare bug-fix al funcționalității dacă e necesar. Testarea manuală a sistemelor încearcă să reproducă toate variațiile de utilizare a sistemului pe care pot fi efectuate de către utilizatori. Testarea manuală se începe cu o verificare vizuală, se verifică dacă toate textele sunt scrise corect, se verifică dacă schema vizuală este bine implementată și design-ul aplicației în ansamblu. După inspecția vizuală se începe testarea funcțională a sistemului.

6.1.1. Black Box Testing

Testarea funcțională a sistemului reprezintă testarea funcționalităților implementate, conform specificațiilor stabilite. La testare black box tester-ul nu are acces la codul de sursă, numai aplicația cu funcționalitățile implementate și documentația acestora stă la disponibilitatea tester-ului. Prin această metodă tester-ul modelează un utilizator al sistemului, astfel tester-ul are o vedere exterioară asupra aplicației.

6.1.2. White Box Testing

Testarea white box este opusul metodei de black box testing, în sensul că tester-ul are acces la codul de sursă al aplicației. Folosind această metodă de testare tester-ul urmăreste codul sursă și pot fi testate părți al unui funcționalitate, prin verificare output-ului al unei block al sistemului. Prin metoda de testare white box putem descoperi problemele aplicațiilor mai precis, în schimb la metoda black box tester-ul observă numai că funcționalitatea nu funcționaeză corect, iar la metoda white box tester-ul poate specifice modului sau funcția care nu funcționează corect.

6.1.3. Scenarii de test

În continuare sunt prezentate niste scenarii de test al sistemului, cere au fost urmărite la testarea aplicației.

Caz de testare 1: Autentificarea utilizatorului în sistem

Descriere: Utilizatorul trebuie să poată să se autentifice în sistem cu ajutorul al unor perechi de nume de utilizator și parolă care este regăsită în baza de date, deci utilizatorul a fost înregistrat in sistem anterior.

Scenariul 1:

1. Se oprește conexiunea la Internet
2. Se accesează aplicația
3. Se apasă butonul de LogIn
4. Se apasă buton-ul de LogIn în fereastra de pop-up

Rezultat așteptat: Un mesaj de erroare din cauza conexiunii la Inernet este afișat.

Scenariul 2:

1. Se pornește conexiunea la Internet
2. Se accesează aplicația
3. Se apasă butonul de LogIn
4. Se apasă buton-ul de LogIn în fereastra de pop-up

Rezultat așteptat: Se afișează mesaje de eroare sub câmpurile de input, câmpurile de nume de utilizator și parolă trebuie completate obigatoriu.

Scenariul 3:

1. Se accesează aplicația
2. Se apasă butonul de LogIn
3. Se completează câmpul nume de utilizator cu date incorecte.
4. Se apasă buton-ul de LogIn în fereastra de pop-up

Rezultat așteptat: Se afișează mesajul de erroare, care indică că numele de utilizator nu se regăsește în baza de date.

Scenariul 4:

1. Se accesează aplicația
2. Se apasă butonul de LogIn
3. Se completează câmpul nume de utilizator
4. Se completează câmpul parola, cu date incorecte
5. Se apasă buton-ul de LogIn în fereastra de pop-up

Rezultat așteptat: Se afișează mesajul de erroare, care indică că numele de utilizator cu parola introdusă nu se regăsește în baza de date.

Scenariul 5:

1. Se accesează aplicația
2. Se apasă butonul de LogIn
3. Se completează câmpul nume de utilizator
4. Se completează câmpul parola
5. Se apasă buton-ul de LogIn în fereastra de pop-up

Rezultat așteptat: Utilizatorul se autentifică în sistem, se reîncarcă pagina aplicației.

Caz de testare 2: Căutarea locațiilor

Descriere: Utilizatorul autentificat trebuie să poată să caute locații care vor fi afișate pe hartă, cu detaliile meteorologice asociate.

Scenariul 1:

1. Se oprește conexiunea la Internet
2. Se accesează aplicația
3. Se întroduce numele locației căutate

Rezultat așteptat: Motorul de căutare nu afișează sugesții.

