KLAIPĖDOS UNIVERSITETAS
JŪRŲ TECHNIKOS IR GAMTOS MOKSLŲ FAKULTETAS
INFORMATIKOS IR STATISTIKOS INŽINERIJOS KATEDRA

JUSTAS JAGMINAS

METEOROLOGINIŲ DUOMENŲ RINKIMO IR VIZUALIZAVIMO SISTEMOS PROTOTIPO KŪRIMAS

DEVELOPMENT OF A PROTOTYPE FOR METEOROLOGICAL DATA COLLECTION AND VISUALISATION SYSTEM

Techninių informacinių sistemų inžinerijos studijų programos 621E15004
magistro baigiamasis darbas

Klaipėda, 2017

MAGISTRO BAIGIAMOJO DARBO LYDRAŠTIS

Pildo bakalauro baigiamojo darbo autorius

Justas Jagminas
(bakalauro baigiamojo darbo autoriaus vardas, pavardė)

Meteorologinių duomenų rinkimo ir vizualizavimo sistemos prototipo kūrimas
(bakalauro baigiamojo darbo pavadinimas lietuvių kalba)

Patvirtinu, kad bakalauro baigiamasis darbas parašytas savarankiškai, nepažeidžiant kitiems asmenims priklausančių autorių teisių, visas baigiamasis bakalauro darbas ar jo dalis nebuvo panaudotas Klaipėdos universitete ir kitose aukštosiose mokyklose.

Justas Jagminas
(bakalauro baigiamojo darbo autoriaus vardas, pavardė ir parašas)

Sutinku, kad bakalauro baigiamasis darbas būtų naudojamas neatlygintinai 5 m. Klaipėdos universiteto studijų procese.

Justas Jagminas
(bakalauro baigiamojo darbo autoriaus vardas, pavardė ir parašas)

Pildo bakalauro baigiamojo darbo vadovas

Bakalauro baigiamąjį darbą ginti .....
(įrašyti – leidžiu arba neleidžiu)

..... doc. dr. Gediminas Gricius
(data) (bakalauro baigiamojo darbo vadovo vardas, pavardė ir parašas)

Pildo katedros, kuruojančios studijų programą, administratorius (sekretorius)

Baigiamasis darbas įregistruotas katedroje .....
(data)

Laima Brazdeikienė
(katedros sekretorės vardas, pavardė ir parašas)

Pildo katedros, kuruojančios studijų programą, vedėjas

Bakalauro baigiamąjį darbą ginti .....
(įrašyti – leidžiu arba neleidžiu)

..... prof. dr. Arūnas Andziulis
(data) (katedros vedėjo vardas, pavardė ir parašas)

Recenzentu(-ais) skiriu .....
.....
(įrašyti recenzento(ų) vardą, pavardę)

..... prof. dr. Arūnas Andziulis
(data) (programos svadovas. vardas, pavardė ir parašas)

TURINYS

PAVEIKSLŲ SĄRAŠAS 7
LENTELIŲ SĄRAŠAS 8
ĮVADAS 9
1. PLŪDURŲ SISTEMŲ ANALIZĖ 11
1.1. Tyrimų instrumentai 11
1.2. Duomenų surinkimo technika 13
1.3. Duomenų perdavimas 16
1.4. Bangų modeliavimas 16
1.5. Duomenų apdorojimo sistemų architektūra 18
2. DUOMENŲ VIZUALIZAVIMO METODŲ ANALIZĖ 20
2.1. Šepardo interpoliacija 20
2.2. Krigingas 21
2.3. Delaunay trianguliacija 21
2.4. Daugiamačių duomenų vizualizavimas 22
3. METODINĖ DALIS 24
3.1. Reikalavimai sistemai 24
3.2. Sistemos architektūra 25
3.3. Duomenų bazė 27
3.4. Duomenų apdorojimo ir perdavimo serveris 28
3.5. Žemėlapių atvaizdavimo biblioteka 29
3.6. Duomenų perdavimo formatas 31
3.7. Šablonų variklis ir teksto ir duomenų atvaizdavimui 32
3.8. Kūrimo aplinka ir darbe naudojami įrankiai 33
3.9. Techninė serverio įranga 35
4. PROTOTIPO ĮGYVENDINIMAS 36
4.1. Duomenų bazės kūrimas 37
4.2. Serverio programinės įrangos kūrimas 38
4.3. Žemėlapio realizavimas su OpenLayers 40
4.4. Duomenų apdorojimas 41
4.5. Gradiento kūrimas 43
4.6. Surinktų duomenų atvaizdavimas 46
4.7. Interaktyvių žemėlapio elementų kūrimas 48
4.8. Duomenų redagavimas 50
5. PROTOTIPO BANDYMAI 52
5.1. Prototipo bandymas 52
5.2. Bandymo rezultatai 52
6. IŠVADOS 54
7. SISTEMOS TOBULINIMO GALIMYBĖS 55
8. LITERATŪRA 56
SANTRAUKA 58
9. PRIEDAI 1
1. Kompaktinė plokštelė 2

SANTRUMPŲ IR TERMINŲ ŽODYNĖLIS

API (angl. *Application Programming Interface*) – aplikacijų programavimo sąsaja, kurią suteikia kompiuterinė sistema, biblioteka ar programa tam, kad programuotojas per kitą programą galėtų pasiekti jos funkcionalumą ar apsikeistų su ja duomenimis.

Asinchroninė funkcija – funkcija, kurią pradėjus vykdyti, programa tęsia darbus nelaukdama funkcijos vykdymo pabaigos.

Biblioteka (angl. *library*) - tai resursų rinkinys programinei įrangai kurti. Jos viduje – konstantos, kintamieji, duomenų tipai, funkcijos ir pan.

Buferis (angl. *buffer*) - laikina kompiuterio atminties vieta arba įtaisas, reikalingas duomenų mainų spartai, duomenų blokų matmenų ir kt. skirtybėms suderinti.

CSS (angl. *Cascading Style Sheets*) – kalba, skirta nusakyti kita struktūrine kalba aprašyto dokumento vaizdavimą. Dažniausiai *CSS* aprašomas *HTML* dokumentų pateikimas, tačiau ją galima taikyti ir įvairiems kitiems *XML* dokumentams.

DOM (angl. *Document Object Model*) – daugiaplatformis ir nepriklausomas nuo programavimo kalbos modelis, kuris aprašo įvykius su objektais *HTML*, *XHTML* ir *XML* dokumentuose.

GPS (angl. *Global Positioning System*) – globalinė padėties nustatymo sistema, Leidžianti nustatyti objekto koordinates bet kurioje pasaulio vietoje visomis oro sąlygomis, kuomet yra galimybė gauti signalus iš keturių ar daugiau palydovų. Sistemos pagrindas – IT technologijų sąveika su planetą gaubiančiu GPS palydovų tinklu.

Grafinė vartotojo sąsaja (angl. *graphical user interface [GUI]*) – grafikos priemonėmis pagrįsta sąsaja tarp žmogaus ir kompiuterio.

HTML (*Hyper text Markup Language*) - kompiuterinė žymėjimo kalba, kurią standartizuoja *W3C* konsorciumas, naudojama pateikti turinį internete.

JavaScript – objektiškai orientuota programavimo kalba, besiremianti prototipų principu. Dažniausiai kalba naudojama internetinių puslapių interaktyvumo realizacijai, bet taip pat naudojama ir kaip galimybė skriptais manipuliuoti tam tikromis programomis.

Karkasas (angl. *framework*) –komponentas, kuris suteikia kitoms programoms galimybę naudotis daugybe jau paruoštų įvairių bibliotekų (pvz., duomenų bazių komponentus, formų komponentus). Be to, karkasas tvarko programos kodą jos vykdymo metu, jei programa parašyta specialiai šiam paketui.

Komandinės eilutės sąsaja (angl. *command-line interface (CLI)*) – kompiuterio sąsaja su vartotoju, kurioje vartotojas užduotis užduoda iškviesdamas komandas teksto forma.

PostgreSQL – viena iš reliacinių duomenų bazių valdymo sistemų, palaikanti daugelį naudotojų, dirbanti SQL kalbos pagrindu.

Modeliavimas (angl. *modelling*) - tiriamojo objekto savybių pakartojimas kitame objekte (modelyje) norint geriau pažinti tiriamąjį objektą. Modeliuojama tada, kai neįmanoma objekto ištirti tiesiogiai arba jį tirti dėl kokių nors priežasčių sudėtinga. Modelis turi būti panašus į tiriamąjį objektą fizinėmis arba funkcinėmis savybėmis. Šiuo požiūriu modeliavimas susijęs su analogijos metodu.

Monitoringas (angl. *monitoring*) - sistemingas tam tikro svarbaus stebėjimas, renkant informaciją, reikalingą sistemos valdymui, reiškinių paieška ir aptikimas.

NodeJS – atviro kodo serverio pusės ir tinklo aplikacijų programavimo aplinka, kurios programavimo kalba yra *JavaScript*.

Pasisveikinimas (angl. *handshake*) – procesas, kuris vyksta kai du įrenginiai pradeda komunikuoti. Vienas įrenginys siunčia pranešimą kitam pranešdamas, kad nori užmegzti ryšį, o kitas į tai atsako.

Pažadas (angl. promise) – asinchroninė duomenų struktūra, naudojama palengvinti asinchroninių užduočių sprendimui.

Atgalinis skambinimas (*angl. callback*) – programinis kodas, pateikiamas kaip funkcijos argumentas ir įvykdomas funkcijai baigus veikti.

Programinis kodas (angl. *program code*) – bet kokia sakinių seka, užrašyta žmogui suprantama programavimo kalba.

Prototipas (angl**.** *prototype*) - pirmasis egzempliorius, pavyzdys, pagal ką daromi visi vėlesni tokie objektai.

RDBMS (angl. *Relation Database Management System*) – reliaciniu modeliu paremta duomenų bazės valdymo sistema.

SQL (angl. *Structured Query Language*) – struktūrizuota užklausų kalba, populiariausia iš šiuo metu naudojamų kalbų, skirtų aprašyti duomenis ir manipuliuoti jais reliacinių duomenų bazių valdymo sistemose.

PAVEIKSLŲ SĄRAŠAS

1 pav. Duomenų asimiliavimo sistema [2] 10
2 pav. Galimi duomenų išgavimo būdai [2] 11
3 pav. METOCEAN architektūra [4]. 13
4 pav. „Seawatch mini II“ plūduro architektūra [3]. 14
5 pav. Plūduras su prie jo pritvirtintais jutikliais [5]. 14
6 pav. Praktinio bangų matavimo aspektai [7]. 16
7 pav. Duomenų apdorojimo sistema [8]. 18
8 pav. Delaunay trianguliacijos pagrindinis kriterijus: a atveju trianguliacija atliekama teisingai, b atveju – klaidingai [13] 21
9 pav. a – tiesinė projekcija, b – netiesinė projekcija. 22
10 pav. Duomenų surinkimo ir vizualizavimo sistemos struktūra 24
11 pav. Sistemos prototipo sudedamosios dalys 25
12 pav. PHPStorm Interaktyvioji kūrimo aplinka 33
13 pav. Duomenų bazės diagrama. 37
pav. 14 Delaunay trianguliacijos algoritmas 43
15 pav. Trikampio vidurio taško radimas ant sferos paviršiaus [27] 44
16 pav. Trikampių fraktalizavimo algoritmas 45
17 pav. Stiliaus funkcijomis stilizuoti duomenys 47
18 pav. Išlendantis langas žemėlapyje 48
19 pav. Plūdurų matavimų redagavimo aplinka 50
20 pav. Oro temperatūra, 2015-10-18 12:00 52
21 pav. Vandens temperatūra, 2015-10-18 12:00 52
22 pav. Atmosferos slėgis, 2015-10-18 12:00 52

LENTELIŲ SĄRAŠAS

1 lentelė. Reikalavimai sistemai 23
2 lentelė: Sistemos prototipo elementų aprašymas 25
3 lentelė. Reikalinga programinė įranga sistemos realizacijai 35
4 lentelė. Panaudoti NPM moduliai 35

ĮVADAS

Atliekant aplinkos ir oro sąlygų matavimus iš automatinių jutiklių platformų, reikalinga duomenis surenkanti ir suprantamai atvaizduojanti programinė įranga, gebanti kaupti, saugoti, apdoroti ir atvaizduoti surinktus duomenis.

Įprastai naudojamos autonominės jutiklių platformos būna didelių gabaritų, stacionarios, retai taisomos ar tobulinamos. Tokių platformų priežiūra ir gamyba daug kainuoja, todėl paprastai jos naudojamos po vieną.

Oro sąlygų matavimui vis dažniau pasitelkiamos išmaniosios technologijos, elektroniniai temperatūros, banguotumo, kritulių kiekio, vėjo krypties ir stiprumo, banguotumo, o kartais ir cheminės oro ar vandens sudėties jutikliai. Dėl sumažėjusių jutiklių ir kitų elektroninių elementų naudojimo kaštų ir matmenų, oro sąlygų tyrimams pasitelkiamos nedidelės, bet daug jutiklių turinčios platformos.

Jutiklių gaunami duomenys priklauso nuo aplinkos sąlygų ir jutiklių kokybės, o iš jutiklių gaunamų duomenų ne visada lengva padaryti išvadas apie esamas, o tuo labiau apie būsimas aplinkos sąlygas. Oro sąlygas daug lengviau suprasti jas atvaizdavus grafiškai. Oro sąlygų atvaizdavimui įvairiose vietose itin naudingi meteorologinių sąlygų žemėlapiai.

Grafiniai meteorologiniai žemėlapiai sudaromi iš jutiklių platformų, meteorologinių stočių ir palydovų duomenų. Jei naudojami vektorinių žemėlapių kūrimo metodai, žemėlapiai kuriami naudojant duomenų išmatavimo vietas ir tam tikras vertes turintys plotai išsaugomi kaip geometriniai objektai. Geometriniais objektais laikomi taškai, daugiakampiai ir kitos geometrinės figūros, kurių viršūnės turi geometrines koordinates. Tokio tipo geometriniai objektai saugomi duomenų bazėse, o kai reikia juos atvaizduoti – rasterizuojami (vektoriniai žemėlapiai paverčiami pikseliniais) žemėlapių kūrimo programų dėka.

Klaipėdos universitete kuriami plūdurai, matuojantys oro ir vandens temperatūrą, banguotumą ir panašius meteorologinius parametrus. Duomenų kaupimo ir vizualizavimo sistema turės gebėti surinkti plūdurų duomenis, juos išsaugoti duomenų bazėje, atlikti duomenų interpoliaciją ir pateikti interpoliacijos rezultatus žemėlapyje.

Tyrimo objektas – oro sąlygų duomenų kaupimo sistema.

Darbo tikslas – sukurti duomenų kaupimo ir vizualizavimo sistemą, kaupiančią meteorologines sąlygas matuojančių plūdurų duomenis.

Tikslui pasiekti keliami **uždaviniai:**

1. atlikti panašių sistemų analizę;
2. ištirti naudojamus vizualizavimo metodus;
3. pasirinkti programinės įrangos kūrimo įrankius;

4. realizuoti sistemos prototipą;
5. atlikti prototipo bandymus vizualizuojant matavimų duomenis.

Tyrimo metodai ir priemonės

- lyginamoji mokslinės literatūros analizė;
- duomeninio programavimo (angl. *data-oriented programming*) metodika;
- *PHPStorm 2016.3.2* programų kūrimo aplinka;
- *PostgreSQL 9.4.2.* duomenų bazė;
- *NodeJS 6.10.2 LTS* karkasas;
- *OpenLayers 4.0.1.* karkasas;
- *VueJS 2.*0 karkasas.

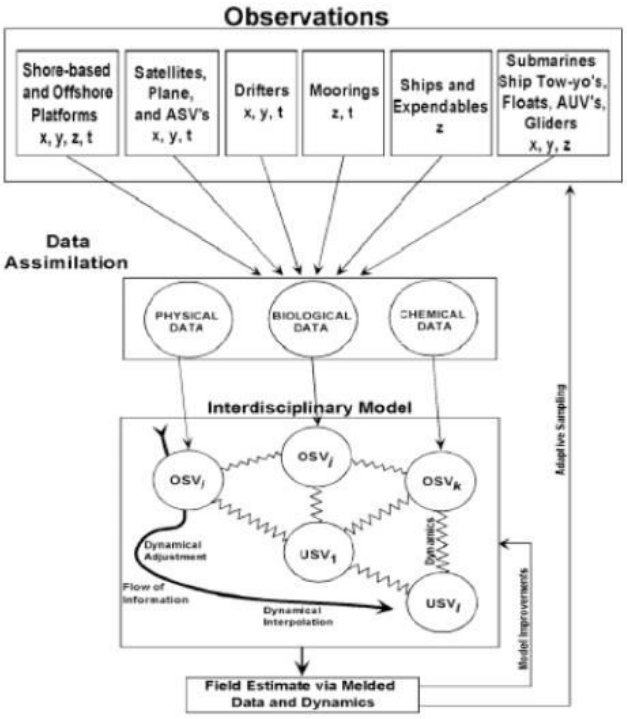
1. PLŪDURŲ SISTEMŲ ANALIZĖ

Baltijos jūra yra nedidelė vidinė jūra, smarkiai veikiama žmogaus sukeltų veiksnių, tokių kaip globaliniai klimato pokyčiai, didelis maisto medžiagų nutekėjimas, tarša, amunicijos laidojimas, intensyvi žvejyba ir įvairios inžinerinės modifikacijos, įskaitant pakrančių miestų augimą, hidro ir atomines elektrines, dideles vėjo jėgainių fermas, dujotiekius bei tiltus. Tuo pat metu, Baltijos jūros baseinas naudojamas įvairioms paskirtims, tokioms kaip intensyvi žemdirbystė, laivyba ir rekreacija.

Dėl šių priežasčių, auga žinių apie jūros ekosistemas bei hidrometeorologinius reiškinius, poreikis. Žinios apie Baltijos jūros sistemą sukuria žinių bazę, padedančią suprasti įvairius reiškinius bei ryšius tarp jų, tokių kaip transformaciniai procesai, kuriems įtaką daro vanduo, šiluma ir energijos apsikeitimo ciklai [1].

1.1. Tyrimų instrumentai

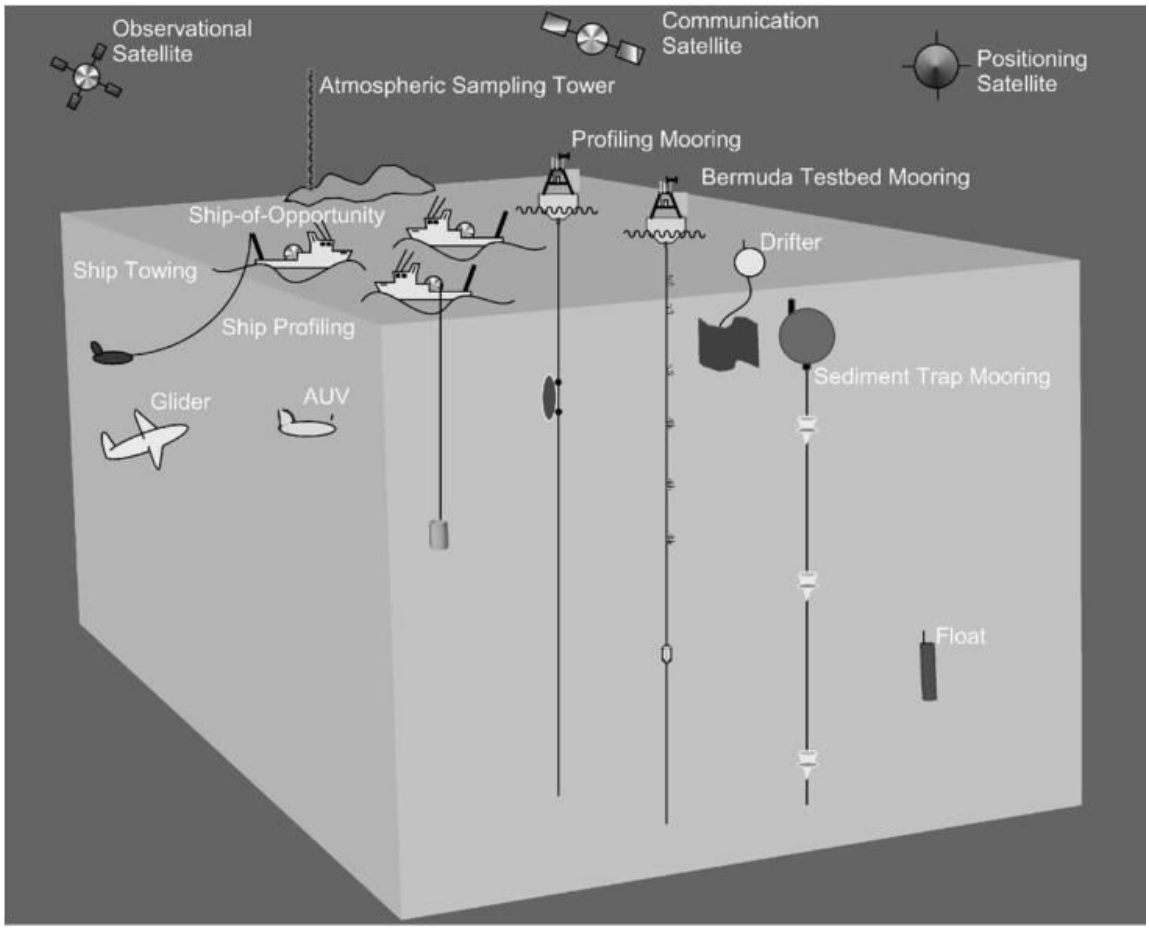
Duomenų asimiliavimo sistema paprastai sudaryta iš trijų pagrindinių elementų: stebėjimo tinklo su duomenų telemetrijos galimybe, tarpdisciplininio modelio ir duomenų asimiliavimo schemos. Tarpdisciplininiai duomenys yra surenkami ir perduodami realiu laiku. Kiekviena platforma turi savo esmines galimybes ir apribojimus priklausomai nuo naudojamų erdvės ir laiko matavimų ribų.



