TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Johann Veispak 179238

Täppisväetamise kasutajaliides

Bakalaureusetöö

Juhendaja: Evelin Halling

tarkvarateaduse instituut, PhD

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Johann Veispak

06.05.2022

Annotatsioon

Tänasel ajal, eriti arvestades praegust geopoliitilist olukorda, mistõttu on võimalik, et maailmas võib tekkida toidupuudus ning väetise hinnatõus, on oluline, et põlde väetatakse võimalikult efektiivselt.

Efektiivse väetamise üheks variandiks on siduda põlluharimine info- ja kommunikatsioonitehnoloogiatega, mis võimaldaks põllumehel saada asjakohast informatsiooni põllu mullastiku seisukorra kohta ning aitaks tal vastavalt mulla seisundile väetada.

Töö eesmärk on luua veebirakendus, mis võimaldab kasutajal märkida ära kaardi kaudu põlde ning kuvada talle tema enda võetud mullaproovide tulemusi. Kasutaja saab mõõtmistulemused põlluga ühendada, mis võimaldab kasutajal näha põllu mullakoostise muutust ajas.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 6 peatükki, 7 joonist.

Abstract

Precise Fertilizer Application User Interface

Nowadays, given the tense geopolitical situation, it is possible that in the near future, there could occur some food shortages, and it is almost certain that fertilizer prices will continue to rise. Therefore it is essential that fertilizer is applied to arable lands optimally.

One way to optimise the application of fertilizer is to, combine agriculture with the endless possibilities of information technology. This can lead to the farmer making informed decisions regarding the use of fertilizer, as information technology can proovide him the means to measure and see the key parameters of the soil on his fields.

The goal of this thesis is to create a web application, which provides the user the means to mark one's fields on a map and then display the results of taken soil samples. The user can connect multiple soil samples to a field to visualize changes in the soil composistion chronologically.

The thesis is in Estonian and contains 25 pages of text, 6 chapters, 7 figures.

Lühendite ja mõistete sõnastik

GeoJSON	JSON vormingul põhinev standardvorming lihtsamate geograafiliste tunnuste esitamiseks.
WMS	*Web Map Service*, protokoll georefereeritud kaardipiltide teenindamiseks üle Interneti.
SRS/CRS	*Spatial reference system, coordinate reference system*, raamistik, mida kasutatakse Maa pinnal olevate asukohtade mõõtmiseks koordinaatidena.
API	*Application Programming Interface*, rakendusliides.
TMS	*Tile Map Service*, protokoll kaardiruutude pärimiseks.
WFS	*Web Feature Service*, protokoll geograafiliste andmete pärimiseks/muutmiseks.
EPSG	*European Petroleum Survey Group*, teaduslik organisatsioon, mis oli seotud Euroopa kütusetööstusega, nüüdseks on tegemist registriga, milles hoiustatakse geograafilise, geodeetilise andmeid.
HTML	*HyperText Markup Language*, veebilehtede märgendamiskeel.
CSS	*Cascading Style Sheets*, küljendamisel kasutatav märgistuskeel.
CORS	*Cross-Origin Resource Sharing*, HTTP päringu päisepõhine mehhanism, mis lubab brauseril saada kätte teises asukohas olevatele andmetele/failidele.
JWT	*JSON Web Token*, väljapakutud internetistandard, kasutaja autentimisel väljastatud luba, mis sisaldab kasutaja andmeid.
JPA	*Java Persistence API*, programeerimisliidese spetsifikatsioon, mis kirjeldab relatsiooniliste andmete haldamist.
ORM	*Object-relational mapping*, programmeerimistehnika, mis rakendab objektorienteeritud programmeerimiskeeli teisendamaks andmeid tüüpsüsteemide vahel.
CI/CD	*Continuous integration / Continuous delivery*, pideva integreerimise ja pideva tarnimise praktika.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Viimase saja aasta jooksul on maailma elanikkond ligikaudu neljakordistunud ning sellest on tulenenud ka vajadus palju suurema toidu koguse järele, et kõiki saaks ära toidetud. Läinud sajandi keskpaigas alanud nii nimetatud roheline revolutsioon, mille käigus põllumajadustoodang märkimisväärselt kasvas, sai võimalikuks eeskätt uute, viljakamate taimesortide ja mineraalväetiste kasutuselevõtuga.