Scenariul 2:

1. Se pornește conexiunea la Internet
2. Se accesează aplicația
3. Se întroduce numele unei locații inexistente

Rezultat așteptat: Motorul de căutare nu găsește locația, și nu oferă o listă de locații din care utilizatorul poate elege una.

Scenariul 3:

1. Se accesează aplicația
2. Se întroduce numele locației căutate
3. Se selectează unul dintre sugestiile motorului de căutare
4. Între timp scade conexiunea la Internet, sau se oprește calculatorul care conține aplicația server.

Rezultat așteptat: Locația căutată nu poate fi salvat și nu apare pe hartă.

Scenariul 4:

1. Se accesează aplicația
2. Se întroduce numele locației căutate
3. Se selectează unul dintre sugestiile motorului de căutare

Rezultat așteptat: Apare locația căutată pe hartă și este salvată în baza de date.

6.2. Testarea automată al aplicației

Testarea automatizată al unui sistem necesită tool-uri de testare. Aceste tool-uri pot interpreata codul sursă al aplicației și să aplice o serie de teste pe sistemul testat. Testarea automată a sistemelor se verifică convențiile aplicate la scrierea codului sursă. Testarea automată folosită în proiect este PHPUnit test, care este un tool de testare al codului scris în limbajul de programare PHP. Acesta metodă testează structura de fișiere al aplicației, numele de clase, numele metodelor. Pot fi setate cazuri de test care aplică automat un set de inputuri și verifică dacă sistemul returnează rezultatele specificate.

Testarea performanței al aplicațiilor poate fi realizat cu tool-uri specifice de măsurare al performanței sistemelor, prin măsurarea timpului de execuție, memorie și alte resurse hardware utilizate la execuția sistemelor.

Un tool care a fost utilizat pentru testarea performaței a sistemului este aplicația Selenium, cu care putem automatiza și modela condițiile de browser. Selenium suportă cele mai mari sisteme browser. Alte tooluri folosite pentru testarea automată al aplicaților web sunt: sggPlant, Sahi, Siculi, TestComplete și Test Studio.

6.3. Dificultăți întâmpinate și rezolvarea acestora

În cursul dezvoltării sistemului au apărut probleme ne prevăzute care au forțat schimbări în construcția sistemului. Deoarece contul utilizat prin care au fost accesate datele OSM a prezentat limitării în utilizare mai mult chiar datele nu au fost accesibile pe perioada dezvoltării proiectului, a fost necesar găsirea unui alt furnizor. Tool-urile alese pentru conectarea la setul de date meteorologică nu au putut fi utilizate pe sistemul Windows și a fost necesar translatarea implementării pe sistemul Linux.

Capitolul 7. Manual de Instalare si Utilizare

În acest capitol sunt prezentate resursele necesare pentru instalarea și rularea aplicației, pașii al instalării și îndrumătorul de utilizare al aplicației.

7.1. Cerințele aplicației

7.1.1. Resurse hardware

Cerințele hardware necesare pentru rularea applicației este deținerea unui server cu sistem de operare Linux instalat. Serverul trebuie sa conțină cel puți 2Gb de memorie, specificațiile serverului pot varia în funcție de numărul așteptat al utilizatorilor.

7.1.2. Resurse software

Serverul web pe care instalăm aplicația trebuie să conțină un LAMPP simplu instalat ce se referă la Linux Apache MySQL PHP Python.

Calculatorul de server trebuie sa conțină un server web, poate fi folosit Apache sau orice altă web server, manualul de instalare va fi exemplificat cu web server-ul Apache. Aplicația are nevoie de o bază de date locală unde se salvează datele de utilizator, și datele meteorologice afișate, baza de date MySQL este folosită pentru îndeplinirea cerințelor. Applicația de server side este scrisă in limbajul de programare PHP, din cauza asta pe sistemul de Linux trebuie sa fie instalat interpretorul de PHP cu librariile mcript, libapache și cu conectorul de MySQL. Aplicația conține un program care actualizează datele meteorologice automate, în fiecare zi, folosind protocolul de Mars Request, API-ul pentru acest protocol este dezvoltat în limbajul de programare Python, deci trebuie instalat pe server și interpretorul pentru limbajul Python insoțit cu pachetele pyproj, numpy și conectorul de MySQL. Librăria care ajută în citirea fișierului de .grib de asemenea este scrisă in limbajul Python.