1 pav. Duomenų asimiliavimo sistema [2]

Duomenys ir galimos paklaidų vertės yra perduodamos į tarpdisciplininį modelį. Tarpdisciplininis dinaminis modelis yra sudarytas iš skirtingų modelių, pavyzdžiui, fiziniams, biologiniams ir cheminiams veiksniams išmatuoti. Viena iš galimų asimiliavimo sistemų architektūrų pavaizduota 1 pav.

Kiekvienas modulis sudarytas iš lygčių rinkinio su vertėmis, atitinkančiomis reikšmingus procesus. Svarbu tai, kad šie modeliai yra suporuoti. Duomenų jungimo žingsnis sukuria sistemos būseną apibrėžiančius kintamuosius ir numatomų parametrų kintamuosius, užfiksuotiems fiziniams kintamiesiems (OSV_i), užfiksuotiems biologiniams kintamiesiems (OSV_j), užfiksuotiems cheminiams kintamiesiems (OSV_k) ir neužfiksuotiems kintamiesiems (USV_s) kartu su jutiklių paklaidomis. Šie duomenys turi grįžtamąjį ryšį su duomenų asimiliavimo modeliu tam, kad būtų panaudoti kitoje modelio iteracijoje ir modelio tikslumo gerinimui. Taip pat gauti rezultatai perduodami stebėjimo tinklui tam, kad leistų atlikti adaptyvų duomenų parinkimą (pvz. tam, kad būtų pašalinami didžiausias paklaidas turintys rezultatai). Adaptyvus duomenų parinkimas turi kelis galimus keitimo būdus: duomenų gavimo dažnio keitimas, jutiklių ir sistemų svarbos modeliui pakeitimas ir mobilių duomenų rinkimo prietaisų perkėlimas į svarbias vietas (pvz. atmosferos frontus) [2]. Galimi duomenų išgavimo būdai pateikti 2 pav.



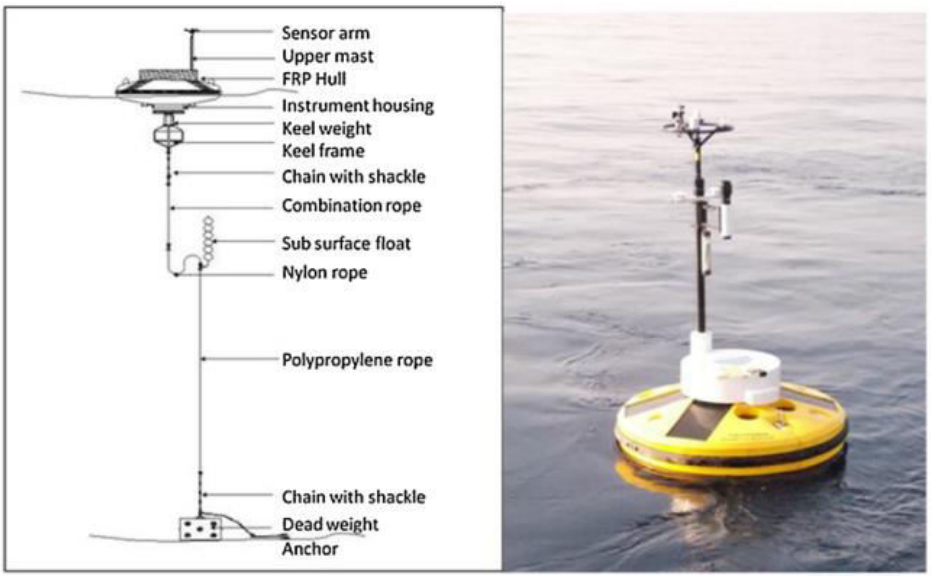
2 pav. Galimi duomenų išgavimo būdai [2]

Mobilūs bangų matavimo plūdurai plačiai naudojami išmatuoti paviršinėms bangoms jūrose ir vandenynuose. Bangas matuojantys plūdurai naudojami ilgalaikių duomenų rinkimui siekiant išmatuoti bangų klimatą ir trumpalaikių duomenų rinkimui specifiniams bangų tyrimams. Šioms paskirtims buvo sukurti įvairių formų ir tipų plūdurai, taip pat ir įvairūs matavimo būdai. Didelių matmenų plūdurai naudojami vandenynuose, o nedidelių matmenų plūdurai naudojami vietovėse netoli krantų. Kaip plūduriuojančios jutiklių platformos, plūdurai gali būti pritaikyti nuolatiniam duomenų perdavimui ir plūdurų tinklas geba pateikti realaus laiko informaciją, svarbią jūros sąlygų nustatymui arba pakrančių stebėjimo programoms.

Jūros bangų tyrimai taip pat svarbūs dėl kuriamos atsinaujinančios energetikos infrastruktūros. Kuriamoms sistemoms reikės išsamių žinių apie bangų charakteristikas bei jas supančias aplinkos sąlygas. Šios žinios darys įtaką kuriamiems prietaisams, energetikos statinių išdėstymui, aptarnavimui ir eksplotavimui. Taip pat, galimo išgauti energijos kiekio ir prietaisų apkrovimo prognozės būtų daug tikslesnės turint tikslesnius bangų matavimus. Aplinkos monitoringo ir įvertinimo projektai naudoja vietovėje esančių bangų duomenis iš plūdurų, kurie gali pateikti optimalų duomenų apie vietovės, kurioje jie šiuo metu yra, būseną [3].

1.2. Duomenų surinkimo technika

Plūdurų sistemos METOCEAN ir OMNI MSB architektūra pavaizduota 3 pav. Schemoje ir nuotraukoje pavaizduotas ant stiebo įmontuotų meteorologinių instrumentų rinkinį, kurie matuoja oro slėgį, vėjo greitį, kryptį, oro temperatūrą ir santykinę drėgmę. Plūduras, užpildytas poliuretano ir pagamintas iš sustiprinto plastiko yra inkaru pritvirtintas prie jūros dugno. Inkarui nejudėti padeda papildomas balastas. Inkaras su plūduru sujungtas grandinių ir virvių sistema, kurioje stačią inkaro poziciją palaiko plūdurai virvės viduryje. Tokia inkaro ir plūduro sistema patikrinta pažangiais matavimo ir stebėjimo metodais ir turėtų veikti be gedimo vidutiniškai 12 metų. Centrinėje, cilindro formos plūduro dalyje įmontuotos baterijos ir duomenų surinkimo sistema, sujungta su matavimo prietaisais. Surinkti duomenys perduodami į valdymo centrą palydoviniu ryšiu. Duomenų gavimo sistema atitinka vibracijų ir aplinkos stresų standartus.



3 pav. METOCEAN architektūra [4].

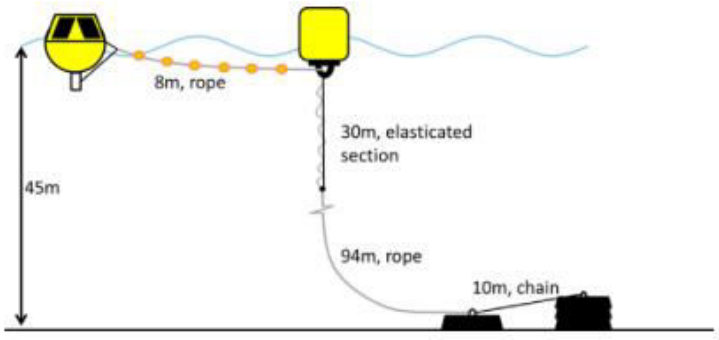
Plūduro sistemos, įskaitant jutiklius, vidinius energijos kaupiklius, duomenų gavimo sistemą, telemetriją ir įtvirtinimą pozicijoje naudojamos gana ilgai ir yra brandžios technologijos. Energijos kaupimo sistema sudaryta iš saulės elementų maitinamų švino ir rūgšties elementų ir dviejų ličio-silicio-chlorido baterijų. Visi naudojami prietaisai atitinka patikimumo ir saugumo standartus [4].

Kitas, panašios architektūros plūduras yra „Seawatch mini II“, nebrangus ir gaminamas kompanijos Fugro Oceanor. Šie plūdurai sukurti sekti jūros paviršiaus judėjimą. Todėl, plūduro judėsių matavimas gali būti panaudotas paviršiaus bangų kryptingam judėjimui išgauti. Pagreičiai išmatuojami naudojant triašius kietosios būsenos akselerometrus. Sujungus šiuos matavimus kartu su tuo pat metu išgautais bangų krypties ir plūduro pasvirimo matavimais, pagreičiai yra padalinami į aukščio (vertikalią), rytų ir šiaurės ašis. Šie duomenys vėliau išfiltruojam ir du kartus integruojami, siekiant gauti pozicijų šiose ašyse seką.

Šis duomenų apdorojimas atliekamas pačio plūduro, naudojant „wavesense“ kompiuterį, esantį pačiame plūdure. Jis sukurtas taip, kad veiktų su jūros bangų dažniu ir nepridėtų naujų, neapibrėžtų kintamųjų į sistemos atliekamus matavimus ar gaunamus signalus. Dėl to, idealios perdavimo funkcijos nurodytos gamintojo ir vartotojui jų keisti nereikia. Šio plūduro jutiklių konfigūracija nėra unikali ir naudojama ir kitų panašių įrenginių, įskaitant TRIAXYS bangų plūdurą, bet naudojamas kitas, taip pat komercinis, duomenų procesorius. Verta pažymėti išimtis yra „Datawell Waverider“ plūduras, kuris naudoja gravitacijos stabilizuojamą platformą vertikalių judėsių matavimui.

Plūdurų inkaravimo dizainas dažniausiai leidžia plūdurams judėti laisvai kartu su bangų judėsiais, tačiau neleidžia potvyniams ir jūros srovėms nusinešti plūduro. Tačiau plūduro inkaras išlieka galimu netikslių matavimų šaltiniu, ypač kai matavimai atliekami netoli nuo kranto. Kartais naudojamas daugiau nei vienas inkaras, sujungti tarpusavyje grandine, kaip pateikta 4 pav.

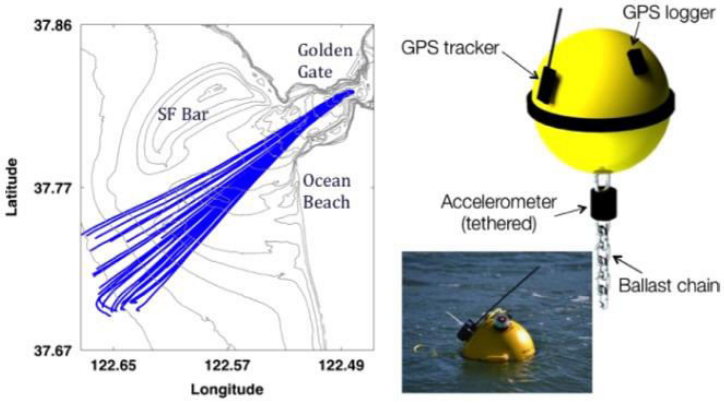
Paprastai plūdurą ir jo inkarą skiria lynas, sudarytas iš elastingos ir neelastingos dalies. Pavyzdžiui, galima naudoti 30 metrų ilgio guminį lyną, kartu su atsargniu lynu. Ši elastinga sekcija šiuo atveju jungiama prie papildomo, 1 metro aukščio, kubo formos plūduro, kuris su instrumentiniu plūduru sujungtas plūduriuojančia, horizontalia, polipropileno virve. Tuo tarpu prie elastingosios virvės kito galo tvirtina kita, neelastinga virvė. Visas inkaro virvės ilgis turėtų būti tris kartus ilgesnis, negu vietovės, kurioje atliekami matavimai, gylis [3].



4 pav. „Seawatch mini II“ plūduro architektūra [3].

Kartais naudojami ir plūdurai, sudaryti iš atskirų jutiklių modelių. Pavyzdžiui, galima naudoti plūdurą, prie kurio primontuotas Locosys GT-31 GPS imtuvas, naudojantis SIRF III mikroschema, renka duomenis 1 herco dažniu ir tiksliai apskaičiuoja horizontaliąją bangos komponentę. Papildomai, horizontalaus greičio apskaičiavimai gali būti išgauti iš GPS signalo L1 fazės pasikeitimo dėl Doplerio efekto. Iš Doplerio greičio duomenų išgauti greičio matavimai laikomi daug tikslesniais nei diferencinio pozicionavimo duomenys ir tai žymiai padidina aukšto dažnio vėjo ir bangų matavimus.

GT-31 GPS jutikliai yra nebrangūs ir geba efektyviai apskaičiuoti bangų energiją ir kryptį. Tačiau jie netinka tiksliam vertikalių judėsių apskaičiavimui. Nors bangų statistiką galima nuspėti ir naudojant tik horizontalius judėsius, tai reikalauja linijinių bangų teorijos, kuri mažiau patikima esant labai stačioms bangoms ir bangų krypties nustatymai gali būti nevisada tikslūs[5].



5 pav. Plūduras su prie jo pritvirtintais jutikliais [5].

1.3. Duomenų perdavimas

Naudojant stacionarias, o ne mobilias platformas, galima naudoti ir kitokius matavimo prietaisus. Pavyzdžiui, atvirojo kodo platforma „Xbeach“ naudojama matuoti ekstremalių audrų paplūdimiams daramą žalą ir pakrančių zonos hidrodinamiką. Xbeach apskaičiuoja bangų propagaciją, vandens srautus, nuosėdų transportavimą ir jūros dugno pokyčius. Xbeach modelis turi nestacionarų bangų matuoklį, kuris išmatuoja bangų poveikį ir iš jo duomenų išgaunama pakrančių srovių dinamika.

Xbeach platforma gana plačiai ir sėkmingai naudojama ir pasižymi hibridiniu stebėjimo ir modeliavimo sprendimu, kuriuo simuliuojamos paplūdimio bangos ir srovės. Xbeach naudojamas modelis turi kelis lengvai parenkamus parametrus, kurie naudojami modeliui sukalibruoti. Kai bandoma matuoti tik hidrodinaminius duomenis, šie parametrai apibrėžiai srovių judėjimą. Modelio hidrodinamika apskaičiuota iš realiai gauto Eulerio duomenų rinkinio ir simuliuotų pajūrio cirkuliacijos duomenų (tekėjimo greičių, krypčių ir srovės dydžio), kurie buvo palyginti su Lagrandžo plūdurų matavimais.

Buvo atliktos kelios modelių simuliacijos, naudojant bangų charakteristikas iš 2012 metų spalio 8 dieną surinktų duomenų. Modelio parametrai, kurie nusako bangų lūžimą buvo sukonfigūruoti taip, kad sumažintų skirtumą tarp realių duomenų ir modelio išvesties.

Aukščiausias tikslumas pasiektas su gama verte γ=0.4, kuri nusako, kada banga lūžta ir beta verte β = 0.2, kuri nusako, kokį aukštį pasiekia banga ir jos lūžimo intensyvumą [6].

1.4. Bangų modeliavimas

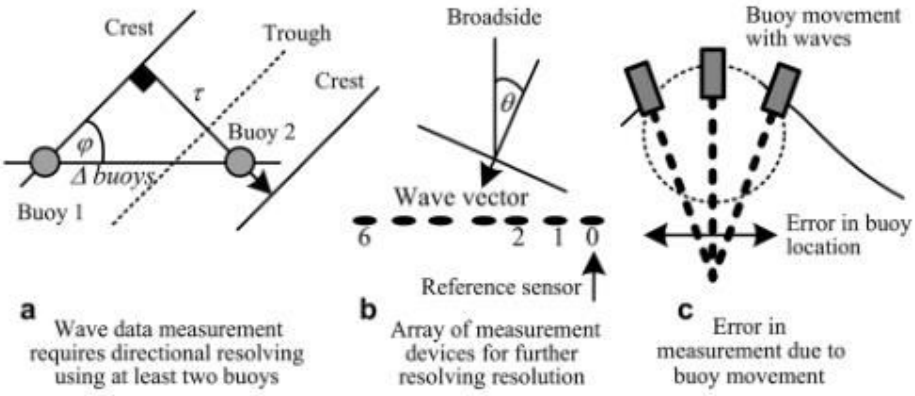
GPS jutiklio matavimams pagerinti ir vertikalių judėsių tikslumui padidinti, panaudotas trijų ašių akselerometras (X6–2 MEMS), pagamintas „Gulf Coast Data Concepts“ buvo pridėtas prie jutiklių paketo. Akselerometras matuoja vertikalius pagreičius iki 72 g, kur g yra laisvojo kritimo pagreitis, lygus 9,81 m/s^2, veikiantis su 0.001 m/s^2 paklaida ir atliekantis patavimus 10 hercų dažniu. Šis akselerometras tinka apskaičiuoti bangų pagreičiams, kurie gali būti tik nedidelė dalis g. Kadangi X6-2 yra atskiras akselerometras, o ne dalis inercinio matavimo vieneto (IMU), jis neapskaičiuoja kūno orientacijos, todėl gali turėti klaidų susijusių su plūduro pasisukimu. Šios klaidos sumažintos pritvirtinus jutiklį tarp plūduro ir balasto grandinės, sumažinančios judėsius, sukeltus jūros paviršiaus nelygumų.

Akselerometro išmatuojamų bangų matavimai yra triukšmingi esant mažiems judėjimo dažniams, kuriuose signalo ir triukšmo santykis nukenčia nuo mažų pagreičių (mažas ilgų bangų statumas) ir veikiamos nelinijinių plūduro judėsių [5].

Siekiant realizuoti oro nuspėjimo technologijas, reikia matavimo metodo, kuris tiksliai išmatuotų paviršiaus aukštį (bangų aukštį) fiksuotoje vietoje. Ekonomiškai efektyviausias metodas yra naudoti kryptinių bangų matavimo plūdurus inkaruotus tam tikru atstumu nuo bangas generuojančio prietaiso, kad būtų išvengta atsispindinčių bangų. Galima apsvarstyti ir kitas technologijas.

Pavyzdžiui, palydoviniai arba iš skraidančio aparato atlikti matavimai suteikia fiksuotus paviršiaus matavimus, bet galėtų būti pernelyg brangūs. LIDAR prietaisas (Šviesos gavimo ir atstumo matavimo, *angl. Light Detection and Ranging*) gali suteikti fiksuotus matavimus, bet turi būti įmontuotas tam tikrame aukštyje virš aukščiausio galimo bangų aukščio, siekiant išvengti bangų šėšėlių efekto. Atliekant matavimus giliose ir audringose jūrose, pvz. Šiaurės Jūroje, reikėtų aukšto profilio platformos, ant kurios tokio tipo prietaisas būtų primontuotas. Tokie reikalavimai patenkinami tik dideliose jutiklių platformose su dideliais statiniais. Neseniai sukurta alternatyva yra akustiniai doplerio srovės profiliuotojai (angl. *Acoustic Doppler Current Profiler*), bet jie tikslūs tik nedideliame gylyje.

Kryptiniai bangų plūdurai nepateiks fiksuoto atskaitos taško dėl savo judėjimo ir atsako į inkarą. Šios plūdurų klaidos galėtų būti ištaisytos naudojant signalų apdorojimą ir įmontuotą aukšto tikslumo akselerometrą. GPS naudojančių plūdurų atsiradimas padeda sekti taško judėjimą ir atstatyti fiksuoto taško įrašus [7].



6 pav. Praktinio bangų matavimo aspektai [7].