Põllumajanduse intensiivistumine on kahtlemata võimaldanud rohkem inimesi ära toita, kuid samas on sellel ka omad varjuküljed. Eeskätt väetiste laialdane kasutamine, eriti liigne väetamine, avaldab ökosüsteemile negatiivset mõju. Näiteks lämmastik, võib sattuda põllumajandusmaadelt nitraatidena (NO_3^-) sattuda veekogudesse, kus liigne kogus seda võib esile kutsuda veekogu eutrofeerumise. Sellest omakorda võivad tekkida nii nimetatud surnud tsoonid, milles elutegevus on erinevatel veeloomadel on raskendatud, kui mitte võimatu madala hapnikusisalduse pärast vees. [1]

Viimasel ajal, seoses geopoliitiliste arengutega, on väetiste hind märkimisväärselt tõusnud. Lääne kehtestatud sanktsioonid Venemaale ja Valgevenele, maailma ühtedele suurimatele väetiseeksportijatele (40% maailma kaaliumkarbonaadi (*potash*) ekspordist tuleb nendest riikidest), on jätnud oma jälje väetiste hindadele. Ameerika Ühendriikides prognoositakse, et väetiste hind kasvab antud aastal ligikaudu 12%. [2]

Kuivõrd üleväetamisel on negatiivsed tagajärjed ökosüsteemile ning aina kasvavad väetisehinnad avaldavad negatiivset mõju põllumeestele, siis seetõttu on oluline, et väetist kasutatakse võimalikult optimaalselt ja efektiivselt, et kasvatatav vili saaks piisavalt toitained. Selles osas saabki antud bakalaureusetöö käigus loodav rakendus põllumehi aidata, et nad näeksid, mis seisus on nende põllu mullakoostis, milliseid mineraalaineid on liiga vähe või palju ning siis vastavalt väetamist, kas suurendada või vähendada.

1.1 Eesmärk

Käesoleva bakalaureusetöö eesmärgiks on luua rakendus, millega on võimalik kasutajal, üldjuhul põllumehel, kaardil üles märkida oma põlde ning iga põlluga seonduvalt kuvada mullakoostise andmeid ajas muutuvalt. See võimaldaks kasutajal optimeerida väetise kasutamist, midagi, mis on üpriski aktuaalne antud hetkel, arvestades praegust geopoliitilist olukorda, kus üks maailma suurimaid väetiseeksportijaid osaleb aktiivselt relvakonfliktis.

Mullakoostise andmete kuvamiseks rakenduses peab põllumees enne analüsaatoriga mullaproovi võtma ning telefonirakenduse kaudu proovi andmed ühisesse andmebaasi laadima. Mobiilirakendus ning analüsaator on realiseeritud teiste projektide käigus.

Antud bakalaureusetööga realiseeritakse projekti andmete visualiseerimise pool, teisisõnu luuakse kasutajaliides, millega saab kasutaja oma mõõtmistulemusi tõlgendada ning hallata oma põlde. Rakenduse peamine funktsionaalsus koosneb kaardist, mida kasutaja saab modifitseerida (lisada, eemaldada, muuta oma põlde). Rakendus on oma olemuselt üleilmne, kuid kaardi kõige täpsem osa tuleb Eesti Vabariigi pinnal, kus kasutatakse Maa-ameti andmeid.

Seatud eesmärgi täitmiseks tuleb lahendada alljärgnevad ülesanded:

- Uurida olemasolevaid täppisväetamise lahendusi.
- Uurida, kuidas inkorporeerida Maa-ameti andmeid kaardile.
- Kasutajaliidese disain ja arendus.
- Rakenduse testimine ning pidev integratsioon (*continuous integration*).

1.2 Ülesehitus

Bakalaureusetöö koosneb järgnevatest osadest:

- Esmalt uuritakse tausta, milliseid olemasolevaid täppisväetamise lahendusi leidub.
- Teiseks, formuleeritakse nõuded loodavale rakendusele ning uuritakse võimalusi kaartide kasutamiseks rakenduses.
- Viimasena tehakse ülevaade rakenduse arhitektuurist ning kasutatavatest tehnoloogiatest ning tuuakse näiteid implementatsioonidest.

2 Taustainfo

Antud peatükis uuritakse olemasolevaid täppisväetamise lahendusi, nende eeliseid ja puuduseid.

2.1 I-Plant Nutrition

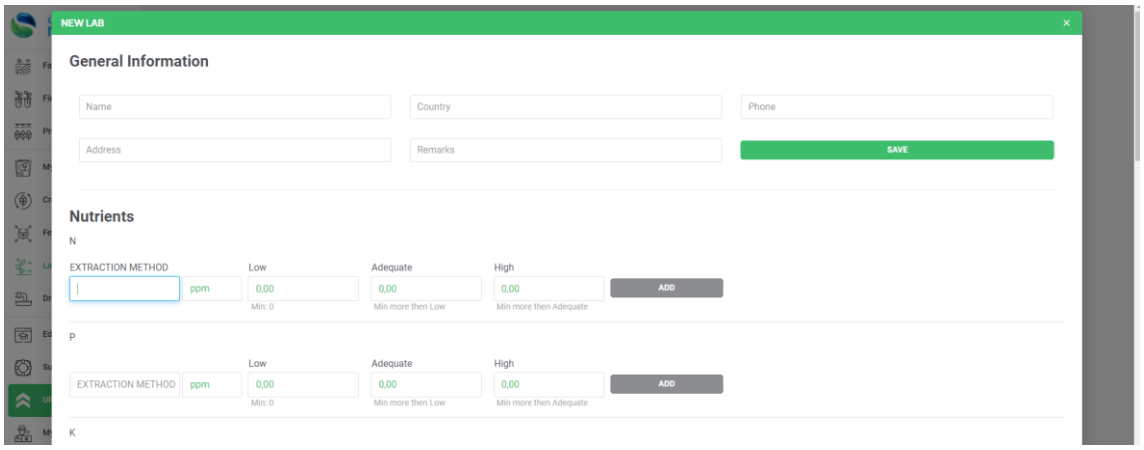
I-Plant Nutrition[1] on veebirakendus, mis pakub kasutajale täisteenust targaks põlluharimiseks. Põllumajandaja saab lisada kaardi peal endale põllu ning koondada mitmeid põlde farmidesse. Peamine funktsionaalsus rakendusel on põldude majandamise, sealhulgas ka väetamise plaanide koostamine. Teenusepakkuja enda agronoomid koostavad kasutajale individuaalsed plaanid.