7.2. Pașii de instalare

7.2.1. Crearea host-ului in Apache

Host-ul trebuie să fie configurat în felul ca numele de domain dorit să redirectioneze clienții la directorul care conține sursele aplicației. Configurarea unui host constă in crearea unui fișier de configurare și autorizarea acestui fișier de configurare.

Fișierul de configurare conține urmatoarele informații. în primul rând este definit portul pe care se va comunica server-ul, portul 80 este portul implicit pentru comunicarea cu protocolul HTTP. În următoarele linii este definit numele serverului care e defapt numele de domain, directiva ServerName va conține domain-ul de baza iar directiva ServerAlias va conține numele de domaim impreună cu prefixul ‘www.’. Directiva DocumentRoot trebuie să conține patch-ul directorului care conține sursele aplicației, si directiva ServerAdmin conține email-ul administratorului de server.

Hostul trebuie activat folosind comanda *a2ensite* cu un singur parametru care trebuie să fie numele fișierului de configurare. Dupa repornirea serverului de web accesând domain-ul definit suntem apți să acessam sursele aplicației.

7.2.2. Crearea bazei de date

Baza de date inițială este salvată în fișierul db.sql, acest fișier conține datele inițiale ale aplicației si structura bazei de date relațională. Acest fișier trebuie rulat pe serverul aplicației.

7.2.3. Instalarea ECMWF Api

Serviciul ECMWF Api permite utilizatorii autorizații sa recuperă și listează datele MARS din afara facilității de ECMWF.

Serviciul poate fi instalat în două feluri, felul recomandat de instalare este folosind comanda pip care va descărca sursele automat și va configura libraria. Comanda care va face toate astea este următorul:

```
sudo pip install https://software.ecmwf.int/wiki/download/attachments/56664858/ecmwf-api-client-python.tgz.
```

A doua metodă de instalare este descarcarea surselor și rularea următorului comenzii:

```
export PYTHONPATH=~/mars/lib:$PYTHONPATH
```

Ca să putem folosi API-ul instalat, trebuie să fie activat cu o cheie de API. Pentru recuperarea cheia de API trbuie sa avem un cont la ECMWF, dupa logarea in site gasim cheia la următorul link: https://api.ecmwf.int/v1/key/. Informația gasită trebuie sa fie salvată in fișierul $HOME/.ecmwfapirc. [17]

```
{
    "url"   : "https://api.ecmwf.int/v1",
    "key"   : "3b4524ce1a0b57ff0d947aef2f10f6f3",
    "email" : "miklos_erik@yahoo.com"
}
```

Figura 7.1 Conținutul fisierului de configurare ECMWF Api

7.2.4. Instalarea librariei pygrib și Grib Api

Libraria pygrib este un pachet pentru Python care este un modul pentru scrierea și citirea fișierelor cu extensia grib. Modulul necesita o instalare de Python 2.4 sau mai nou intsoțit cu pachetele pyproj și numpy și libraria C Grib API de la ECMWF care realizează decodarea și codarea datele din fișierele de .grib.

Instalarea pachetului pygrib se poate făcut urmărind următoarele pași. Se cloneaza repzitoriul de pe github.com/jswhit/pygrib sau se descarcă sursa librariei de la pypi.python.org/pypi/pygrib. Se copiaza conținutul fișierului setup.cfg.template in fișierul setup.cfg. Se rulează următoarea comanda într-un terminal 'python setup.py build'. Se execută comanda 'python setup.py install'. Pentru testarea instalării se poate rula script-ul 'python test.py'. [15]

7.2.5. Setarea Cron-ului

Un cron este un program in sistemele Linux care se rulează scripturile configurate periodic. Formatul unui cron este următorul *(minute) *(ore) *(zile) *(luni) *(ziua din săptămână) (patch-ul la script). Un cron poate fi setat urmărind următorii pași, executăm comanda sudo su, executăm comanda crontab -l. Adăugăm linia de configurare, exempul nostru de cron este * 1 * * * /var/www/LICENTA/cron.sh care inseamnă că script-ul nostru va fi rulat în fiecare zi la ora 01:00. Dupa salvarea modificărilor, cron-ul este configurat.