Vienas iš pirmųjų žingsnių analizuojant duomenų seką, kurios trukmė T sekundžių yra naudoti Furjė *(angl. Fourier)* transformaciją. Visų pirma, pagal šį metodą, galima laikyti, kad laiko funkcija išplėsta iki begalybės virš originalaus įrašo ilgio. Pagal standartinę teoriją, greitoji Furjė transformacija paverčia laiko funkciją $f(\Delta t \rightarrow T)$ į harmoniškai susijusių sinusoidžių sumą su unikaliomis amplitudėmis F_n ir fazėmis φ_n.

Paprastose nuspėjimo aplikacijose, ši sinusoidinė funkcija gali būti naudojama kartu su paprasta visakrypte linijine bangos lygtimi, kurios dėka būtų įmanoma perskaičiuoti sinusoidžių

sumas bet kokiu metu, didesniu už T_s. Šiai rekonstrukcijai pasiekti naudojama dažna Furjė bangų spektro lygtis su aukščiu, dažniu ir faze:

$$\xi(x,t) = \sum_{n=1}^{N} F_n \cos(k_n x - \omega_n t - \phi_n)$$

Kur F_n yra sinusoidės amplitudė, k_n yra bangos numeris, φ_n yra sinusoidės fazė ir

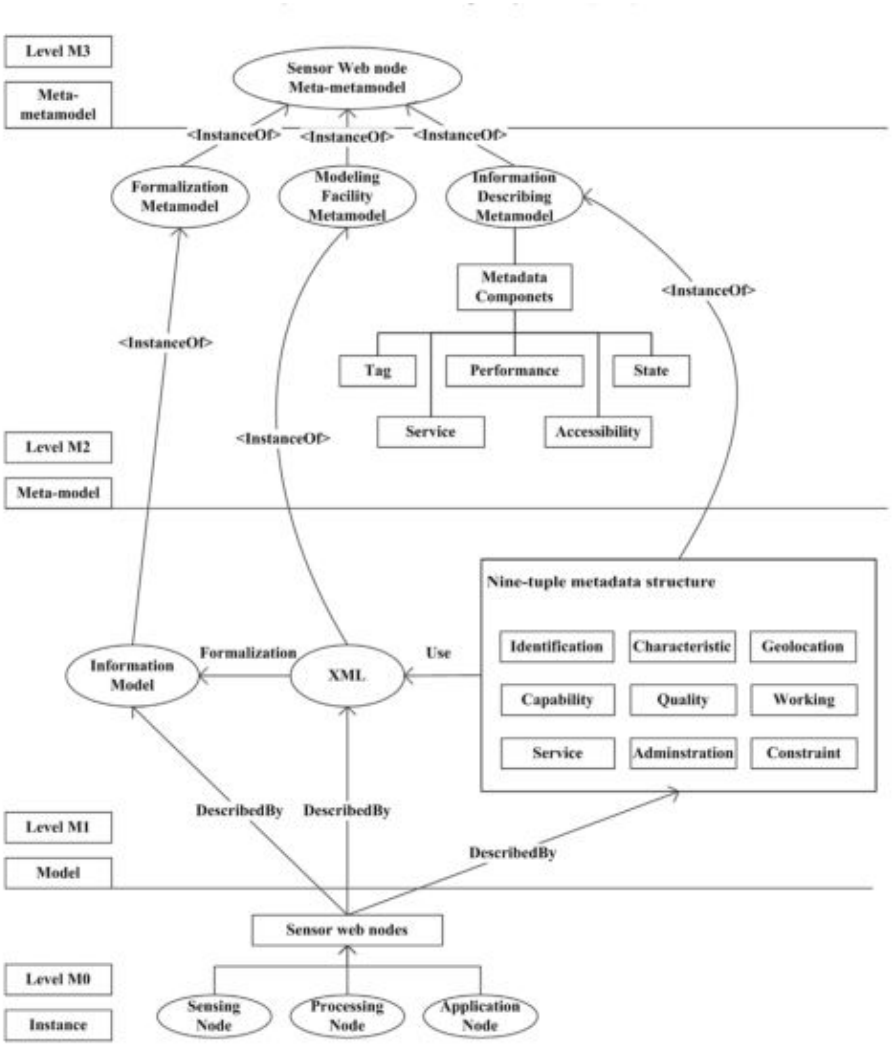
$$\omega_n^2 = g k_n \tanh(k_n h)$$

Yra santykio dispersija [7].

1.5.Duomenų apdorojimo sistemų architektūra

Sistemos, kurios gauna duomenis iš jutiklių, juos saugo ir apdoroja naudoja įvairias architektūras. Viena iš tokių architektūrų vadinama heterogeninių mazgų metamodeliu. Ji sudaryta iš jutiklinių, apdorojimo ir pritaikymo mazgų.

Jutikliniai mazgai yra sistemos naudojami, jutiklius turintys įrenginiai, dažnai įmontuoti įvairiose platformose, kurių tikslas yra išmatuoti įvairius tiriamos sistemos parametrus. Jutikliniais mazgais galima laikyti palydovus, lėktuvus, laivus, antžemines važiuokles ir stotis. Apdorojimo mazgais laikomi įrenginiai, gebantys saugoti ir apdoroti jutiklinių mazgų duomenis ir juos naudoti modelių kūrimui, duomenų jungimui, simuliacijai ar prognozavimui. Pritaikymo mazgais laikomi paslaugų ištekliai, skirti įvairioms sritims, tokioms kaip hidrologiniai ir atmosferos stebėjimai, skirti išankstiniams pavojaus pranešimams ar sprendimų priėmimui.



7 pav. Duomenų apdorojimo sistema [8].

Pvz. Potvynių nuspėjimo sistemos, naudojančios jutiklių tinklą, atveju, duomenys surenkami iš antžeminių hidrologinių, kritulių surinkimo ir meteorologinių stočių. Per prognozavimo modelius ir duomenų žemėlapius, sudarytus apdorojimo mazgų dėka ir pateiktus į pritaikymo mazgus, tinklas gali teikti potvynių prognozavimo paslaugą ir teikti duomenis apie oro sąlygas [8].

2. DUOMENŲ VIZUALIZAVIMO METODŲ ANALIZĖ

Surinkus duomenis, reikia juos atvaizduoti ant žemėlapio. Sensorių duomenis užrašyti ant žemėlapio nesudėtinga, tačiau daug daugiau sako izolinijomis ar panašiais metodais atvaizduoti aplinkos sąlygų parametrai, tokių kaip temperatūra. Duomenų vizualizavimui naudojami įvairūs metodai, išmatuotų geografinių taškų vertes paverčiantys tam tikrą vertę turinčiais plotais. Toks duomenų konvertavimas vadinamas variatine interpoliacija. Dažniausiai naudojami interpoliacijos metodai: Šepardo interpoliacija, krigingas ir Delaunay trianguliacija.

2.1. Šepardo interpoliacija

Šepardo interpoliacija – atvirkščią atstumui svorį naudojantis interpoliacijos algoritmas, plačiai naudojamas praktikoje ir gerai veikiantis su daug triukšmo turinčiais duomenimis. Šepardas apibrėžė nepertraukiamą funkciją, kur pasvertas vidurkis yra atvirkščiai proporcingas atstumui nuo interpoliuotos vietos. Algoritmas numano, kad kuo toliau taškai D_i yra nuo interpoliuotos pozicijos P, tuo mažesnį poveikį jie turės interpoliuojamai reikšmei.

Svoriai duomenims priskiriami pagal svėrimo laipsnį, apibrėžiantį kaip svorį veikiantys faktoriai mažėja atstumui nuo P didėjant. Kuo didesnis svėrimo laipsnis, tuo mažesnę įtaką nutolę taškai turi interpoliuojamam rezultatui. Laipsniui didėjant, interpoliuojama reikįmė tiesiog prisiima artimiausio kaimyno metodui, kur interpoliuojama reikšmė prisiima artimiausio kaimyno matavimo vertę. Šepardo reikšmė globaliai modeliuojamam paviršiui yra:

$$\mathrm{f}_1(P) = \begin{cases} \dfrac{\sum_{i=1}^{N} \left[(d_i)^{-u} \times z_i\right]}{\sum_{i=1}^{N} (d_i)^{-u}} & \text{if} \quad d_i \neq 0 \; \forall D_i (u > 0) \\ z_i & \text{if} \quad d_i = 0 \end{cases}$$

Kur d_i yra atstumas nuo P iki D sunumeruotas indeksu i N žinomų taškų sekoje ir z_i yra žinoma matavimo vertė taške D_i. Eksponentė u naudojama apibrėžti interpoliacijos lygumui. Kai P artėja prie D_i, d_i artėja prie nulio, o i-tojo taško skaitiktis ir vardiklis viršyja ribas, kai kitos vertės ribų neturi. Todėl naudinga riba $lim_{P \to Di}\, f_1(P) = z_i$ ir funkcija $f_1(P)$ pastoviai integruojama, net ir vietinių funkcijų sankirtose.

Originali Šepardo interpoliacijos forma nelaikoma visada patogia skaičiavimams. Taškų aplinka pagal originalią formą laikoma vientisa, taip pat jos apskaičiavimas pareikalauja nemažai išteklių [9, 10].

2.2. Krigingas

Krigingas arba **Gauso proceso regresija** – interpoliacijos metodas, kuriame vertės manipuliuojamos pagal Gauso procesą, pagal ankstesnių kintamųjų skirtumus. Tinkamai pasirinkus kintamuosius, tai tiksliausias metodas rasti linijinio dydžio kitimą tarp dviejų išmatuotų taškų.

Krigingo metodai geriausiai veikia su reguliariai išdėstytais duomenimis. Erdvinė koreliacija arba priklausomybė gali būti išmatuojama (semi)variogramomis. Krigingas susijęs su semivariograma, puse kvadratinio skirtumo tarp suporuotų duomenų verčių z(x) ir z(x+h) su atstumo skirtumu h, pagal kurį atskirtos matavimo vietos:

$$\mathbf{y}\,(\mathbf{h}) = \frac{1}{2}\,\mathrm{E}\left[\mathrm{z}\,(\mathrm{x}) - \mathrm{z}\,(\mathrm{x}+\mathrm{h})\right]^2$$

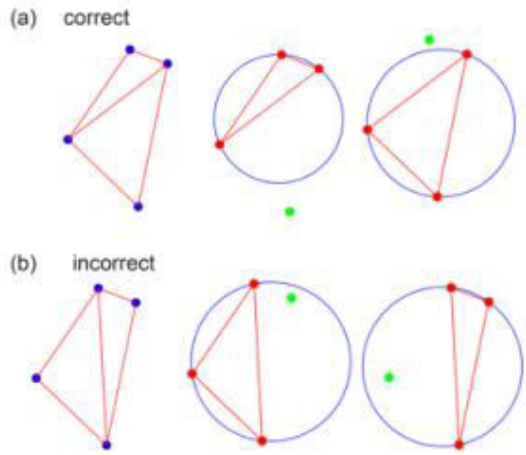
Su diskrečiomis matavimo vietomis yra aprašomas forma:

$$\gamma(\mathbf{h}) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} \left[\mathbf{z}(\mathbf{x_i}) - \mathbf{z}(\mathbf{x_i} + \mathbf{h})\right]^2$$

Kur $z(x_i)$ yra kintamojo Z vertė vietoje x_i, h yra atstumo skirtumas ir $N(h)$ yra matavimo taškų porų skaičius, atskirtas h. Nereguliariam matavimui, retai kada atstumas tarp matavimo vietų būna lygus h. Semivariogramos brėžinys gaunamas apskaičiuojant vertes su skirtingais atstumais. Šios vertės paprastai pasirenkamos pagal teoretinį modelį: apskritiminė, sferinį, eksponentinė arba Gauso. Modeliai teikia informacijos apie erdvinę struktūrą ir įvesties parametrus krigingo interpoliacijai. Yra skirtingų krigingo metodų, iš kurių paprasčiausias ir tiksliausias yra Paprastasis krigingas (*angl. Ordinary Kriging*) [11, 12].

2.3. Delaunay trianguliacija

Delaunay trianguliacija – metodas, skirtas erdvėje išdėstytams taškams sujungti į trikampius taip, kad trikampių kampai būtų kuo didesni – vengiama labai smailių trikampių. Toje pačioje tiesėje esantiems taškams Delaunay trianguliacijos nėra, nes tarp tokių taškų trianguliacija nebrėžia trikampių. Dažniausiai naudojamas trianguliacijos kriterijus – rasti trikampius, apie kurių kampus nubrėžus apskritimą į tą apskritimą nepapultų kitas taškas (7 pav.).



8 pav. Delaunay trianguliacijos pagrindinis kriterijus: a atveju trianguliacija atliekama teisingai, b atveju – klaidingai [13]

Euklidinėje erdvėje keturiems arba daugiau taškų paprastai nebūna „geriausio“ sprendimo – bet kuris iš dviejų būdų padalinti stačiakampį padalinti į du trikampius laikomas vienodai teisingu. Delaunay trianguliacijai svarbu, kad trikampio viduje nebūtų kitų geometrinių figūrų. Trianguliavimo algoritmai naudojami kompiuterinėje grafikoje, geografinės informacijos sistemose ir belaidžiuose jutiklių tinkluose [13, 14].

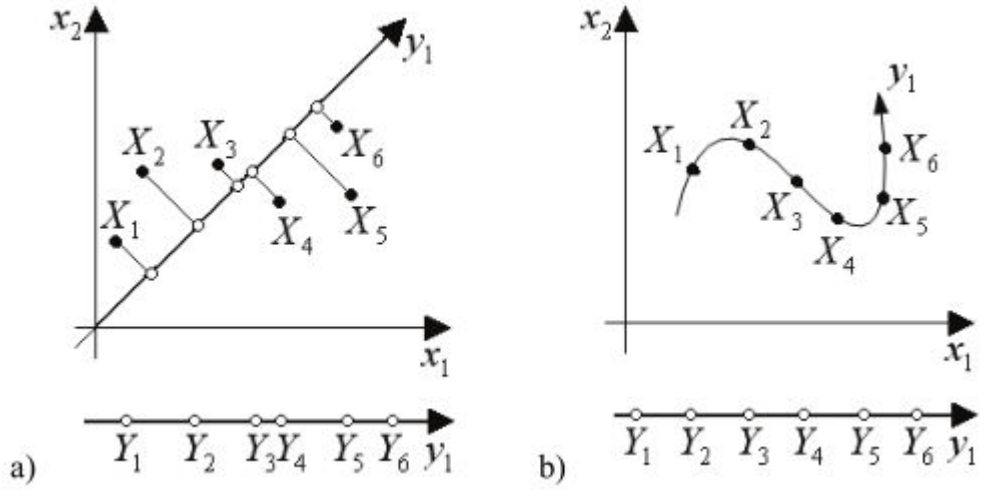
2.4. Daugiamačių duomenų vizualizavimas

Informacinei sistemai pateikiami duomenys turi didelį parametrų skaičių, t. y. duomenys yra labai didelės dimensijos, todėl labai sunku šiuos duomenis suprasti. Daugeliu atvejų neaiški, reikalaujanti gilios analizės duomenų struktūra, gali būti aprašyta mažu požymių skaičiumi. Dimensijai mažinti sukurta daugybė metodų, dar vadinamų projekcijos metodais.

Projekcijos metoduose yra naudojamas formalus matematinis kriterijus, pagal kurį kiek įmanoma sumažinamas projekcijos iškraipymas. Projekcijos metodai skirstomi į tiesinius ir netiesinius:

1. tiesinės projekcijos metodai: pagrindinių komponenčių analizė (*angl. principal component analysis*), tiesinė diskriminantinė analizė (*angl. linear discriminant analysis*), faktorinė analizė (*angl. factor analysis*), nepriklausomų komponenčių analizė (*angl. independent component analysis*), projekcijos paieška (*angl. projection pursuit*) ir kiti;
2. netiesinės projekcijos metodai: daugiamatės skalės (*angl. multidimensional scaling*) ir šio metodo atskiras atvejis – Sammono projekcija (*angl. Sammon mapping, projection*), pagrindinės kreivės (*angl. principal curves*), saviorganizuojantys neuroniniai tinklai (*angl. self organizing map, SOM*), trianguliacijos metodas (*angl. triangulation method*), lokaliai tiesinis vaizdavimas (*angl. locally linear embedding*) ir kiti.

Tiesinės projekcijos metodais ieškoma tiesinės analizuojamų duomenų transformacijos, o netiesinės projekcijos – netiesinės transformacijos. Tiesinės projekcijos metodai labai efektyvūs, kai duomenys pasiskirsto po tam tikrą poerdvį. Be to, jie reikalauja mažiau skaičiavimų negu netiesinės projekcijos metodai. Tačiau tikslesnė duomenų struktūra išlaikoma naudojant netiesinės projekcijos metodus. Deja ir šiuo atveju duomenų vizualizavimo iškraipymai yra neišvengiami.



9 pav. a – tiesinė projekcija, b – netiesinė projekcija

Plokštumos taškai X_1, X_2, ..., X_6 išdėstyti taip, kad tarp artimiausių taškų būtų vienodi atstumai, t. y $d(X_i + X_{i+1}) = d(X_{i+1} + X_{i+2})$. Jei šiuos taškus atvaizduosime į vienmatę erdvę (į tiesę y_1) naudodami tiesinę projekciją, tai atstumai tarp taškų nebus išlaikyti (9a pav.). Tačiau netiesinės projekcijos atveju, radus tinkamą transformaciją, atstumai tarp artimiausių taškų išliks vienodi (9b pav.) [15].

3. METODINĖ DALIS

Metodinėje darbo dalyje sudaromas sistemos modelis, iškeliami reikalavimai sistemos prototipui ir pagal sudarytą modelį parenkamos technologijos, programinė ir techninė įranga skirta prototipo realizacijai. Taip pat išaiškinami matematiniai niuansai, kurie bus naudojami kuriant sistemos programinį kodą.

3.1.Reikalavimai sistemai

Duomenų surinkimo sistema grindžiama belaidžių jutiklių tinklu, sudarytu iš oro sąlygas matuojančių jutiklių platformų, iš kurių ši sistema surenka duomenis, kurie per vartotojo sąsają atvaizduojami žemėlapyje kartu su platformų – plūdurų vieta žemėlapyje.

Informacinės sistemos oro sąlygų stebėjimui ir atvaizdavimui specifikuoti funkciniai ir nefunkciniai sistemos reikalavimai, kurie apibrėžia kuriamos sistemos prototipo vykdomas funkcijas ir apribojimus. Sistemos reikalavimai pateikti 1 lentelėje.

1 lentelė. Reikalavimai sistemai

Funkciniai reikalavimai
Tinkamu formatu įrašyti ir išsaugoti meteorologonius duomenis duomenų bazėje
Gaunami duomenys turi būti atvaizduojami grafiškai, kaip gradientas ir skaitmeniniu pavidalu
Informacija apie plūdurą ir jo matavimą pasiekiama pažymėjus matavimo vieta
Kokį išmatuotą parametrą vaizduoja gradientas pasirenkama iš parametrų sąrašo
Turi būti galimybė peržiūrėti, koreguoti ir ištrinti plūdurų matavimus iš sistemos administravimo aplinkos
Nefunkciniai reikalavimai
Sistema turi veikti išmaniuosiuose telefonuose, planšetiniuose kompiuteriuose, nešiojamuose kompiuteriuose, stacionariuose kompiuteriuose su interneto prieiga
Sistemos vartotojų kiekis turi būti apribotas tik serverio techninėmis galimybėmis
Sistemą galima pasiekti be papildomos programinės įrangos, tai yra iš naršyklės
Koordinačių atvaizdavimas turi būti atliktas žemėlapyje
Sistemą turi palaikyti populiariausios naršyklės

Pagal iškeltus sistemos reikalavimus, kuriuose akcentuojamas sistemos lankstumas, prieita prie išvados, kad oro sąlygų stebėjimo ir atvaizdavimo sistema turi veikti interneto tinkle ir būti pasiekiama naudojant įvairių tipų įrenginius, kurie turi interneto prieigą ir standartinę interneto naršyklę.

Siekiant įvykdyti duomenų išsaugojimą internetinėje duomenų bazėje, duomenų bazė turi būti pritaikyta geografinių duomenų saugojimui ir darbui su geografiniais duomenimis. Taip pat turi būti apibrėžtas duomenų formatas, kad duomenys nebūtų klaidingai interpretuojami ir neteisingai atvaizduojami.

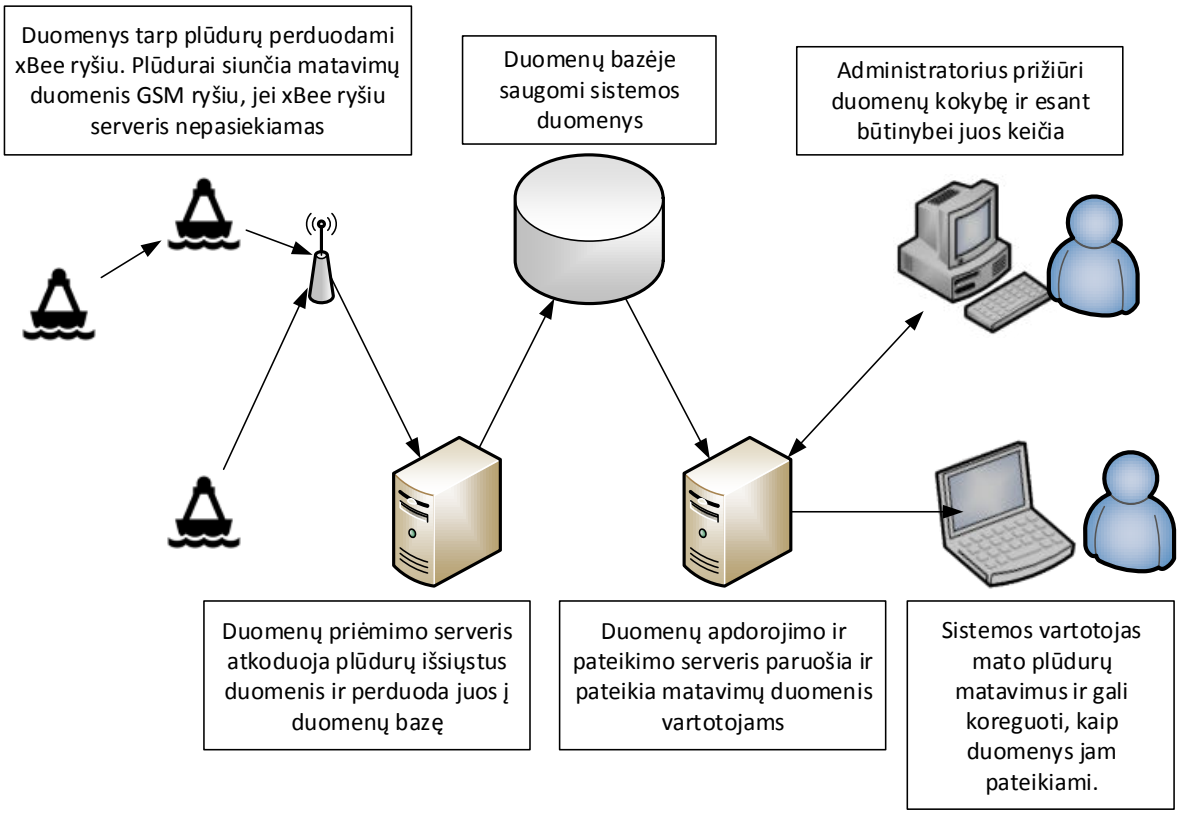
Taip pat iš pateiktų reikalavimų nustatyta, kad sistemoje duomenys turės būti atvaizduojami grafiškai, naudojant programinį, kintamo mastelio žemėlapį, o vartotojo grafinė sąsaja privalo būti

interaktyvi. Šiems reikalavimams įvykdyti reikalingos atvaizdavimo technologijos, kurios būtų suderinamos naudojamomis technologijomis.

Pagrindinė problema, kuri turi būti išspręsta sistemoje yra oro sąlygų nustatymas iš kelių matavimo taškų. Tai reiškia, kad duomenys turi būti atvaizduojami ne tik matavimo taškuose, bet ir modeliuojama, kokios sąlygos tikėtinos tarp stebėjimo taškų. Šiai problemai išspręsti būtina pritaikyti technologijas, kuriomis būtų galima atvaizduoti matavimo duomenis į žemėlapyje ir pagal matavimo duomenis jį keisti.

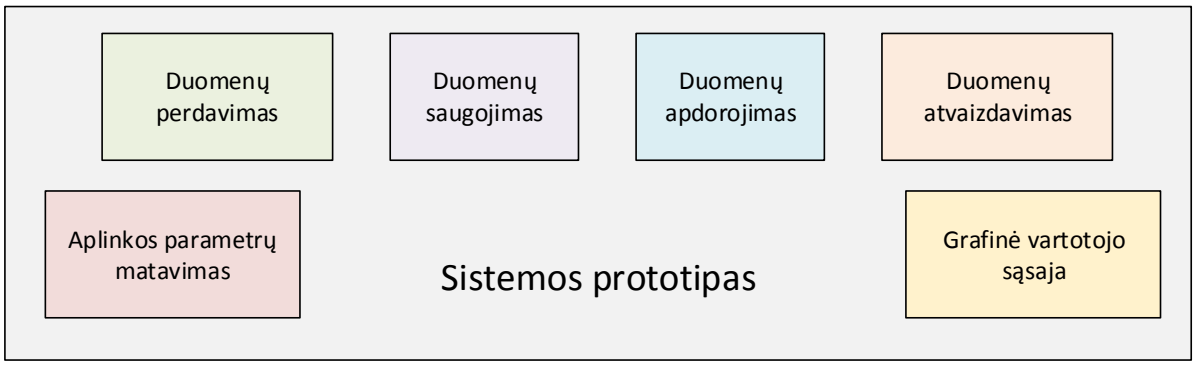
3.2. Sistemos architektūra

Daugumą duomenų asimiliavimo sistemų sudaro: duomenų šaltiniai, iš kurių gaunami duomenys; serveris, valdantis duomenų perdavimą ir apdorojimą; duomenų bazė, kurioje saugomi duomenys; vartotojų klientai, kurie peržiūri sistemos pateikiamus duomenis bei administratorių klientai, kurie gali redaguoti vartotojams rodomus duomenis (10 pav.).



10 pav. Duomenų surinkimo ir vizualizavimo sistemos struktūra

Aplinkos sąlygų stebėjimo ir analizavimo sistemos pagrindinė užduotis – surinkti aplinkos sąlygų matavimų duomenis iš plūdurų ir juos pateikti vartotojams. Pagal šį reikalavimą sudaromi sistemos dalių modeliai.



11 pav. Sistemos prototipo sudedamosios dalys

Sistemos prototipas išskaidytas į šešis funkcinius blokus. Funkcinių blokų reikšmės ir jų atliekamos funkcijos aprašytos 2 lentelėje.

2 lentelė: Sistemos prototipo elementų aprašymas

Elementas	**Aprašymas**	**Funkcijos**
Aplinkos parametrų matavimas	Belaidės, dreifuojančios jutiklių platformos - plūdurai	Intervalinis parametrų matavimas
Duomenų perdavimas	Ryšio užmezgimas tarp sistemos mazgų: plūdurų, duomenų priėmimo serverio ir duomenų bazės	Intervalinis duomenų perdavimas kitiems plūdurams arba į serverį Duomenų kodavimas ir atkodavimas
Duomenų saugojimas	Išmatuotų duomenų kaupimas duomenų bazėje vėlesniam naudojimui	Duomenų bazės pritaikytos duomenų struktūrai sukūrimas Duomenų įrašymas į duomenų bazę Duomenų pateikimas pagal užklausas
Duomenų apdorojimas	Programinė įranga paverčianti taškinius duomenis ir suteikianti jiems erdvinę reprezentaciją	Duomenų apdorojimas Daugiamačių duomenų parengimas atvaizdavimui
Duomenų atvaizdavimas	Programinė įranga, grafiškai atvaizduojanti matavimų duomenis žemėlapyje	Duomenų vizualizavimas žemėlapyje
Grafinė vartotojo sąsaja	Programinė įranga leidžianti pasirinkti, kokius duomenis matyti bei koreguoti	Sąsajos parametrų redagavimas

Remiantis duomenimis pateiktais 2 lentelėje numatyta, kad sistemai įgyvendinti bus reikalingi programiniai įrankiai, kurie leis realizuoti sistemos elementus: duomenų bazę; duomenis apdorojantį ir perduodantį serverį; duomenų atvaizdavimo įrankius ir grafinę vartotojo sąsają.

Sistemos prototipo aplinkos parametrų matavimo ir duomenų perdavimo posistemės buvo įgyvendintos ankstesniame bakalauro darbe, dėl to šiame darbe nebus realizuojamos [16].

3.3. Duomenų bazė

Dauguma interneto puslapių ir paslaugų naudoja kokią nors duomenų bazę. Tai galioja ir kartografinėms interneto paslaugoms. Svarbu pasirinkti duomenų bazę, kuri turėtų gerą geografinių ir geometrinių duomenų tipų ir užklausų palaikymą. Tokių duomenų tipų palaikymas supaprastina informacijos apie vietovę ar geometrinių figūrų, vaizduojamų ant žemėlapių, saugojimą, agregavimą ir ieškojimą tarp geometrinių įrašų.

PostgreSQL – objektinė-reliacinė duomenų bazė, atvirojo kodo (naudoja BSD licenziją). Palaiko tokius duomenų tipus:

- daugiamačius masyvus;
- geometrinius duomenis;
- JSON objektus (įskaitant ir dvinarį, vietą taupantį ir indeksuojamą JSONB);
- vartotojo sukurtus tipus.

Kiti PostgreSQL privalumai yra:

- ANSI-SQL:2008 standarto palaikymas;
- ACID (atomiškumas, vientisumas, izoliavimas ir atsparumas) suderinamumas;
- dalinių, GiST, GIN, išraiškos, pirminių ir kitų indeksų palaikymas;
- virtualios lentelės (CTE su WITH komanda);
- rekursinės užklausos;
- sąjungos užklausos;
- vartotojo sukurtos funkcijos.

Geometrinių ir geografinių duomenų palaikymui PostgreSQL duomenų bazė reikalauja PostGIS papildinio. PostGIS platinama pagal GPL atvirojo kodo licenziją. Turi daugiau kaip 300 erdvinių funkcijų, tiek 2D, tiek ir 3D, 4D duomenimis (gali saugoti Z ir M koordinates, bet dauguma erdvinių funkcijų jas ignoruoja). Kaip geometrines figūras palaiko daugiakampius, taškus, linijų sekas, taškų masyvus, daugiakampių masyvus, linijų sekų masyvus, geometrijos rinkinius, apskritimines sekas, sudėtines kreives, kreivių daugiakampius, kreivių masyvus, paviršių masyvus. Geba transformuoti koordinačių sistemas *ST_transform* funkcijos dėka. Gali išvesti geometrinius duomenis *KML, GeoJSON, SVG, GML,* tekstiniu ir dvejetainiu formatais [17, 18].

Darbui buvo pasirinkta PostgreSQL duomenų bazė, nes ji pritaikyta kitiems pasirinktiems duomenų bazės komponentams ir turi platų geometrinių duomenų palaikymą ir jos platus duomenų tipų palaikymas daro ją tinkama prototipinėms sistemoms: net jei galutiniam sistemos variantui pasirenkama kita duomenų bazės valdymo sistema, jos struktūrai reikės labai mažai pakeitimų.

3.4. Duomenų apdorojimo ir perdavimo serveris

Duomenų apdorojimui ir perdavimui turi būti naudojamas serveris, gerai suderinamas su pasirinkta duomenų baze ir gerai suderinamas su žemėlapių atvaizdavimo serveriais bei aplikacijomis. Vienas tokių serverių yra NodeJS.

NodeJS – atviro kodo, daugiaplatforminė serverio pusės ir tinklo aplikacijų veikimo aplinka (angl. *runtime environment*), kuri iš esmės Javascript bibliotekų rinkinys, veikiantis kartu su *V8*, *Javascript* interpretavimo varikliu, kuris buvo sukurtas *Google Chrome* naršyklei, paleidžiantis *JavaScript* kodą serveryje ar asmeniniame kompiuteryje, be naršyklės. Tai leidžia su *JavaScript* kurti interneto serverius ir programas.

Dauguma šiuolaikinių serverio pusės programinių karkasų naudoja sinchroninę architektūrą, *NodeJS* naudoja asinchroninę architektūrą, kuri gerai suderinama su *JavaScript* kalba. Tai reiškia, kad serveris reaguoja į įvykius, pvz. vartotojo veiksmą ir išsiunčia įvykius kaip žinutes, pvz. į duomenų bazę. Toks programavimo stilius smarkiai skiriasi nuo sinchroninio stiliaus ir jį gali būti sudėtinga naudoti su kitomis kalbomis. NodeJS naudoja asinchroninį stilių su neblokuojama asinchronine įvestimi ir išvestimi – tai privalumas, kuriuo *NodeJS* išskiria iš *PHP*, nes *PHP* yra blokuojanti kalba (įvykdo komandą tik tada, kai prieš tai buvusi komanda yra atliekama, kai tuo tarpu *NodeJS* komandas vykdo lygiagrečiai, o apie užduoties atlikimas atliekamas naudojant Atgalinis skambinimus (angl. *callbacks*), pažadus (angl. *promises*) ir asinchronines funkcijas (angl. *async functions*) [19].

Vienas iš didžiausių NodeJS privalumų: NodeJS papildinių organizavimo vedlys *(angl. NodeJS Package Manager),* arba *NPM*. Iš *NPM* archyvų galima greitai parsisiųsti reikiamas kodo bibliotekas, pradedant matematinių funkcijų bibliotekomis ir baigiant serverių karkasais. *NPM* taip pat parsiunčia ir administruoja bibliotekų ir projektų priklausinius (angl. *dependencies*). Į projektui skirtą package.json failą surašius projekto priklausinių pavadinimus ir versijų numerius ir paleidus komandą „*npm install*“ bus surašomos visos projektui reikalingos bibliotekos ir jų priklausiniai.

Vienas iš greičiausių būdų sukurti interneto serverį su *NodeJS* yra *ExpressJS* serverių karkasas [20]. Tai minimalistinis ir lankstus karkasas, kuris leidžia pasinaudoti jau realizuota metodų logika, tokia kaip:

- maršrutizavimas (angl. *routing*);
- tarpinės programinės įrangos (angl. *middleware*) naudojimas;
- šablonų (angl. *templates*) variklių naudojimas.

Duomenų apdorojimo ir perdavimo serveriui sukurti buvo pasirinkta *NodeJS* veikimo aplinka ir *ExpressJS* serverio karkasas.

3.5. Žemėlapių atvaizdavimo biblioteka

Siekiant atvaizduoti surinktus duomenis žemėlapyje, lengviausia tai padaryti yra naudoti žemėlapių atvaizdavimo biblioteką. Yra kelios tokios bibliotekos, iš kurių labiausiai ištobulinta laikoma OpenLayers.

OpenLayers – nemokamas atvirojo kodo karkasas, parašytas su *Javascript*, skirtas internetu prieinamų dinaminių žemėlapių aplikacijų kūrimui, veikiantis iš naršyklės. *OpenLayers* privalumai:

- pikselinių žemėlapių, tokių kaip OSM, Bing, MapBox, Stamen ir kt. palaikymas;
- vektorinių žemėlapių sluoksnių, gaunamų GeoJSON, TopoJSON, KML, GML, Mapbox palaikymas;
- žemėlapių piešimas su Canvas 2D, WebGL ir kitais HTML5 grafikos metodais;
- žemėlapių valdymo elementų stilizavimas su paprastu CSS.

OpenLayers pagrindinis objektas vadinamas žemėlapiu ir aprašomas kaip *ol.Map* objektas. Žemėlapis sudarytas iš grafinių sluoksnių, aprašomų *ol.Layer* objektų. Šiuose objektuose laikomi visi grafiniai žemėlapio elementai.

Žemėlapio sluoksniai gali būti įvairių tipų. Pavyzdžiui, sluoksnis ol.layer.Vector bus vektorinio tipo ir jo duomenys bus atvaizduojami ir apdorojami kliento pusėje, kaip vektorių rinkinys. Vektorinius duomenis taip pat galima pavaizduoti kaip karščio žemėlapį (*angl. Heatmap*), naudojant sluoksnį *ol.layer.Heatmap.*

Jei duomenys buvo apdoroti prieš jiems patenkant į kliento pusę, duomenys dažniausiai pateikiami sluoksnio ol.layer.Tile, kuriame žemėlapio duomenys pateikiami kaip 256 pikselių dydžio kvadratiniai paveikslėliai. Pakeitus žemėlapio mastelį, duomenys pateikiami iš naujo. Taip duomenis atvaizduoja tokios paslaugos, kaip *Google Maps*. Jei naudojami skirtingų dydžių paveikslėliai, naudojamas sluoksnis *ol.layer.Image.*

Žemėlapio sluoksnį sudaro jo pavadinimas ir duomenų šaltinis nurodomas kaip *ol.Layer* objekto *source* atributas, aprašomas kaip *ol.Source* objektas. Sluoksnio šaltiniu gali būti bet koks OpenLayers palaikomas duomenų šaltinis, įskaitant *OpenStreetMap*, *Google Maps*, *Bing Maps, Mapbox* ar panašių paslaugų teikiamus žemėlapius.

Kiekvienas duomenų šaltinio objektas turi projekcijos, nuorodos ir formato atributus. Dažniausiai naudojamos projekcijos yra Merkatoro (EPSG:3857), maždaug atitinkanti vietovės atstumą metrais iki (0,0) koordinačių ir GPS projekcija (EPSG:4326), atitinkanti atstumą dešimtainiais laipsniais iki (0,0) koordinačių. GPS projekcija daug tikslesnė kai reikia apskaičiuoti atstumą tarp dviejų ar daugiau taškų ir neiškraipo žemėlapio kraštų, tuo tarpu Merkatoro projekcija naudinga, kai reikia pateikti žemėlapį kaip plokščią pasaulio reprezentaciją.

Iš šaltinio nuorodos paimami duomenys, kurie gali būti įvairių formatų. Formatas nurodomas formato atribute. Pavyzdžiui, jei naudojamas GeoJSON formatas, atributas nurodomas kaip *ol.format.GeoJSON()*.

Visi žemėlapio šaltinių formatai pateikia žemėlapių duomenis kaip reiškinių masyvą, kuriuos *OpenLayers* interpretuoja kaip objektus *ol.Feature*, turinčius atributus *id, geometry* ir *properties*. *Geometry* turi objekto tipą ir geografines koordinates, atvaizduojamas žemėlapyje, o *properties* gali turėti papildomus duomenis, nurodytus vartotojo.

Kaip perduoti duomenys atrodys žemėlapyje aprašoma per sluoksnio stiliaus atributą. Kuris gali būti Javascript objektas su nurodytais duomenų tipais arba stiliaus funkcija, kurioje galima keisti stiliaus atributus priklausomai nuo kitų kintamųjų. Žemėlapio geometrijos tipai, su kuriais veikia *OpenLayers* stiliai ir stiliaus funkcijos yra:

- Taškas (*angl. Point*);
- Linija (*angl. Line*);
- Linijų masyvas (*angl. LineString*);
- Linijų žiedas (*angl. LinearRing*);
- Daugiakampis (*angl. Polygon*);
- Keli taškai (*angl. MultiPoint*);
- Kelių taškų masyvas (*angl. MultiLineString*);
- Daugiakampių masyvas (*angl. MultiPolygon*);
- Geometrijos rinkinys (*angl. GeometryCollection*);
- Apskritimas (*angl. Circle*).

Žemėlapio valdymo elementai valdomi specialių valdiklių (*angl. widgets*) valdymo elementų *ol.Control* ir sluoksnio *ol.Overlay*, kurie leidžia kurti interaktyvius žemėlapius.

Objektai *ol.Control* turi fiksuotą vietą ekrane ir yra įdedami į konteinerį, kurio stilius koreguojamas su CSS. Jie gali reaguoti į darbo su žemėlapiu veiksmus ir gaudyti su žemėlapiu susijusius *Javascript* įvykius (*angl. events*) arba būti tik informaciniai. Galima sukurti ir savo valdymo elementus, plečiant *ol.control.Control* klasę.

Interneto puslapio elementas ol.Overlay yra rodomas virš žemėlapio ir yra susietas su tam tikromis žemėlapio koordinatėmis, todėl tempiant žemėlapį į šoną, *ol.Overlay* objektas judėti, o *ol.Control* objektas – ne [21].

OpenLayers biblioteka buvo pasirinkta dėl išsamios dokumentacijos, plataus funkcijų rinkinio ir galimybės plėsti bibliotekos funkcijas savo sukurtomis valdymo ir stiliaus funkcijomis.

3.6. Duomenų perdavimo formatas

GeoJSON – vienas iš *OpenLayers* bibliotekos ir *PostgreSQL* duomenų bazės palaikomų duomenų formatų. *GeoJSON* yra populiaraus *JSON* formato plėtinys, turintis daugiau taisyklių ir skirtas geografinių duomenų apsikeitimui.