7.3. Utilizarea aplicației

După instalarea aplicației și configurarea serverului aplicația este accesibilă la numele de domain setat din orice browser. În acest capitol va fi prezentat cu ajutorul capturi de ecran modurile in care aplicația poate fi manipulată.

7.3.1. Afișarea detaliilor al unui punct de pe harta

In fiecare layer afișat utilizatorii autorizații sunt apți sa se află mai multe detalii despre un punct existent de pe hartă. Fiecare marcer poate fi apasat, ca să apară un pop-up cu informațiile de temperatură, presiune atmosferica, radiația UV și precipitația. La urmatorul click fereastra de pop-up dispare. Aceasta funcționalitate este disponibil numai pentru utilizatorii autentificații.

Figura 7.4 Informații meteorologice detaliate

7.3.2. Înregistrarea si autentificarea în aplicație

Aplicația poate fi utilizat și ca utilizator guest, utilizatorul autentificat are mai multe drepturi și poate accesa mai multe date meteorologice. Accesând pagina applicației apare harta, pagina principală al aplicației.

Pentru a înregistra sau a autentifica in aplicație trebuie apăsat butonul de LogIn din colțul stanga jos. Dudpă apăsarea butonului apare o fereastră de pop-up în mijlocul fereastrei de browser. Utilizatorul se poate autentifica întrodus numele de utilizator și parola apoi apasând buton-ul de Log In din fereastra pup-up. Numele de utilizator și parola trebuie sa fie regăsite in baza de date. În cazul în care utilizatorul nu este înregistrat în aplicație se poate înregistra prin apăsarea butonului Register din fereastra pop-up de login, dupa apăsarea butonului fereastra de pop-up devine extinsă cu câmpuri adiționale necesare pentru înregistrarea utilizatorului. După introducerea datelor de prenume, nume, nume de utilizator, parolă și adresa de emial apasând butonul de Sign Up utilizatorul nou este creat și salvat în baza de date, fereastra de pop-up revine la starea inițială cu numele de utilizator și parola deja întrodusa. Dupa apăsarea butonului de Log In se reâncarca pagina și în colțul stânga jos în loc de butonul LogIn apare butonul de LogOut.