JSON – *JavaScript* objektų notacija, yra lengvas, tekstinis, nuo programavimo kalbos nepriklausomas duomenų apsikeitimo formatas. Jis buvo sukurtas pagal *ECMAScript* programavimo kalbos standartą. *JSON* turi nedidelį rinkinį formatavimo taisyklių nuo platformos nepriklausomai, struktūrizuotų duomenų reprezentacijai. *JSON* formatavimo taisykles apibrėžia *RFC 7159* standartas, pagal kurį JSON objektai sudaryti iš atributų ir jų verčių porų (*angl. key value pairs*). Atributai išskiriami kabutėmis, tarp atributo ir jo vertės turi būti dvitaškis, o tarp atributų dedamas kablelis. Po paskutinio objekte atributo arba po paskutinio masyvo elemento, kablelio dėti negalima. *JSON* notacijoje atributų vertėmis gali būti skaičiai, simbolių eilutės (išskiriamos kabutėmis), masyvai (gali saugoti skirtingų tipų kintamuosius, išskiriami laužtiniais skliaustais), objektai (išskiriami figūriniais skliaustais) ir specialios reikšmės true, false ir null, kurios rašomos tik iš mažųjų raidžių ir be kabučių [22].

GeoJSON – geografinių erdvinių duomenų apsikeitimo formatas, pagrįstas *JSON*. *GeoJSON* apibrėžtas RFC 7946 standarto. Standartas apibrėžia kelias skirtingas *JSON* objektų formas ir kaip jie sujungiami duomenims apie geografinius objektus, jų savybes ir erdvės ribas. *GeoJSON* naudoja geografinę koordinačių sistemą, *World Geodetic System 1984*, ir dešimtainių laipsnių vienetus.

GeoJSON objektas gali apibūtinti erdvės regioną (*Geometry*), erdvėje esančią esybę (*Feature*), arba esybių rinkinį (*FeatureCollection*), kuri sudaryta kaip esybių masyvas. GeoJSON palaiko visus *OpenLayers* palaikomus geometrinių objektų tipus, išskyrus apskritimą (*Circle)*. Erdvinės esybės sudarytos iš geometrijos objekto ir papildomų parametrų (*Properties*). Papildomi parametrai gali būti įvairių tipų validūs *JSON* objektai. GeoJSON esybių rinkinys pateikiamas taip:

```
{
        "type": "FeatureCollection",
        "features": [{
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [102.0, 0.5]
            },
            "properties": {
                "prop0": "value0"
            }
        }, {
            "type": "Feature",
            "geometry": {
                "type": "LineString",
                "coordinates": [
```

```
                    [102.0, 0.0],
                    ...
                ]
            },
            "properties": {
                "prop0": "value0",
                "prop1": 0.0
            }
        }, {
            "type": "Feature",
            "geometry": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [100.0, 0.0],
                        ...
                    ]
                ]
            },
            "properties": {
                "prop0": "value0",
                "prop1": {
                    "this": "that"
                }
            }
        }]
    }
```

GeoJSON formatas plačiai naudojamas JavaScript interneto bibliotekose, *JSON* pagrįstose dokumentų duomenų bazėse ir internetinėse aplikacijų programavimo sąsajose (*angl. Application Programming Interface, API*) [23].

3.7. Šablonų variklis ir teksto ir duomenų atvaizdavimui

Sistemose, kuriamose su Node.js serveriu, kuriose interneto puslapių turinys dinamiškai keičiasi, plačiai naudojami šablonų varikliai. Kuriamame sistemos prototipe bus galima peržiūrėti plūduro matavimo duomenis nuspaudus ant geometrinės plūduro buvimo vietos. Siekiant pakeisti rodomus duomenis neperkraunant puslapio, buvo pasirinktas VueJS.

VueJS – progresyvus karkasas (*angl. progressive framework*) skirtas greitam vartotojo sąsajos kūrimui, parašytas su *JavaScript* ir išleistas su MIT licenzija. Ne taip kaip kiti, monolitiniai karkasai, *VueJS* sukurtas taip, kad galėtų būti taikomas inkrementiškai. Bibliotekos branduolys dirba tik su peržiūros sluoksniu ir lengvai integruojamas su kitomis bibliotekomis ar karkasais. Taip pat Vue gali būti naudojama vieno puslapio aplikacijomis kartu su papildomomis bibliotekomis [24].

VueJS turi daug panašumų su *ReactJS* karkasu, abu karkasai naudoja virtualius *DOM* elementus, naudoja reaktyvius ir sudėtinius peržiūros sluoksnio objektus. Atvaizduojant vartotojo

sąsają naršyklėje, daugiausiai sistemos resursų ir laiko išeikvojama *DOM* manipuliavimui. Siekiant sumažinti aplikacijos vartotojo sąsajos paleidimo laiką, abu lyginami karkasai mažina reikalingų *DOM* mutacijų kiekį ir bet *VueJS* prideda mažiau papildomų *JavaScript* operacijų negu *ReactJS*.

ReactJS taip pat bando iš naujo sukurti HTML ir CSS, nes ReactJS aplikacijos rašomos JSX, tam tikra XML notacija, naudojama JavaScript failų viduje. Tuo tarpu VueJS naudoja HTML ir CSS, bet prideda šabloną kaip HTML elementą ir elementų atributus, prasidedančiu „v-„ priesaga.

Naudojant VueJS, atributu „v-if“ aprašomi sąlyginiai sakiniai, o „v-for“ atributais aprašomi ciklai. Kintamieji iš elementų išskiriami dvigubais figūriniais skliaustais. Naudojant VueJS, HTML kodas atrodo taip:

```
<template>
  <div class="list-container">
    <ul v-if="items.length">
      <li v-for="item in items" :key="item.id">
        {{ item.name }}
      </li>
    </ul>
    <p v-else>No items found.</p>
  </div>
</template>
```

Naudojamiems kintamiesiems reikšmės priskiriamos kaip Vue objekto dalis ir aprašomos JavaScript faile. Kintamojo reikšmei pasikeitus, VueJS automatiškai pakeičia su juo susijusius DOM elementus. Pvz. jei objektų masyvas atvaizduojamas kaip lentelė, pridėjus prie masyvo naują objektą, lentelėje atsiranda eilutė su masyvo duomenimis [25].

3.8.Kūrimo aplinka ir darbe naudojami įrankiai

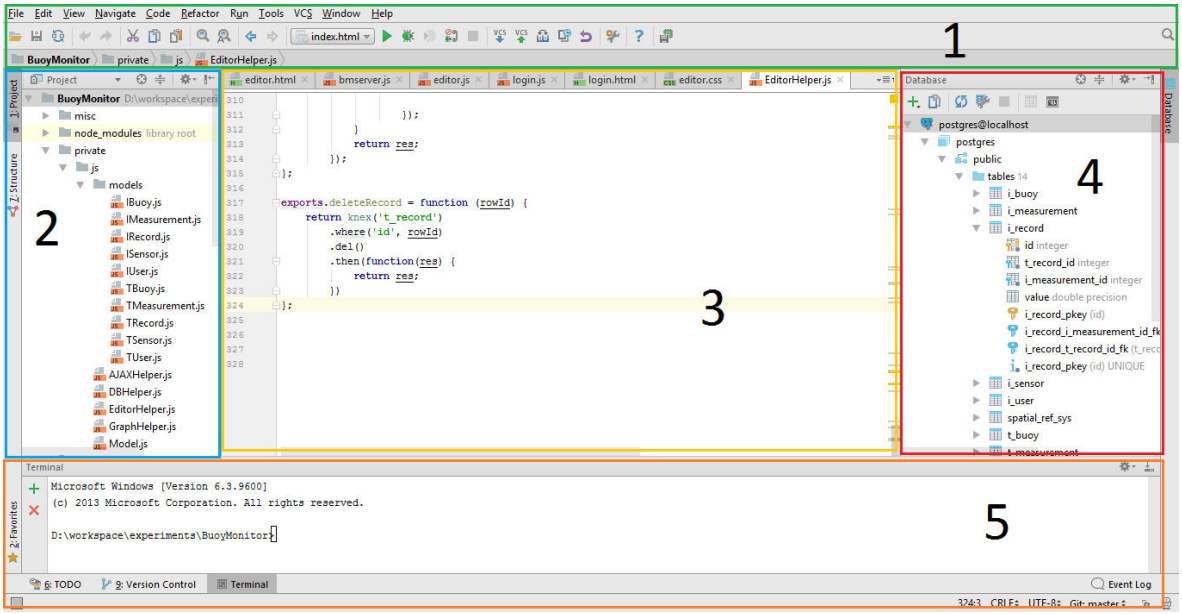
Pasirinkus reikiamą duomenų bazę, serverį ir naudojamas bibliotekas, reikia pasirinkti su kokiais programavimo įrankiais, tokiais kaip integruota kūrimo aplinka (*angl. Integrated Development Environment, IDE*), reikės dirbti.

PHPStorm – komercinė interaktyvi kūrimo aplinka, skirta *PHP, HTML, CSS* ir *JavaScript* kalboms, sukurta *JetBrains* kompanijos, IntelliJ aplikos pagrindu. PHPStorm siūlo tokias funkcijas kaip:

- kodo užbaigimas, įskaitant vartotojo aprašytas funkcijas;
- klaidų taisymas ir vienetų testų palaikymas;
- automatiškas kodo stiliaus taikymas;
- vietinio ir nuotolinių terminalų pasiekiamumas, įskaitant SSH sesijas;
- *FTP* ir *SFTP* serverių pasiekiamumas iš *IDE,* įskaitant automatinį failų įkėlimą;
- SQL sintaksės klaidų taisymas;

- duomenų bazių duomenų peržiūrėjimas ir redagavimas;
- versijavimo sistemų integracijos, palaikančios *Git*, *Mercurial, SVN* ir *Perforce* versijavimo sistemas.

PHPStorm darbo aplinka pateikta 12 pav:



12 pav. PHPStorm Interaktyvioji kūrimo aplinka

Kaip ir dauguma kitų interaktyviųjų kūrimo aplinkų, *PHPStorm* turi kelias sritis, atliekančias tam tikras funkcijas:

1. *Įrankinės*, kuriose pasiekiamos failų manipuliacijos, programos paleidimo ir versijavimo sistemų sąsajos.
2. *Hierarchinė projekto struktūra*, kurioje atvaizduojama programos naudojamų modulių, direktorijų ir failų struktūra. Taip pat atskiroje kortelėje pateikiama ir programos struktūra.
3. *Pagrindinis darbo langas*, kuriame atvaizduojamas ir rašomas programinis kodas. Taip pat jame pasirenkama tarp atidarytų failų, kuriuos galima sukurti, įkelti ir atidaryti failų struktūroje.
4. *Nuotolinės sąsajos peržiūros langas*, atsirandantis susiejus savo projektą su duomenų baze arba nuotoliniu serveriu. Iš jo pasiekiamos galimybės kurti ir modifikuoti duomenų bazės lenteles ir įrašus, o serverio atveju – galima su serveriu keistis projekto failais.
5. *Terminalo langas*, iš kurio galima siųsti terminalo komandas į vietines ir nuotolines sistemas, versijavimo sistemas, tokias kaip Git arba Mercurial. Prie nuotolinių sistemų galima prisijungti ir naudojant SSH (*angl. secure shell*) sesijas, leidžiančias manipuliuoti nuotolinius serverius.

Įdiegus *PHPStorm* ir kitą reikiamą programinę įrangą bei suderinus ją su bandymams naudojamu nuotoliniu arba vietiniu serveriu ir duomenų baze, galima kurti internetines aplikacijas skirtas PHP, *NodeJS* ir kitų technologijų interneto serveriams [26].

Funkcija, kurios trūksta *PHPStorm* – automatinis duomenų bazių diagramų braižymas. Šiai funkcijai buvo pasitelktas kitas JetBrains produktas *DataGrip*, skirtas tik darbui su duomenų bazėmis.

3.9. Techninė serverio įranga

Pasirinkus programinę įrangą, reikia techninės įrangos, kuri suderinama su visa pasirinkta programine įranga. *PostgreSQL* duomenų bazė veikia su *BSD, Linux, Solaris, macOS* ir *Windows* operacinėmis sistemomis ir jų palaikomomis architektūromis. *NodeJS* ir *PHPStorm* suderinamas su Linux, macOS ir Windows operacinėmis sistemomis.

Duomenų bazė ir duomenų surinkimo ir apdorojimo serveris kūrimo ir bandymų metu veikė ant asmeninio kompiuterio, kurio specifikacija yra:

- procesorius: Intel i5-4210U, 1.7-2.7 GHz, 64 bitų architektūra;
- operatyvioji atmintis: 8 GB;
- kietasis diskas: 500 GB SSD;
- operacinė sistema: Windows 8.1.

Tokių techninių parametrų sistemos prototipui užtenka, tačiau jie riboja galimų prisijungimų prie sistemos skaičių. Realizavus sistemos prototipą, bus naudojamas galingesnis serveris, kuris gerokai padidins galimų vartotojų skaičių lyginant su prototipu.

4. PROTOTIPO ĮGYVENDINIMAS

Išanalizavus metodus, kurie skirti sistemos realizacijai bei programinius įrankius ir technologijas, pradedamas sistemos praktinis realizavimas. Prieš pradedant programuoti būtina įdiegti reikalingą programinę įrangą ir jos priedus. Įdiegta įranga pateikta 3 lentelėje.

3 lentelė. Reikalinga programinė įranga sistemos realizacijai

Programinė įranga	Aprašymas	Skirta
PostgreSQL	Atvirojo kodo objektinė-reliacinė duomenų bazė	Plūdurų jutiklių duomenų įrašų talpinimui ir saugojimui
PostGIS	PostgreSQL papildinys darbui su geometriniais duomenimis	Plūdurų koordinačių saugojimui geometriniu formatu
PHPStorm	Programinio kodo redaguotojas su įvairių programinių kalbų ir technologijų palaikymu	Programinė įranga, kuria rašomas programinis kodas ir atliekamas jo testavimas ir versijų saugojimas
NodeJS	Serverio pusės veikimo aplinka	Serverio pusės programavimo aplinka
ExpressJS	*NodeJS* karkasas	*NodeJS* aplikacijos kūrimo proceso spartinimui
OpenLayers	Žemėlapių valdymo biblioteka	Duomenų vizualizavimui žemėlapyje
VueJS	Reaktyvių interneto puslapių šablonų variklis	Tekstinių duomenų atvaizdavimui ir keitimui priklausomai nuo vartotojo įvesties

Serverio pusės kodas dėl *NodeJS*, *ExpressJS* galimybių ir architektūros realizuojamas *JavaScript* kalba. Kliento pusėje vykdomi scenarijai (duomenų apdorojimas, perdavimas ir vizualizavimas), pasinaudojant tokiomis bibliotekomis, kaip *OpenLayers ir VueJS,* kurios taip pat programuojamos *JavaScript* kalba. Klientų puslapių struktūrai realizuoti taip pat reikės pasinaudoti *HTML*, *CSS* žymėjimo kalbomis ir *VueJS* elementais.

Įgyvendinant įvairias sistemos funkcijas, panaudoti įvairūs NodeJS moduliai, kuriuos galima parsisiųsti *Node* paketų valdymo sistemos dėka (*NPM, angl. Node Package Manager*). Visų panaudotų modulių sąrašas pateiktas 4 lentelėje:

4 lentelė. Panaudoti NPM moduliai

Programinė įranga	Aprašymas	Skirta
axios	AJAX duomenų siuntimo biblioteka	Dalintis duomenimis tarp programos serverio ir klientų per asinchronines užklausas
bcrypt	Šifravimo biblioteka	Šifruoti vartotojų slaptažodžiams
body-parser	Per AJAX gautų duomenų interpretavimo modulis	Perskaityti šifruoto URL ir JSON tipo duomenims, kurie gaunami iš redagavimo aplinkos formų
client-sessions	Šifruotų slapukų (angl. cookies) kūrimo ir nuskaitymo modulis	Saugoti prie redagavimo aplinkos prisijungusio vartotojo sesijos slapukui
knex	Duomenų bazės užklausas rašyti padedantis modulis	PostgreSQL užklausų automatiniam generavimui
nodemon	Biblioteka, iš naujo paleidžianti duomenis, kai programuotojas padaro kodo pakeitimus.	Greitesniam programos testavimui ir klaidų ieškojimui
pg	Biblioteka, skirta NodeJS programai susieti su PostgreSQL duomenų bazei	Serverio ir duomenų bazės komunikacijai

Projektuojant NodeJS aplikacijas, dažnai susiduriama su sunkumais, kai reikia pasirinkti reikiamus NPM modulius, tačiau tinkamas modulių naudojimas pagreitina sistemos kūrimo procesą.

4.1. Duomenų bazės kūrimas

Duomenų bazėje saugomi sistemos surinkti duomenys, kurie naudojami duomenų vizualizavimui. Iš pradžių planuota naudoti JSON objektų duomenų bazę, tačiau tokias duomenų bazes sudėtinga redaguoti. Tradicinę reliacinę duomenų bazę pavyko realizuoti kuriant po dvi duomenų bazės lenteles kiekvienam techniniam sistemos elementui:

- tipų lentelės, kuriose saugomi elementų tipai, jų pavadinimai prasideda *„t_“;*
- įrašų (instancijų) lentelės, kuriose saugomi konkretūs elementai, jų pavadinimai prasideda *„i_“*.

Pavyzdžiui, jei turime kelių tipų plūdurus, su skirtingo tipo jutikliais arba skirtingu jų kiekiu, kiekvienas skirtingas plūduro variantas turės savo tipą – tipo lentelės *„t_buoy“* įrašą, su modelio pavadinimu. Tuo tarpu kiekvienas individualus plūduras turės savo instancijos lentelės *„i_buoy“* įrašą, su tipo numeriu, vardu, spalva ir statusu.

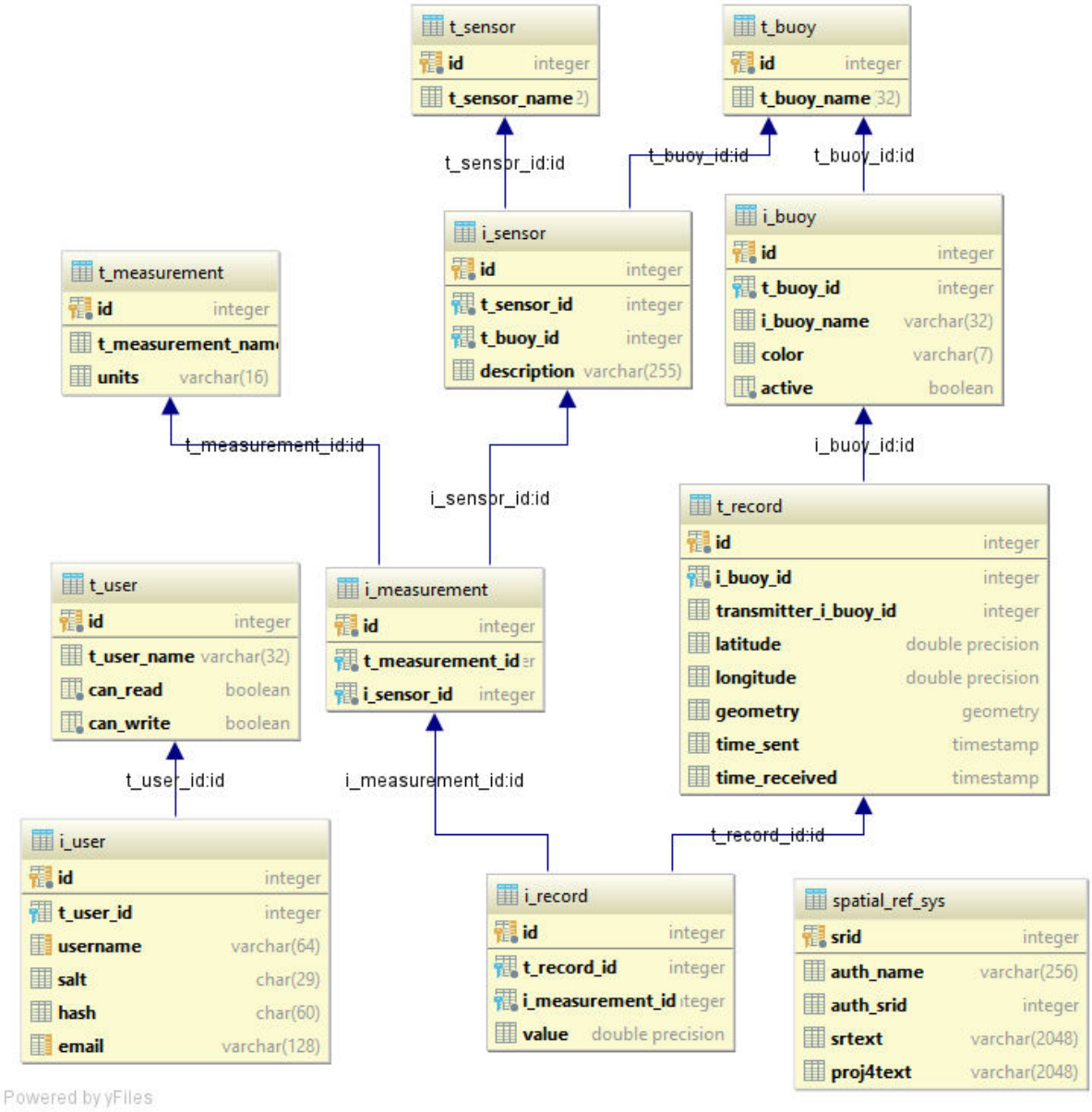
Tam tikro tipo plūdurai turės tam tikro jutiklius, aprašytus lentelėse *„t_sensor“* ir „*i_sensor“*. Jutiklių tipo lentelėje saugomas jutiklio tipo pavadinimas, o instancijos lentelėje saugoma, kokio tipo plūdurai turi to tipo jutiklį. Kelis to tipo jutiklius turintis plūduro tipas turės kelis jutiklio instancijos įrašus.

Kai kurie jutikliai gali matuoti daugiau nei vieną parametrą, dėl to jutiklių matuojami parametrai saugomi atskirose lentelėse. Matuojamo parametro tipo lentelė *„t_measurement“* aprašo parametro pavadinimą ir matavimo vienetus. Matuojamo parametro instancijos lentelė *„i_measurement“* susieja matavimo tipą su jutiklio instancija.

Įrašų, kuriuose saugomi plūdurų jutiklių matavimai, lentelėje *„t_record“* saugomi atskiri įrašai, koks plūduras atliko matavimus ir iš kokio plūduro duomenys perduoti, jei duomenys perduoti per plūdurų belaidį tinklą, plūduro geografinės koordinatės duomenų perdavimo metu ir perdavimo bei gavimo laikas. Geografinės koordinatės sistemos prototipe saugomos kaip atskiri lentelės įrašai ir kaip geometrijos objektas. Galutiniame sistemos modelyje duomenys bus saugomi tik geometriniame objekte. Tuo tarpu įrašo instancijos lentelėje *„i_record“* saugomi atskiri jutiklių matavimai ir matavimų vertės.

Vartotojo tipų lentelė *„t_user“* atstoja vartotojo vaidmenų (*angl. user roles*) sistemą, dažnai naudojamą turinio valdymo sistemose. Vartotojo tipų lentelėje saugomas tipo pavadinimas ir ką gali daryti vartotojas: ar gali skaityti apriboto pasiekiamumo puslapius ir ar gali kurti naujus įrašus.

Duomenų bazės struktūros diagrama, sukurta su DataGrip duomenų bazių valdymo sistema, pateikta 13 pav.:



13 pav. Duomenų bazės diagrama.

PostgreSQL duomenų bazės papildinys *PostGIS* sukuria savo lentelę „spatial_ref_sys“, naudojamą kaip atskaitos taškų saugyklą erdvinei atskaitos sistemai (*angl. spatial reference system*), naudojamai atliekant geografinės paieškos ir kitas funkcijas.

4.2. Serverio programinės įrangos kūrimas

Naudojant *ExpressJS* serverį, jį aprašančiame *JavaScript* faile inicializuojamos visos serveryje naudojamos paslaugos ir importuojami visi sistemos naudojami moduliai, bei užregistruojamos vartotojui pasiekiamos direktorijos. Failai importuojami naudojant NodeJS komandą *„require“*. Pavyzdžiui, serverio biblioteka importuojama ir paleidžiama šitaip:

```
var express = require('express');
var app = express();
```

Vartotojo klientas serverio failus gali pasiekti, kai įvesto puslapio *URL* atitinka tam tikrą simbolių seką ir užklausa vykdoma kaip *GET* metodas, pasiekiamas *ExpressJS* funkcija *„get()“* arba *POST* metodas, pasiekiamas *ExpressJS* funkcija *„post()“*. Pavyzdžiui, suvedus svetainės adresą ir užbaigus jį pasvirusiu brūkšniu, galima aktyvuoti taip aprašytą funkciją, jei ji gaunama *GET* metodu:

```
app.get('/', function(req, res) {
    res.sendFile(ABSOLUTE_PATH + 'public/html/index.html');
});
```

ExpressJS funkcijos *„get()“* ir *„post()“* kaip antrą argumentą priima atsakomąją funkciją (*angl. callback function*), su užklausos (*angl. request*) ir atsako (*angl. response*) argumentais, dažnai trumpinamais kaip „*req“* ir „*res“*. Atsako argumentas pateikiamas užklausą teikusiam klientui. Funkcija *„sendFile()“* nurodoma, kokį failą nusiųsti vartotojui. Naudojamoje sistemos versijoje funkcija reikalauja pilno kelio kompiuterio failų sistemoje.

GET metodu taip pat galima į *URL* užkoduoti parametrus, bet galima juos atskirti pasirinktais simboliais. Parametro pavadinimas atskiriamas su dvitaškiu. Toliau pateikiamamas kodo fragmentas, kuriame kaip argumentas pateikiamas norimos iškviesti *AJAX* funkcijos pavadinimas:

```
app.get('/ajax/:fName/', function(req, res) {
    var fName = req.params.fName.toString();
    var fn = function () {};

    switch(fName) {
        case 'ajaxGetBuoyData':
            fn = DBHelper.buildBuoyFeaturesJSON();
            break;
        case 'ajaxGetGraphData':
            fn = GraphHelper.buildData();
            break;
        default:
            return 'error: unknown AJAX function!';
    }
    fn.then(function (results) {
        res.send(JSON.stringify(results))
    });
});
```

Toks metodas patogus, kai siunčiamas tik vienas failas. Kai *HTML* puslapis naudoja *CSS* ir *JavaScript* failus, patogiau nurodyti statinę direktoriją, kurioje tokio tipo failai saugomi. Statinė direktorija nurodoma su funkcija *„express.static()“*. Pirmoji funkcija užregistruoja aplanką tam, kad jį galėtų naudoti sistema, o antroji – kad aplankas būtų pasiekiamas vartotojams:

```
app.use(express.static('public'));
app.use('/public', express.static('public'));
```

Taip nurodžius direktoriją, *„/public/failo_pavadinimas“* galima nurodyti kaip *HTML* elemento *„href“* atributo reikšmę.

4.3. Žemėlapio realizavimas su OpenLayers

Internetinėse aplikacijose, žemėlapiai dažniausiai pateikiami kaip kvadratinių paveikslėlių rinkinys, kur paveikslėliai asinchroniškai užkraunami pakeitus peržiūrimą vietą arba mastelį. Tai reikalauja asinchroninio duomenų perdavimo, kad žemėlapio užkrovimas nereikalautų per daug laiko ar puslapio perkrovimų.

Dėl šios priežasties, OpenLayers biblioteka gerai pritaikyta gauti žemėlapio duomenis asinchroniškai, tereikia sluoksnio objektui pateikti nuorodą į šaltinį, tačiau reikia aprašyti kokio tipo, formato ir projekcijos duomenis sistema turėtų priimti.

Žemėlapis su matavimų duomenimis bus atvaizduojamas sistemos pagrindiniame puslapyje. Jame patalpinamas HTML elementas, kuris vėliau susiejamas su OpenLayers žemėlapio objektu. Šio elemento viduje atvaizduojami visi OpenLayers žemėlapio sluoksniai ir visi jo interaktyvaus sluoksnio elementai.

OpenLayers papildinys įdiegiamas į aplikaciją įkėlus jo CSS ir Javascript failus ir serverio dėka paduodant jos vartotojui. Žemėlapio objektas inicializuojamas taip aprašius žemėlapį:

```
var map = new ol.Map({
    target: 'map',
    layers: [osmLayer, triangulationLayer, buoyLayer],
    view: view,
    controls: ol.control.defaults().extend([
        new ol.control.ScaleLine()
    ])
});
```

Kur *„target“* atribute nurodomas naudojamo *HTML* elemeneto *„id“* atributas, *„layers“* atribute saugomas žemėlapio sluoksnių masyvas, *„view“* nurodomas matomo žemėlapio centro koordinatės ir mastelis, o *„controls“* nurodomi žemėlapio valdymo elementai.

Žemėlapio sluoksniai aprašomi atskirai, kaip *„layer“* tipo objektai. Svarbiausias sluoksnio atributas – jo šaltinis. Pavyzdžiui, kaip šaltinį pateikti *OpenStreetMap* žemėlapius galima sluoksnį aprašius šitaip:

```
var osmLayer = new ol.layer.Tile({
    source: new ol.source.OSM()
});
```

Vartotojo sukurtiems šaltiniams panaudoti reikia nurodyti daugiau atributų: projekciją, nuorodą į duomenis ir formatą. Taip pat sluoksniui nurodomas stilius, pagal kurį nuspalvinami geometriniai elementai. Pavyzdžiui, dabartinė plūdurų buvimo vieta ir jų matmenys perteikiami per plūdurų sluoksnį *„buoyLayer“,* kuriam panaudoti reikalingas toks sluoksnio objektas:

```
var buoyLayer = new ol.layer.Vector({
    source: new ol.source.Vector({
        projection: 'EPSG:3857',
        url: URL + 'ajax/ajaxGetBuoyData/',
        format: new ol.format.GeoJSON()
```

```
    }),
    style: buoyStyleFunction
});
```

Stilius pateikiamas kaip stiliaus funkcija, kurioje geometrinio objekto išvaizda gali būti pakeista priklausomai objekto tipo ir nuo objekto ypatybių.

4.4. Duomenų apdorojimas

Duomenų bazėje duomenys saugomi kaip eilutės įvairiose lentelėse, o *OpenLayers* turi gauti *GeoJSON* objektų masyvą. Tam, kad duomenys būtų teisingai interpretuojami ir atitiktų *GeoJSON* standartą, reikia pakeisti duomenų formatą ir gradinetų kūrimo atveju juos apdoroti.

Jei reikia pateikti duomenis vartotojo naršyklei, atliekama tokia veiksmų seka:

1. Suformuluojama duomenų bazės užklausa.
2. Užklausa pateikiama duomenų bazei.
3. Gaunamas duomenų bazės atsakymas.
4. Atsakymo duomenys suformatuojami *GeoJSON* formatu.
5. Gradiento kūrimo atveju, atliekami papildomi veiksmai su duomenimis.
6. Duomenys pateikiami vartotojo naršyklei.

Duomenų bazės užklausų formulavimui ir vykdymui naudojamas *KnexJS* modulis. Prieš pradedant darbus su KnexJS, reikia importuoti jo modulį ir nurodyti duomenų bazės nustatymus. *PostgreSQL* atveju bus naudojamas klientas *„pg“, „searchPath“* nurodo kuriose duomenų bazės vietose *KnexJS* ieškos lentelių, o *„connection“* bus nurodomas kaip prisijungimo eilutė, kurioje suvedami prisijungimo duomenys:

```
var knex = require('knex')({
    client: 'pg',
    connection: 'postgres://vartotojas:slaptažodis@nuoroda:5432/duombazė'
    searchPath: 'knex,public'
});
```

Knex palaiko kelis skirtingus duomenų bazių klientus, kuriais galima prisijungti prie įvairių duomenų bazių, įskaitant *MySQL, SQLite*, *Oracle* ir *MSSQL*. Dėl šios priežasties, šiuo muduliu suformuotos užklausos nepriklauso nuo konkretaus SQL dialekto ir nereikalauja daug pakeitimų, jei pakeičiama naudojama duomenų bazė.

Duomenų bazės užklausa, iš kurios gaunami įrašų ir plūdurų duomenys *KnexJS* funkcijomis aprašoma taip:

```
var query = knex.select
(
    't_record.id AS id',
    knex.raw('ST_AsGeoJSON(t_record.geometry) :: JSON AS geometry'),
    'i_buoy.id AS buoy_id',
    't_record.transmitter_i_buoy_id AS transmitter_id',
```

```
    'i_buoy.i_buoy_name AS buoy_name',
    ...
)
    .from('t_record')
    .leftJoin('i_buoy', 'i_buoy.id', 't_record.i_buoy_id')
    .leftJoin('t_buoy', 'i_buoy.t_buoy_id', 't_buoy.id')
```

KnexJS yra JavaScript pažadais pagrįsta aplikacijų programavimo sąsaja (*angl. promise-based API*). Užklausos duomenų bazei pateikiamos asinchroniškai, todėl vykdomos su pažadų sprendimo funkcija *„then()“*, kurios viduje turi būti kodas, atliekamas gavus atsakymą iš duomenų bazės. Šiuo atveju sprendimo funkcijoje iš užklausos gauti rezultatai sudedami į *JSON* objektą:

```
return query
    .then(function (response) {
        var geometry_array = [];
        var rows = response;
        var keys = [];

        for (var key = 0; key <= rows.length - 1; key++) {
            var geometry =
                {
                    "id": rows[key].id,
                    "type": "Feature",
                    "geometry": rows[key].geometry,
                    "properties": {
                        "buoy_id": rows[key].buoy_id,
                        "buoy_name": rows[key].buoy_name,
                        ...
                        "sensors": []
                    }
                };
            keys.push(rows[key].id);
            geometry_array.push(geometry);
        }

        return getMeasurementsAndSensors(keys, geometry_array);
    }).then(function (geometry_array) {
        var feature_collection = GEOGRAPHY_METADATA;
        feature_collection.features = geometry_array;
        return feature_collection;
    });
```

Pirmoji „then()“ funkcija gražina papildomos funkcijos rezultatą, kuri paduoda papildomą užklausą, kuria gaunama informacija apie jutiklius bei plūdurų jutiklių matavimų duomenys, kurie kiekvienam masyvo objektui pridedami kaip atributo *„sensors“* masyvas. Gavus atsakymą, įvykdyma antroji „then()“ funkcija, naudojanti pirmosios rezultatą kaip atsakymo funkcijos argumentą. Taip jungiant pažado funkcijas, pasiekiamas funkcijų sinchroniškumas: antroji funkcija vykdoma tik prieš tai įvykdžius pirmąją. Antrojoje funkcijoje užklausų rezultatai įvelkami į *GeoJSON* objektą, kuris vėliau perduodamas vartotojui. Konstantoje *„GEOGRAPHY_METADATA“* nurodomi *GeoJSON* objekto tipas ir naudojama koordinačių sistema:

```
const GEOGRAPHY_METADATA =
    {
        "type": "FeatureCollection",
        "crs": {
            "type": "name",
            "properties": "urn:ogc:def:crs:OGC:1.3:CRS84"
        },
        "features": []
    };
```

Taip suformatuotą objektą OpenLayers priima ir pateikia vartotojui kaip žemėlapio sluoksnį, kuriame plūdurai vaizduojami kaip taškai žemėlapyje. Funkcijos, kuriomis atliekamos užklausos į duomenų bazę, kurių rezultatai naudojami plūdurams ir jų duomenims ant žemėlapio atvaizduoti yra faile *„GraphHelper.js“*.

4.5.Gradiento kūrimas

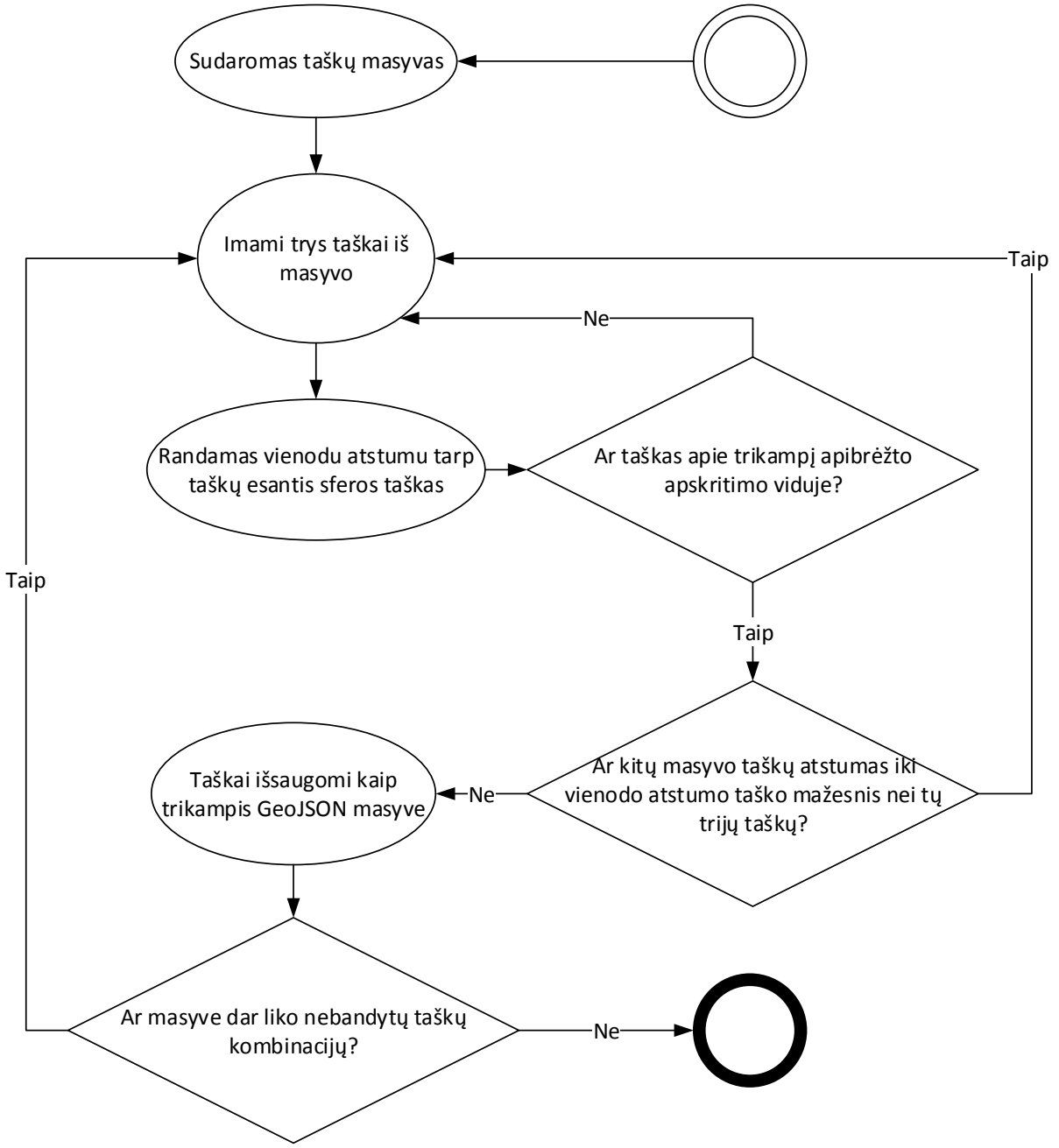
Turint kelis taškus su išmatuotais meteorologiniais parametrais, kurie atvaizduoti žemėlapyje, naudinga išmatuotomis vertėmis nuspalvinti ribas tarp išmatuotų taškų. Toks duomenų atvaizdavimas vadinamas gradientu.

OpenLayers stiliaus funkcijos palaiko dažnio žemėlapius (*angl. heatmap*) ir linijinius gradientus, bet nei vienas jų netinka šios aplikacijos atveju. Tai sprendžiama simuliuojant gradientinį žemėlapį. Visų pirma, taikoma Delaunay trianguliacija:

1. Sudaromas taškų masyvas.
2. Iš eilės imami trys taškai.
3. Apskaičiuojamas taškas, esantis ant sferos paviršiaus vienodu atstumu tarp tų trijų taškų (*angl. equidistant*).
4. Tokie taškai ant sferos galimi du, todėl patikrinama, ar taškas apie trikampį apibrėžto apskritimo viduje, o ne jo išorėje.
5. Jei taškas tinkamas, tikrinama, ar nėra kitų taškų, kurių atstumas iki gauto trikampio vidurio taško mažesnis, nei parinktų trijų taškų (ar taškas nėra apie trikampį apibrėžto apskritimo viduje).
6. Jei tokių taškų nėra, taškai išsaugomi į *GeoJSON* masyvą kaip trikampis.
7. Pasirenkama kita taškų kombinacija ir tęsiamas skaičiavimas, kol išbandomos visos taškų kombinacijos.