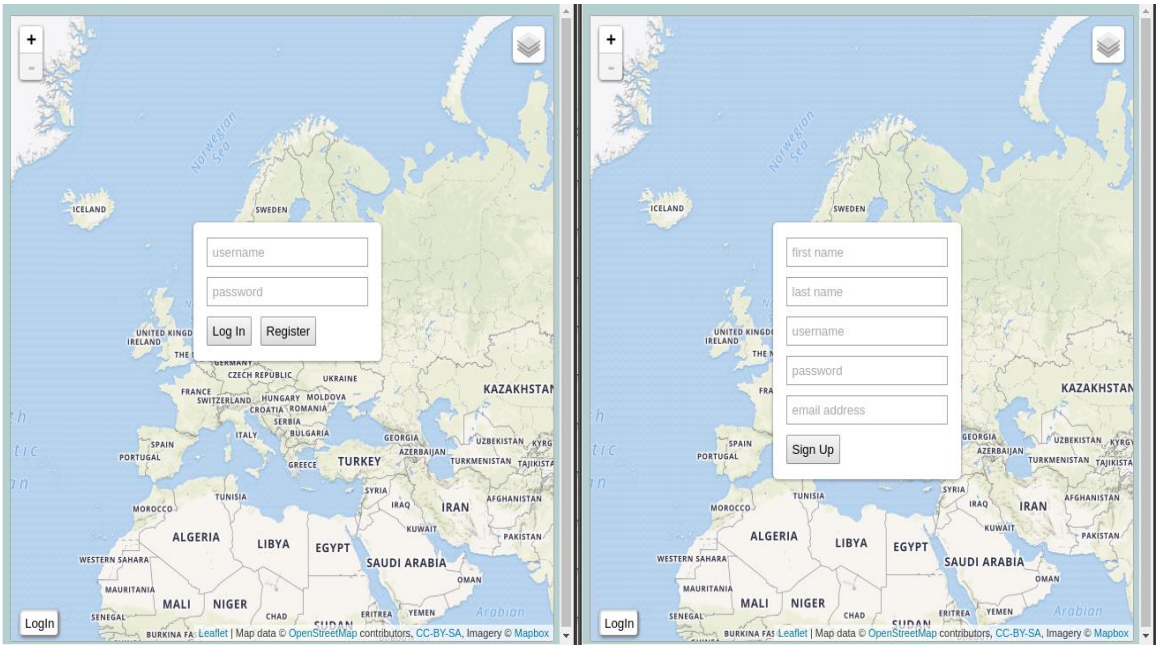

Figura 7.2 Fereastra pop-up pentru înregistrare și autentificare în sistem

7.3.3. Selectarea layer-ul afișat pe hartă

Aplicația ne permite afișarea patru feluri de informații meteorologice, astea pot fi selectate cu ajutorul controlerului de layer. Un singur layer poate fi selectat deodată. Pentru utilizatorii neautentificați este accesibilă numai layer-ul de temperatura. Layer-ele de presiune atmosferică, radiația UV și de precipitație sunt accesibile numai pentru utilizatorii autentificați.

Figura 7.3 Afișarea datelor meteorologice în aplicație

7.3.4. Cautarea unei locații pe hartă

Căutarea unei locații pe hartă este realizată cu extensia Google Search aplicată pe harta OSM. Localitatea căutată trebuie introdusă în câmpul de text găsit în dreapta paginii. După apăsarea tastei enter apare un punct pe hartă care primește datele marker-ului cel mai apropiat de la rezultatul căutării. Harta se focusează la locația căutată după apăsarea tastei Enter, deci punctul căutat ajunge în mijlocul ferestrei. Această funcționalitate poate fi accesată de către utilizatorii autentificați.

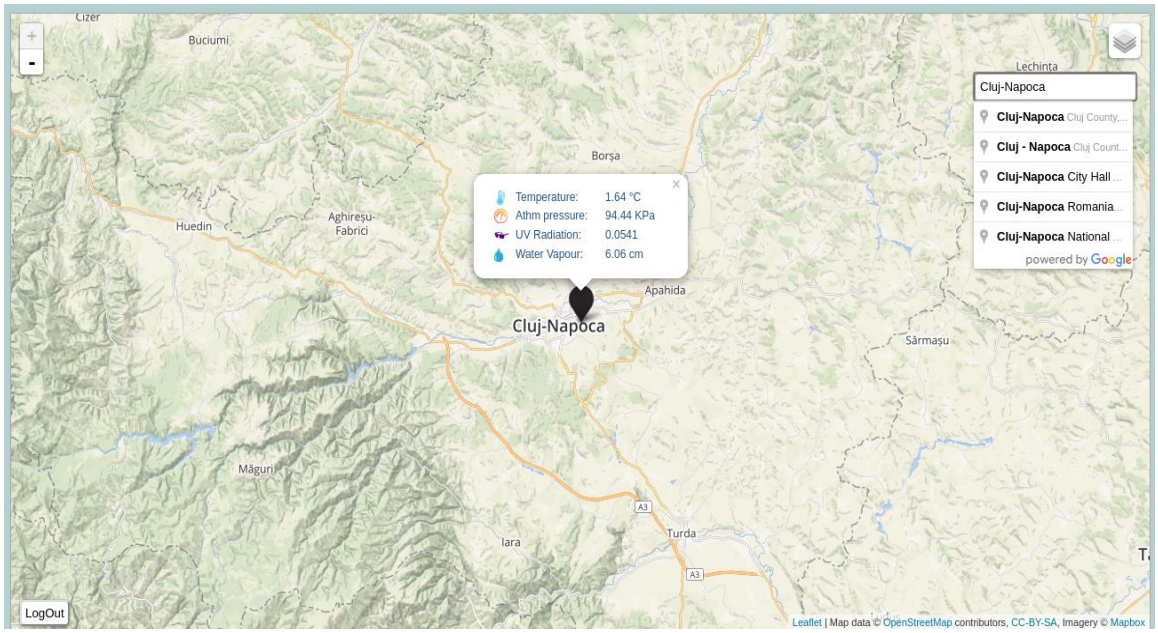

Figura 7.5 Căutare locațiilor în aplicație

Capitolul 8. Concluzii

În acest capitol sunt prezentate realizările obiectivelor și cerințelor sistemului, și descrierea dezvoltării ulterioare.