Kaip jau žinome iš antrojo skyriaus, taikant Delaunay trianguliaciją, trikampiai sudalinami taip, kad apie trikampį nubrėžus apskritimą, apskritimo viduje nebūtų taškų, kurie nepriklauso tam trikampiui. Dvimatėje erdvėje tai padaryti nėra sudėtinga, tačiau žemėlapis yra sferinis, todėl

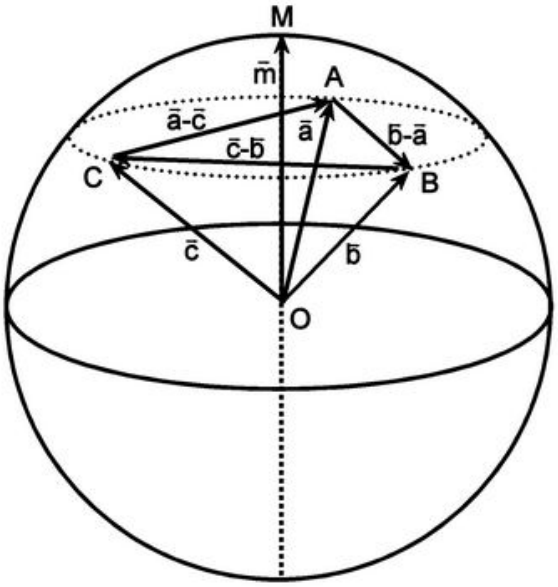
reikėjo taikyti geometrinius ir trigonometrinius metodus, skirtus sferos paviršiaus dalių apskaičiavimui. Delaunay trianguliacijos algoritmas pateikiamas 14 pav.:



pav. 14 Delaunay trianguliacijos algoritmas

Pavyzdžiui, vienodo atstumo taškas apskaičiuojamas formule, gaunama iš sferinės geometrijos. Ant sferos turimi trikampio taškai A, B ir C ir reikia rasti vidurio tašką M. Taškai apibrėžia plokštumą, kertančią sferą į dvi dalis. Bet kokia linija, statmena dviem susikertančioms linijoms bus statmena plokštumai, apibrėžtai tų linijų. Dėl to, linija per sferos vidurio tašką O, kuri statmena linijoms AB, BC ir AC kirs sferą per tašką M, kurio atstumas iki A, B ir C bus vienodas. Kiekvieną tašką galima laikyti vektoriumi, kuris prasideda nuo žemės centro ir baigiasi ties taško koordinatėmis, o liniją tarp dviejų taškų galima išreikšti dviejų vektorių skirtumu. Tokiu atveju vidurio taško M vektorius μ apskaičiuojamas kaip vektorinė b-a ir c-b sandauga:

$$\mu = (\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{b}) = \mathbf{b}\times\mathbf{c} - \mathbf{b}\times\mathbf{b} - \mathbf{a}\times\mathbf{c} + \mathbf{a}\times\mathbf{b} = \mathbf{a}\times\mathbf{b} + \mathbf{b}\times\mathbf{c} + \mathbf{c}\times\mathbf{a}$$



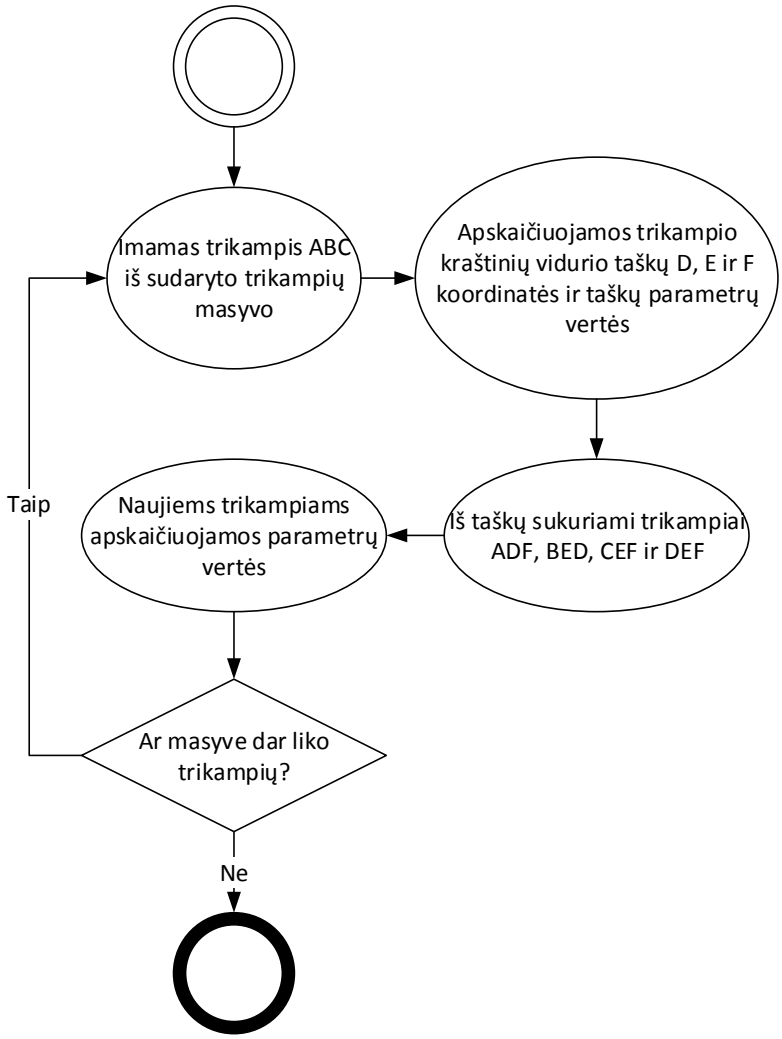
15 pav. Trikampio vidurio taško radimas ant sferos paviršiaus [27]

Skaičiavimams supaprastinti, GPS koordinačių sistema pakeičiama į trimatę de Karto koordinačių sistemą, o sferos spindulys prilyginamas vienetui. Trikampio taškų koordinatės konvertuojamos į tos sferos taškus ir atliekami aukščiau pavaizduoti skaičiavimai. Atlikus vektorinę sandaugą su atskiromis erdvės koordinatėmis, iš de Karto sistemos vektorius konvertuojamas į tašką su GPS koordinatėmis [27].

Tačiau šia formule gali būti apskaičiuojami du taškai: taškas M ir jam priešingas taškas Ṁ, esantis kitoje sferos pusėje. Koks taškas bus apskaičiuotas, priklauso nuo trikampio taškų eilės tvarkos. Jei taškai ant sferos trikampio taškai išdėstyti prieš laikrodžio rodyklę, tada bus gaunamas taškas M, jei palei laikrodžio rodyklę – taškas Ṁ. Kadangi nėra lengvo būdo apskaičiuoti, kokia tvarka eina taškai, apskaičiavus vidurio tašką, pamatuojamas jo atstumas iki trikampio taškų. Jei atstumas per didelis, toks, trikampis atmetamas.

Apskaičiavus visus pagal Delaunay trianguliaciją validžius trikampius, taikoma fraktalizacija [28]. Trianguliacijos metu gauti trikampiai sudalinami į keturis panašius, vienodo ploto trikampius. Trikampių sudalinimas vyksta tokia tvarka (16 pav.):

1. Iš Delaunay trianguliacijos metu sudaryto trikampių masyvo pasirenkamas trikampis ∆ABC;
2. apskaičiuojami trikampio kraštinių vidurio taškų D, E, F koordinatės ir taškų parametrų vertės;
3. sukuriami keturi nauji trikampiai: ∆ADF, ∆BED, ∆CEF ir ∆DEF;
4. trikampiams apskaičiuojamos parametrų vertės, pagal kurias trikampiai bus spalvinami.



16 pav. Trikampių fraktalizavimo algoritmas

Kiek kartų trikampiai dalinami priklauso nuo sistemos administratoriaus pasirinkimo. Kuo daugiau kartų dalinamas trikampis, tuo jis bus panašesnis į gradientą skirtingomis spalvomis jį nuspalvinus pagal išmatuotus parametrus, tačiau daugiau trikampių taip pat reiškia didesnį *GeoJSON* objektą (kompiuterio atminties atžvilgiu) ir viršyjus tam tikrą dydį arba baigsis *AJAX* užklausai skirtas laikas bekuriant objektą, arba naršyklė nepajėgs atvaizduoti tiek daug geografinių objektų.

4.6. Surinktų duomenų atvaizdavimas

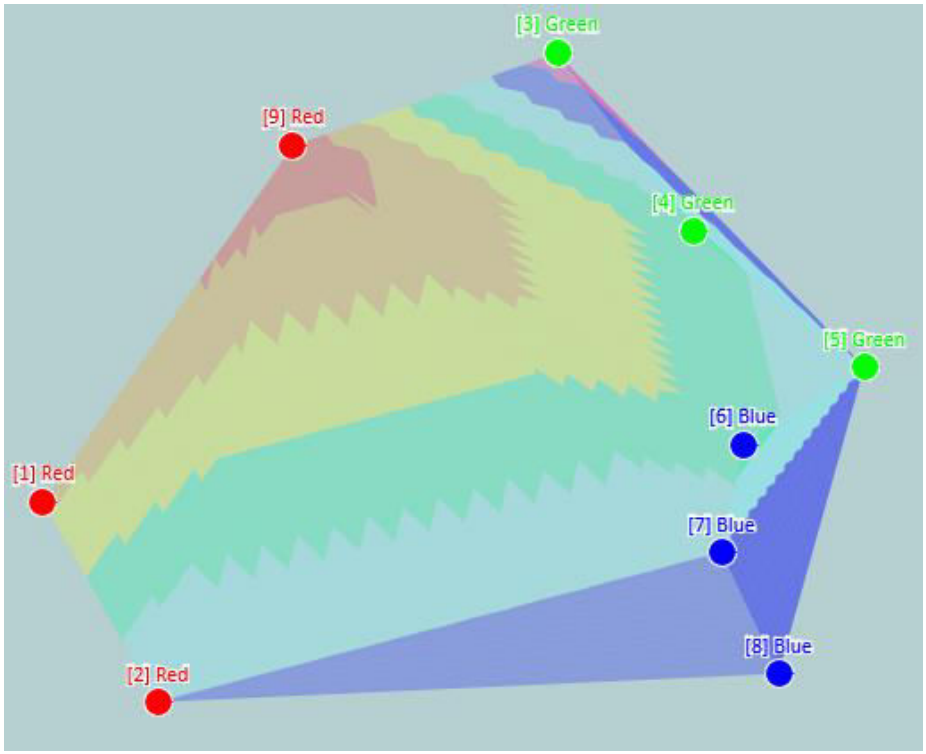
Sistemos surinkti duomenys vaizduojami dviejuose sluoksniuose: gradiento sluoksnyje „triangulationLayer“ ir plūdurų sluoksnyje „buoyLayer“. Duomenims perduoti naudojamas *GeoJSON* formatas aprašo tik geometrinių figūrų vietą ir formą ir neaprašo jų stiliaus atributų: linijų storio, spalvų ir pan. Stilius aprašomas *OpenLayers* sluoksnio stiliaus objekto atributais arba stiliaus funkcija.

Plūdurų sluoksnio šaltinis sudarytas iš taškų tipo objektų, todėl jų stiliaus funkcijoje suformuojamas taškų stilius. Plūduras atvaizduojamas kaip apskritimas, kurio spindulys 8 pikseliai, kuris apvestas 1 pikselio storio baltais kraštais, o jo spalva priklauso nuo plūduro plūdurui nurodytos spalvos. Virš plūduro pateikiama jo etiketė, kurioje pateikiamas įrašo eilės numeris ir plūduro pavadinimas, gaunamas iš duomenų bazės. Plūduro sluoksnio stiliaus funkcija aprašoma taip:

```
var buoyStyleFunction = function (feature, resolution) {
    var feature_id = feature.getId();
    var properties = feature.getProperties();
    var color = properties.color;
    var label = properties.buoy_name;

    buoyStyleCache[feature_id] = new ol.style.Style({
        image: new ol.style.Circle({
            fill: new ol.style.Fill({
                color: color
            }),
            stroke: new ol.style.Stroke({
                color: '#ffffff',
                width: 1
            }),
            radius: 8
        }),
        text: new ol.style.Text({
            font: '12px Calibri,sans-serif',
            fill: new ol.style.Fill({color: color}),
            stroke: new ol.style.Stroke({
                color: '#ffffff',
                width: 2
            }),
            text: '[' + feature_id + '] ' + label,
            offsetY: -15
        })
    });
    return [buoyStyleCache[feature_id]];
};
```

Alternatyviai aprašoma ir trikampių sluoksnio stiliaus funkcija, tačiau joje spalva parenkama pagal tai, kiek pasirinkto rodyti matmens vertė skiriasi nuo maksimalios ir minimalios to matmens vertės. Kaip į duomenų bazę įvedus 3 plūdurus, kurių kiekvienas atliko po 3 matavimus, atrodo tų matavimų duomenys, pavaizduota 17 pav.:



17 pav. Stiliaus funkcijomis stilizuoti duomenys

Stiliaus funkcijos ima du atributus: geometrinę figūrą, kuriai taikomas stilius ir žemėlapio rezoliuciją. Geometrinės figūra naudojama tam, kad pagal jos atributų vertes būtų galima keisti funkcijos stilių. Tuo tarpu žemėlapio rezoliucija leidžia pakeisti objekto stilių, kai vartotojas pakeičia žemėlapio mastelį. Taip pat stiliaus funkcijos kešuojamos, kad nereikėtų jų naudoti kiekvieną kartą perpiešiant žemėlapį.

4.7.Interaktyvių žemėlapio elementų kūrimas

Visi *OpenLayers* žemėlapiai yra interaktyvūs: juose galima keisti žemėlapyje matomą plotą ir mastelį. Tačiau *OpenLayers* turi ir kitų interaktyvių funkcijų bei leidžia jų susikurti programuotojui. Kuriant sistemos prototipą buvo sukurtas interaktyvus išlendantis langas (*angl. popover*), atsidarantis nuspaudus ant plūduro objekto žemėlapyje, kuriame parodoma informacija apie plūdurą, matavimo laiką ir geografines koordinates, toje vietoje atliktus matavimus ir galimybes pakeisti matavimą, pagal kurį spalvinamas trianguliacijos sluoksnis.

Išlendančiam langui atidaryti reikia, kad naršyklė žinotų, kada vartotojas paspaudžia pelės klavišą ir kokia geometrinė figūra yra pelės paspaudimo vietoje. Pelės paspaudimui ant žemėlapio *OpenLayers* turi JavaScript įvykį *on('click')*, kuriam įvykus nuspaustą geometrinę figūrą galima su funkcija *getFeatureAtPixel()*, kuri randa visus geometrinius objektus pelės paspaudimo vietoje. Jei paspaudimo vietoje rastas objektas, atidaromas išlendantis langas ir jame pateikiama objekto informacija. Objekto gavimo funkcija atrodo taip:

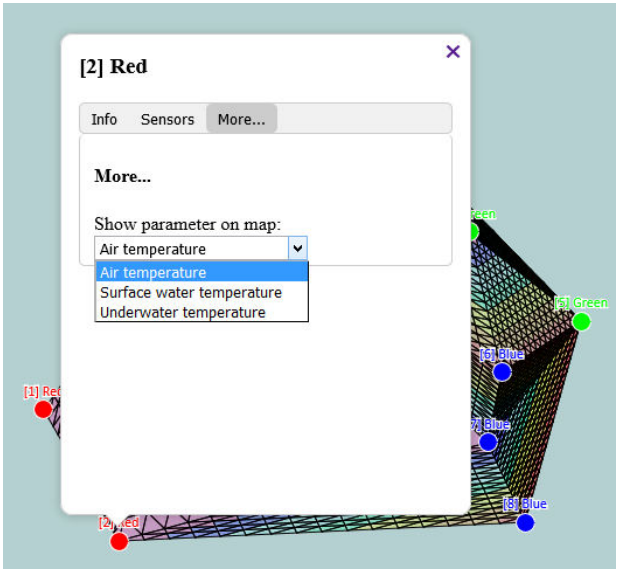
```
map.on('click', function (evt) {
    var feature = map.forEachFeatureAtPixel(evt.pixel,
        function (feature, layer) {
            return feature;
        });
    if (feature) { ... }
});
```

Informacijos atnaujinimui lange panaudojama *VueJS* biblioteka. Užkrovus žemėlapio langą, sukuriamas *VueJS* objektas, sudarytas iš nuorodos į *DOM* elementą, objekto kintamųjų ir metodų bei funkcijų, įvykdomų objekto veikimo metu. Šiuo atveju nuspaudus ant žemėlapio objekto, jo atributai perduodami VueJS objektui. Pakeitus objektus, pasikeičia ir iššokančio lango turinys.

Išlendančiame lange galima pakeisti rodomą trianguliacijos sluoksnį. Tam panaudojamas *VueJS* metodas, įgyvendinamas pakeitus rodomą parametrą *HTML select* elemente. Pakeitus VueJS elementus, kurie atvaizduojami puslapyje, elementų vertės puslapyje taip pat automatiškai pakeičiamos. Pakeitus *„select“* pasirinkimą, įvyksta *„change“* įvykis, kuris surišamas su VueJS metodu naudojant *@change="metodo_pavadinimas“* tipo sintaksę:

```
<select id="display_options" @change="changeDisplayOption">
    <option v-for="option in optionsDisplay"
:value="option.i_measurement_id">{{option.description}}</option>
</select>
```

Pasirinkimo atributas *„v-for“* aprašo *VueJS* ciklą pasirinkimams aprašyti, value vertė nurodo *VueJS* objekto atributą, kuris pakeičiamas objekto verte puslapio krovimo metu, o dvigubais figūriniais skliaustais įvedamas objekto atributas, naudojamas kaip vartotojui rodomas pasirinkimo pavadinimas. Išlendantis lango dalis su parametro pasirinkimu pateikiama 18 pav.:



18 pav. Išlendantis langas žemėlapyje

4.8.Duomenų redagavimas

Duomenų redagavimui sukurta redagavimo aplinka, prie kurios vartotojui reikia prisijungti. Prisijungimui įvedamas vartoto vardas ir slaptažodis ir tikrinamas su duomenų bazėje esančiais vartotojais. Tam, kad slaptažodis duomenų bazėje nebūtų saugomas nešifruotas, naudojama *„bcrypt“* šifravimo biblioteka. Pirmą kartą vartotojui prisijungus sukurianti šifravimo druską (*angl. salt*), kurios dėka užšifruojamas slaptažodis ir vėliau naudojama slaptažodžiui patikrinti naudojant *„bcrypt.compare()“* funkciją. Sukūrus vartotojo druską ir užšifravus slaptažodį, vartotojas išsaugomas duomenų bazėje:

```
exports.createUser = function (username, password, email, t_user_id) {
    bcrypt.genSalt(10, function (err, salt) {
        bcrypt.hash(password, salt, function (err, hash) {
            knex.insert({ ... }).then();
        });
    });
};
```

Prisijungus prie administravimo sistemos, atsidaro interneto puslapis su nuoroda į įrašo pridėjimo langą ir lentele, kurioje atvaizduojami plūdurų matavimai su plūduro pavadinimu, matavimo laiku ir geografinėmis koordinatėmis ir trimis galimais veiksmais: įrašo pilna peržiūra, įrašo koregavimu ir ištrynimu.

Kiekvienas iš pasirenkamų veiksmų atidaro modalinį langą (*angl. modal window*). Pats langas sukurtas su *CSS* ir *HTML5* technologijomis, o jo turinys valdomas su *VueJS*. Kokį langą atidaryti, nuorodos elementas sprendžia iš nuorodos *„href“* atributo, o kokius duomenis rodyti – iš atributo *„data-row-id“,* kurio vertė pateikiama iš *VueJS* objekto ir su nuorodos paspaudimo įvykiu susieto VueJS metodo [28].

Duomenys, kuriuos reikia atvaizduoti gaunami iš duomenų bazės asinchroniškai, serveriui pateikus *AJAX* užklausą, o serveris kaip atsakymą pateikia *JSON* objektą. Užklausos pateikiamos su anksčiau naudota *„Axios“* biblioteka. Redagavimo sistemos *VueJS* objekto kodas iš esmės atrodo taip:

```
var vueApp = new Vue({
    el: '#app',
    data: {
        tableData: {},
        formData: {},
        viewModalRow: {},
        buoyTypes: undefined,
        buoyInstances: [],
        buoySensors: undefined,
        insertFormInputs: { ... },
        updateFormInputs: { ... }
    },
    methods: {
        selectRows: _selectRows,
```

```
        changeRow: _changeRow,
        ...
        loadInstances: _loadInstances
       },
    mounted: function() {
        var self = this;
        self.selectRows(0);
    }
});
```

Kur „mounted“ funkcija įvykdoma pasikeitus objekto duomenims. Pakeitimas vykdomas asinchroniškai ir neperkraunant puslapio. Redagavimo aplinka pateikiama 19 pav.:

Buoy Record Data

[Add]

Record ID	Buoy Type	Buoy	Time	Location	Actions
2	Plūduras	[1] Red	2017-03-29T22:27:31.176Z, 2017-03-29T22:28:44.433Z	55.673241, 21.070147	[View] [Edit] [Delete]
3	Plūduras	[2] Green	2017-03-29T22:27:16.621Z, 2017-03-29T22:30:24.071Z	55.677646, 21.074953	[View] [Edit] [Delete]

19 pav. Plūdurų matavimų redagavimo aplinka

5. PROTOTIPO BANDYMAI

Sukūrus duomenų kaupimo ir vizualizavimo sistemos prototipą, reikia atlikti prototipo tyrimus. Vizualizavimo sistemą būtų galima išbandyti su simuliuotais arba realiais duomenimis.