8.1. Realizări

Sistemul dezvoltat îndeplinește scopul, de a fi un sistem care afișează datele meteorologice pe o hartă digitală online, și poate actualiza datele meteorologice automat și periodic.

Sistemul dezvoltat folosește harta digitală online de la OpenStreetMaps la care se conectează cu librăria Leaflet care este scris în limbajul JavaScript. Sistemul descarcă datele meteorologice din setul de date ECMWF folosiind înterogări MARS, această interogare descarcă datele într-un fișier .grib care este citit cu ajutorul librariei pygrib scris pentru libmajul de programare Python.

Sistemul este alcătuit din două părți, o aplicație care apdatează datele meteorologice în baza de date, și o aplicație client-server care ajută la afișarea datelor. Datele sunt afișate în browser-ul utilizatorului.

Prin dezvoltarea sistemului au fost parcurse următoarele etape.

- S-au studiat tehnologiile care pot fi folosite la dezvoltarea explicației
- S-au studiat și analizat technologiile existente de mapare digitală, s-au studiat mai detaliat proiectul OpenStreetMap
- S-a analizat piața de seturi de date meteorologice, am aflat de organizația ECMWF care oferă o paletă de tool-uri care ajută la încorporarea în sistem
- S-au analizat sistemele similare
- S-a dezvoltat architectura și conceptul sistemului și conform acestor informații s-a planificat construcția sistemului
- S-a facut implementarea tuturor obiectivelor principale și cele specifice, care sunt următoarele:
- Înregistrarea utilizatorilor
- Autentificarea utilizatorilor
- Selectarea informației afișată pe hartă
- Căutarea locațiilor, și salvarea căutărilor
- Afișarea informațiilor detaliate al fiecărui punct de pe hartă
- Clusterizarea datelor meteorologice

8.2. Dezvoltări ulterioare

Sistemul pot fi folosit în starea curentă de implementare, dar sistemul este capabil la evoluare și acceptă usor extensii, funcționalității noi.

Sistemul poate fi extins cu mai multe informații meteorologice, setul de date oferă mai mult decâr 50 tipuri de date meteorologice. Informații noi pot fi adaugate în sistem ușor.

Sistemul în starea curentă salvează locațiile căutate de către utilizatori, iar utilizatorii nu pot șterge căutările lor.

În starea curentă sistemul descarcă date meteorologice din anul 2007, din cauza că un utilizator simplu al setului de date ECMWF nu are acces la datele curente și cele mai noi. Cu un cont de utilizare mai avansat pot fi descărcate și datele de prognoză meteorologică din setul de date ECMWF.

Sistemul folosește algoritmul de clusterizare Kmeans, acest algoritm este un algoritm de clusterizare mai simplu, pentru a îmbunătăți performața sistemului algoritmul folosit de sistem poate fi actualizat la unul mai avansat și mai complex, ca exemplu metode bazate pe metrica Minkowski care folosește calcule complexe și probabilități pentru gruparea datelor, un exemplu de algoritm bazat pe metrica Minkowski este algoritmul Expectation-Maximization.

Bibliografie

[1] T. Lethbridge and R. Laganiere. "Object-Oriented Software Engineering" (2nd edition). McGraw-Hill, 2005. http://www.lloseng.com

[2] Ioan Salomie, Tudor Cioara, Ionut Anghel, Tudor Salomie "Distributed Computing and Systems", Albastra Publishing House 2008, 367 pages, ISBN 978-973-650-234-7

[3] Anil K. Jain, Richard C. Dubes "Algorithms for Clustering Data", Michigan State University, 1988 Prentice-Hall

[4] Sylvain Bouveret, Philippe Genoud, Niklas Petersen. "A quick glimpse at OpenStreetMap", Séminaire de l'équipe LIG-Steamer, Friday, April the 12th, 2013

[5] Kraak, Menno-Jan and Allan Brown (2001), "*Web Cartography – Developments and prospects*" Taylor & Francis, New York, ISBN 0-7484-0869-X

[6] "K-Means Clustering Algorithm", *Saravanan Thirumuruganathan*, January 27, 2010, https://saravananthirumuruganathan.wordpress.com/2010/01/27/k-means-clustering-algorithm/

[7] Vivek Gite, "Add Jobs To cron Under Linux or USIX", *nixCraft*, May 19, 2015, https://www.cyberciti.biz/faq/how-do-i-add-jobs-to-cron-under-linux-or-unix-oses/

[8] Aleks Buczkowski, "Why would you use OpenStreetMap if there is Google Maps?", October 23, 2015, http://geoawesomeness.com/why-would-you-use-openstreetmap-if-there-is-google-maps/

[9] Baudouin Raoult, "Architecture of the new MARS server", Meteorological Applications ECMWF, Februarie, 2017

[10] Carsten Maass, "MARS-Introduction and basic concepts", ECMWF, Martie 3, 2015

[11] The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition, http://pubs.opengroup.org/onlinepubs/007904975/utilities/crontab.html

[12] MySQL Internals Manual, https://dev.mysql.com/doc/internals/en/

[13] ECMWF Web API Home, https://software.ecmwf.int/wiki/display/WEBAPI/ECMWF+Web+API+Home

[14] MARS service, https://software.ecmwf.int/wiki/display/WEBAPI/MARS+service

[15] Module pygrib, https://jswhit.github.io/pygrib/docs/pygrib-module.html

[16] Leaflet, http://leafletjs.com/

[17] MARS user documentation, https://software.ecmwf.int/wiki/display/UDOC/MARS+user+documentation

[18] What is PHP?, https://translate.google.com/#ro/hu/What%20is%20PHP%3F

[19] Why Learn Python?, http://www.bestprogramminglanguagefor.me/why-learn-python

[20] Python documentation, https://www.python.org/doc/

[21] Crontab – Quick Reference, http://www.adminschoice.com/crontab-quick-reference

[22] OSM Data Overview, http://learnosm.org/en/osm-data/data-overview/

[23] OpenStreetMap, http://www.openstreetmap.org/about

[24] A Tutorial on Clustering Algorithms, http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/
[25] Global surface temperature data sets, https://climatedataguide.ucar.edu/climate-data/global-temperature-data-sets-overview-comparison-table
[26] Global Historical Climatology Network (GHCN), https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/global-historical-climatology-network-ghcn.
[27] Netamo weather map, https://weathermap.netatmo.com/
[28] WeatherLink, https://www.weatherlink.com/map.php
[29] Wudermap, https://www.wunderground.com/wundermap
[30] JavaScript, https://en.wikipedia.org/wiki/JavaScript
[31] Apache web server, https://www.apache.org/foundation/

Capitolul 9. Anexa 1

Glosar de termeni

API	Applicationn Programming Interface
GIS	Geographic Information System
CGI	Computer-Generated Imagery
PARC	Palo Alto Research Center
USGC	United States Global Change
NASA	National Aeronautics and Space Administration
ESRI	Environmental Systems Research Institute
OSM	OpenStreetMap
NOAA	National Oceanic and Atmospheric Administration
GHCN	Global Historical Climatology Network
ECMWF	Centre for Medium-Range Weather Forecast
HTTP	Hypertext Transfer Protocol
MARS	Meteorological Archival and Retrieval System
SQL	Structured Query Language
GPS	Global Positioning System
POI	Point(s) Of Interest
XML	Extensible Markup Language
PBF	Protocol Buffered Binary Format
JOSM	Java OpenStreetMap
AJAX	Asynchronous JavaScript and XML
HTLM	Hypertext Markup Language
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
PHP	Hypertext Preprocessor
CERN	European Organization for Nuclear Research
UML	Unified Modeling Language
XHR	XMLHttpRequest
LAMPP	Linux Apache MySQL PHP and Python

Capitolul 10. Anexa 2

Lista figurilor din lucrare

Lista tabelelor din lucrare