5.1. Prototipo bandymas

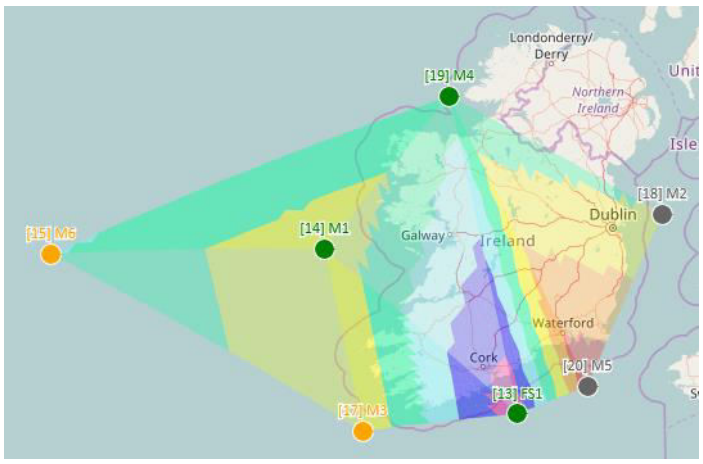
Tyrimą pasirinkta atlikti su realių matavimų duomenimis, tačiau dar nebuvo atlikti tyrimai su Klaipėdos universitete kuriamais belaidžiais mobiliais plūdurais, todėl pasitelkti internete prieinami duomenų šaltiniai. Toliau pateikiamas duomenų šaltinio aprašymas ir kaip vizualizavimo sistema interpretuoja šaltinio duomenis.

Naudojamas duomenų šaltinis gautas iš Airijos Jūrų instituto meteorologinių stočių inkaruotų plūdurų tinklo. Surinkti tinklo parametrai sudaryti iš geografinių koordinačių, datos ir laiko, atmosferos slėgio, oro temperatūros, rasos taško temperatūros, vėjo greičio (mazgais), maksimalaus vėjo gūsio greičio, vėjo krypties (laipsniais), jūros paviršiaus temperatūros, bangų periodo (sekundėmis), bangų aukščio (metrais) ir santykinės drėgmės (%). Darbo rašymo metu, duomenys iš plūdurų M2, M3, M4, M5 ir M6 prieinami realiu laiku. Istoriniai duomenys taip pat prieinami iš plūdurų M1, FS1 ir M4 originalios vietos. Plūdurai inkaruoti plote tarp 55 ir 51.2167 laipsnių šiaurės platumos ir 5.425, 15.558 laipsnių rytų ilgumos. Koordinatės pateikiamos laipsnių šimtosiomis (ne laipsniais, minutėmis ir sekundėmis), naudojama *EPSG:4326* geografinių koordinačių sistema [29].

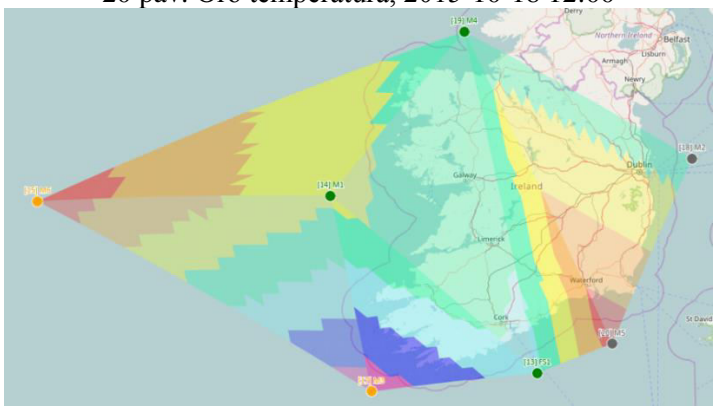
Eksperimento metu fraktalizacija atlikta 4 kartus. Tai reiškia, kad 1 Delaunay trianguliacijos metu gautas trikampis sudalinamas į 256 trikampius. Tai nėra pakankamai daug gauti lygius perėjimus iš vieno parametro vertės intervalo į kitą, tačiau bandant dalinti trikampius daugiau kartų, sistema tampa nestabili. Sistemos stabilumą pagerinti būtų galima sluoksnį pateikus ne vektoriniu, o rasteriniu formatu, tačiau tokiu atveju sistema netektų greito rodomo parametro keitimo galimybės.

5.2. Bandymo rezultatai

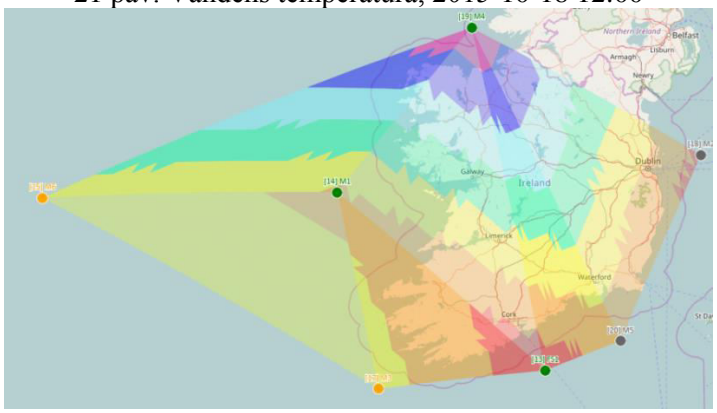
Sistemos prototipo patikrai, gaunami plūdurų duomenys ir pasirenkami laikas ir data, kai visi plūdurai aktyvūs. Pasirinkta 2005 metų spalio 18 diena, 12 valanda. Suvedus plūdurų matavimų duomenis į sistemą, gaunami plūdurų ir trianguliacijos sluoksniai. Rezultatai pateikiami 20-22 pav.:



20 pav. Oro temperatūra, 2015-10-18 12:00



21 pav. Vandens temperatūra, 2015-10-18 12:00



22 pav. Atmosferos slėgis, 2015-10-18 12:00

Kadangi atstumai tarp bandyme naudotų matavimo taškų buvo labai dideli (apie 400 km), gradientų patikimumas mažas. Naudojamas vizualizavimo metodas neatsižvelgia į atstumus tarp taškų, sistema žymiai tiksliau veiktų tinkluose, kuriuose matavimo taškų yra daugiau ir atstumai tarp jų yra mažesni. KU Kuriamame plūdurų tinke planuojamas maksimalus atstumas tarp plūdurų - 1 km., todėl gradientų patikimumas bus tikslesnis. Antra vertus, daugumai „karščio žemėlapio“ (*angl. heatmap*) algoritmų reikia apie 400 taškų, kad jie būtų patikimi, todėl negalima tikėtis, kad septynių plūdurų tinklas duos patikimus rezultatus. Nelygumams tarp gradiento ribų sumažinti, būtų galima trianguliacijos trikampius dalinti nelygiomis dalimis, tačiau tai padidintų fraktalizacijos algoritmo sudėtingumą ir skaičiavimų kiekį [30].

6. IŠVADOS

1. Atlikta duomenų kaupimo ir vizualizavimo sistemų analizė ir nustatyta, dažniausiai naudojami *METOCEAN*, *OMNI MSB* ir *Xbeach* ir kt. architektūrų plūdurai, bangų modeliavimui naudojantys akselerometrus, kurių duomenys apdorojami Furjė transformacijos algoritmu ir GPS imtuvus plūduro vietai nustatyti.
2. Atlikus vizualizavimo, interpoliacijos ir matmenų mažinimo metodų analizę, buvo nustatyta, kad kuriamos sistemos meteorologinių duomenų interpoliavimui, dėl nesudėtingos algoritmo realizacijos *JavaScript* programavimo kalba ir jo greitaveikos, labiausiai tinkama Delaunay trianguliacija.
3. Pagal sudarytus sistemos reikalavimus, pasirinkti programinės įrangos kūrimo įrankiai: *PostgreSQL* duomenų bazė, dėl jos palaikomų GIS duomenų tipų, *NodeJS* ir *Express* karkasai, dėl asinchroninių funkcijų palaikymo, *OpenLayers,* dėl plataus geografinių objektų palaikymo ir *VueJS* karkasas, dėl reaktyvaus programavimo palaikymo bei *GeoJSON* duomenų formatas, suderinamas su *JavaScript* programavimo kalba ir skirtas GIS duomenų perdavimui.
4. Realizuotas sistemos prototipas, kuris žemėlapyje atvaizduoja duomenis ir vizualizuoja atliktos interpoliacijos rezultatus. Taip pat sukurta galimybė redaguoti ir išsaugoti duomenis.
5. Naudojant Airijos plūdurų tinklo meteorologinių parametrų stebėjimo duomenis, atliktas vizualizavimo sistemos bandymas. Bandymui naudoti 7 plūdurų, išdėstytų apie 400 km atstumu vienas nuo kito, duomenys. Nustatyta, jog toks atstumas ir plūdurų kiekis yra per mažas, o norint, kad interpoliacijos rezultatai būtų tikslesni, reikalingas didesnis matavimo taškų kiekis ir mažesni atstumai tarp jų.

7. SISTEMOS TOBULINIMO GALIMYBĖS

Sukurtas veikiantis, minimalų funkcionalumą turintis sistemos prototipas. Siekiant platesnio sistemos pritaikymo, prototipą reikėtų praplėsti toliau aprašomomis sistemos funkcijomis.

Numatoma išbandyti daugiau duomenų vizualizavimo algoritmų. Skirtingi algoritmai gali geriau veikti su skirtingo tipo duomenimis. Taip pat planuojama įgyvendinti kelių matuojamų parametrų atvaizdavimą vienu metu. Tai planuojama atlikti naudojant ant žemėlapio rodomomis etiketėmis ir matmenų mažinimo metodais. Taip pat galutinėje sistemoje būtina atvaizduojamo parametro verčių legenda, nes dabar nėra aišku, kiek mažiausia parametro vertė skiriasi nuo didžiausios.

Siekiama sistemą naudoti ne tik viešiems duomenims pateikti, bet ir leisti vartotojams kurti privačius žemėlapius, kuriuos vartotojai pasiektų su specialia nuoroda arba tik prisijungę prie sistemos. Pagrindinio puslapio žemėlapį siekiama naudoti tik viešai pasiekiamiems duomenų matavimams.

Redagavimo sistemos galimybes planuojama praplėsti su visų esamų duomenų bazės lentelių redagavimu, skirtingais vartotojų privilegijų tipais, atliktų pakeitimų istorija ir panašiomis funkcijomis.

Duomenų pateikimas į žemėlapį vykdomas *AJAX* užklausų dėka. Šių užklausų dėka galima sukurti sistemos *API*, kuris leistų duomenis pateikti naudotojams, kurie nori rodyti sistemos matavimų duomenis savo interneto puslapiuose arba aplikacijose.

8. LITERATŪRA

[1] Omstedt, A., Elken, J., Lehmann, A., Leppäranta, M., Meier, H., Myrberg, K. and Rutgersson, A. Progress in physical oceanography of the Baltic Sea during the 2003–2014 period, 2014. Progress in Oceanography, 128, p. 139-171.

[2] Dickey, T. Emerging ocean observations for interdisciplinary data assimilation systems, 2003. Journal of Marine Systems, 40-41, p. 5-48.

[3] Ashton, I. and Johanning, L. On errors in low frequency wave measurements from wave buoys, 2015. Ocean Engineering, 95, p. 11-22.

[4] Venkatesan, R., Vengatesan, G., Vedachalam, N., Muthiah, M., Lavanya, R. and Atmanand, M. Reliability assessment and integrity management of data buoy instruments used for monitoring the Indian Seas, 2016. Applied Ocean Research, 54, p. 1-11.

[5] Pearman, D., Herbers, T., Janssen, T., van Ettinger, H., McIntyre, S. and Jessen, P. Drifter observations of the effects of shoals and tidal-currents on wave evolution in San Francisco Bight, 2014. Continental Shelf Research, 91, p. 109-119.

[6] Scott, T., Austin, M., Masselink, G. and Russell, P. Dynamics of rip currents associated with groynes — field measurements, modelling and implications for beach safety, 2016. Coastal Engineering, 107, p. 53-69.

[7] Halliday, J., Dorrell, D. and Wood, A. An application of the Fast Fourier Transform to the short-term prediction of sea wave behaviour, 2011. Renewable Energy, 36(6), p. 1685-1692.

[8] Chen, N., Wang, K., Xiao, C. and Gong, J. A heterogeneous sensor web node meta-model for the management of a flood monitoring system, 2014. Environmental Modelling & Software, 54, p. 222-237.

[9] Hammoudeh, M., Newman, R., Dennett, C. and Mount, S. Interpolation techniques for building a continuous map from discrete wireless sensor network data. Wireless Communications and Mobile Computing, 13(9), 2011, p. 809-827.

[10] Dell'Accio, F., Di Tommaso, F. and Hormann, K. On the approximation order of triangular Shepard interpolation. IMA Journal of Numerical Analysis, 2015, p. 359-379.

[11] Gorai, A. K., Kumar, S. Spatial Distribution Analysis of Groundwater Quality Index Using GIS: A Case Study of Ranchi Municipal Corporation (RMC) Area. Geoinfor Geostat: An Overview 1:2, 2013.

[12] Meng, Q., Liu, Z. and Borders, B. Assessment of regression kriging for spatial interpolation – comparisons of seven GIS interpolation methods. IMA Journal of Numerical Analysis, 2015, 36 (1) p. 359-379.

[13] Chen, K., Anthony, S. and Granick, S. Extending Particle Tracking Capability with Delaunay Triangulation. Langmuir, 30(16), 2014, p. 4760-4766.

[14] Liu, J., Yan, J. and Lo, S. A new insertion sequence for incremental Delaunay triangulation. Acta Mechanica Sinica, 29(1), 2013, pp. 99-109.

[15] Karbauskaitė, R. Daugiamačių duomenų vizualizavimo metodų, išlaikančių lokalią struktūrą, analizė, Vytauto Didžiojo Universitetas, Vilnius, 2010.

[16] Radzevičius, V. Nuotolinių stebėjimų informacinių sistemų analizė ir prototipo kūrimas, Klaipėdos universitetas, Klaipėda, 2016.

[17] The PostgreSQL Global Development Group. PostgreSQL: About, 2017. [Tinkle]. Prieiga per internetą: < https://www.postgresql.org/about/ > [žiūrėta 2017 04 27].

[18] BostonGIS. Cross Compare SQL Server 2008 Spatial, PostgreSQL/PostGIS 1.3-1.4, MySQL 5-6. 2008. [Tinkle] Prieiga per internetą: < http://www.bostongis.com/PrinterFriendly.aspx?content_name=sqlserver2008_postgis_mysql_compare > [žiūrėta 2017 04 27].

[19] Node.js Foundation. About | NodeJS, 2017. [Tinke]. Prieiga per internetą: < https://nodejs.org/about/ > [žiūrėta: 2017-04-28].

[20] StrongLoop, IBM, and other expressjs.com contributors. Express routing, 2016. [Tinke]. Prieiga per internetą: < http://expressjs.com/guide/routing.html > [žiūrėta: 2017-04-28].

[21] OpenLayers Contributors. OpenLayers API doc: Index, 2017. [Tinkle]. Prieiga per internetą: < http://openlayers.org/en/latest/apidoc/ > [žiūrėta: 2017-04-29].

[22] Internet Engineering Taskforce. RFC 7159: The JavaScript Object Notation (JSON) Data Interchange Format, 2014 [Tinkle]. Prieiga per internetą: < https://tools.ietf.org/html/rfc7159 > [žiūrėta: 2017-04-29].

[23] Internet Engineering Taskforce. RFC: 7946: The GeoJSON Format, 2016 [Tinkle]. Prieiga per internetą: < https://tools.ietf.org/html/rfc7946 > [žiūrėta: 2017-04-29].

[24] VueJS contributors. Introduction – VueJS, 2017 [Tinkle]. Prieiga per internetą: < https://vuejs.org/v2/guide/ > [žiūrėta: 2017-04-29].

[25] VueJS contributors. Comparison with Other Frameworks – VueJS, 2017 [Tinkle]. Prieiga per internetą: < https://vuejs.org/v2/guide/comparison.html > [žiūrėta: 2017-04-29].

[26] Jetbrains.com. PhpStorm Features, 2017. [Tinkle] Prieiga per internetą: < https://www.jetbrains.com/phpstorm/features/ > [žiūrėta: 2017-05-23].

[27] Vries-Uiterweerd, G. Spherical Geometry in Mirifiques aventures de maître Antifer, Verniana: Verniana. Jules Verne Studies. Volume 2, p. 1–10, 2009. [Tinkle]. Prieiga per internetą: < http://www.verniana.org/volumes/02/HTML/SphericalGeometry.html > [žiūrėta: 2017-05-18].

[28] Reinl, H. C., et al. CSS Modal - Modals built out of pure CSS, 2017. [Tinkle]. Prieiga per internetą: < http://drublic.github.io/css-modal/ > [žiūrėta: 2017-05-19].

[29] Ireland‘s Open Data Portal. Weather Buoy Network. [Tinkle]. Prieiga per internetą: < https://data.gov.ie/dataset/weather-buoy-network > [žiūrėta: 2017-05-20].

[30] Liu, Y. and Li, M. Iso-Map: Energy-Efficient Contour Mapping in Wireless Sensor Networks. 27th International Conference on Distributed Computing Systems (ICDCS '07), 2007.

Jagminas Justas

Meteorologinių duomenų rinkimo ir vizualizavimo sistemos prototipo kūrimas:

Techninių informacinių sistemų inžinerijos studijų magistro darbas. Vadovas: Doc. Dr. Gediminas Gricius.

Darbo apimtis – 58 p. teksto, 22 paveikslų, 4 lentelės, 30 bibliografinių šaltinių, 1 priedas – 1 p.

SANTRAUKA

Mažėjant belaidžių jutiklių kainoms, jei vis plačiau naudojami įvairiose srityse, įskaitant ir meteorologinių sąlygų stebėjimą. Naudojant daug jutiklių, iškyla problema su jutiklių matavimų duomenų kaupimu ir vizualizavimu. Analizuojama, kaip veikia tokio tipo sistemos bei jų veikimo principai. Analizuojami galimi duomenų vizualizavimo metodai. Pasinaudojus analizės rezultatais, sudaromi reikalavimai sistemai ir pagal juos pasirenkami programiniai įrankiai. Kuriamas informacinės meteorologinių duomenų sistemos prototipas. Vizualizavimo sistemos prototipas sukuriamas pritaikinius Delaunay trianguliaciją ir kitus geometrijos metodus. Vėliau sistemos prototipas išbandomas su realių belaidžių jutiklių sistemų surinktais duomenimis.

Raktažodžiai: GPS, monitoringas, plūdurai, NodeJS, OpenLayers, vizualizavimo sistemos, Delaunay trianguliacija.

Jagminas Justas

Development of a Prototype for Meteorological Data Collection and Visualisation System

Technical Information Systems Engineering studies Master‘s thesis. Supervisor: Doc. Dr. Gediminas Gricius.

Work size – 58 p. text, 22 figures, 4 tables, 30 bibliographic sources, 1 appendix – 1 p.

SUMMARY

With the prices of wireless sensors decreasing, they are more broadly used in various fields, including meteorological data collection. While using many sensors, a problem with data storage and visualisation emerges. In this work, the systems of this type and their means of operation are analysed. By using the results of the analysis, requirements for the data collection and visualisation system are developed. The prototype of visualisation system is created by using Delaunay trianguliation and other methods of geometry. Later the system of the prototype is tested with data from real wireless sensor systems.

KEYWORDS: GPS, monitoring, buoys, NodeJS, OpenLayers, visualisation systems, Delaunay trianguliation.

9. PRIEDAI

1 PRIEDAS

1. Kompaktinė plokštelė

Kompaktinis diskas, kuriame pateikiami:

Darbo katalogas: „*Jagminas_Justas/..*“.

Darbo aprašymas: „../Jagminas_Justas_baigiamasis_darbas.docx“.

Darbo prezentacija: „../Jagminas_Justas_prezentacija.pptx“.

Sukurtos aplikacijos programinis kodas: „../Programa“.