Rakenduse tugevaks küljeks on kahtlemata individuaalne lähenemine igale kliendile, kasutajal on näiteks võimalik konsulteerida i-Plant Nutrition-i enda agronoomidega, muutmaks oma põlluharimist efektiivsemaks.

Samas tuleb tõdeda, et rakendus eeldab, et kasutaja täidaks palju informatsiooni, mis on kasutajakogemuse mõttes pigem halb. Näiteks peab põllumees ise leidma labori, kes tema asukohariigis võtaks mullaproove, pärast mida peab kasutaja sisestama mullaanalüüsi tulemused rakendusse. Antud tulemuste põhjal koostavad teenusepakkuja spetsialistid rakenduse kasutajale plaani, kuidas efektiivsemal oma põldu harida.

Antud rakendust kahjuks põhjalikult läbi uurida ei saanud, kuna enamus funktsionaalsusest eeldas, et kasutaja on endale paketi tellinud, millest odavaim on 600€ aastas. Samas nii palju kui oli võimalik maksmata näha, siis oli näha, et rakendus tahab päris palju kasutajapoolset sisendit. Joonisel 1 on näha näidet i-Plant Nutrition rakenduse ühest vormist, milles kasutaja peaks sisestama andmeid labori kohta.

[1] I-Plant Nutrition https://i-plantnutrition.com/



Joonis 1: Näide ühest vormist I-Plant Nutrition rakenduses.

Kokkuvõttes pakub i-Plant Nutrition kasutajale põhjalikku ülevaadet oma põldude/farmide kohta ning aitab põllumehel optimeerida põlluharimist, kuid samas näiteks eeldab see rakendus seda, et kasutaja ise otsiks oma põllu mullakoostise analüüside jaoks mingisuguse labori üles, pärast mida teenusepakkuja agronoomid tõlgendavad kasutajale mõõtmisetulemusi ning loovad talle vastavad põlluharimisplaanid.

2.2 AgroCares

Võrreldes eelneva ettevõttega, pakub AgroCares[1] lisaks rakendusele ka enda loodud analüsaatorit, millega on võimalik kasutajal taimede kasvatamiseks olulisi mullaparameetreid, näiteks mulla happelisust (pH taset) või lämmastiku sisaldust, mõõta ning analüüsida. Mõõtmistulemusi saab kasutaja näha mobiilirakendusest, kus need on tehtud tavakasutajale lihtsasti arusaadavaks. Joonisel 2 on näha näidet AgroCares rakenduse mullaanalüüsi tulemuste kuva. Oma olemuselt seisneb rakendus

Joonis 2: AgroCares rakenduse mullaanalüüsi tulemuste kuva.

[1] AgroCares https://www.agrocares.com/

selles, et kasutajale ülevaate oma põllu mullastiku seisundist.

Esmasel vaatlusel tundub, et rakenduse tugevaks küljeks ongi see, et kasutaja saab ise mulla analüüse teha ning seejärel rakendusest tulemusi vaadata ning omad järeldused teha, mitte nagu eelneva rakenduse, I-Plant Nutrition-i puhul, kus kasutaja pidi ise oma asukohariigis labori otsima, mis mullaanalüüse teeks ning siis pärast need tulemused ise veebirakendusse sisestama, pärast mida teenusepakkuja agronoomid loovad kasutajale põlluharimisplaanid.

Antud rakenduse nõrkuseks võib välja tuua selle, et rakendus pole üleilmselt kasutatav, seda saab vaid ligikaudu kahekümnes riigis kasutada, põhjuseks see, et teenusepakkuja peab kalibreerima oma rakendust, et see arvestaks erinevate riikide mullastike spetsiifikaga.

Jällegi kahjuks ei olnud võimalik antud teenusepakkuja lahendust täpsemalt uurida, kuna mobiilirakendus eeldab litsentsi olemasolu, mis maksab 1700 € aastas. Lisaks oleks teenuse kasutamiseks vaja soetada ka AgroCares-i loodud skänner, mis siis ühildub mobiilirakendusega.

Kokkuvõttes tundub, et AgroCares-i rakendus on suhteliselt kasutajasõbralik, selles mõttes, et kasutaja saab ise oma mullaproove võtta, mitte ei pea pöörduma kolmanda osapoole juurde ning saab mõõtmistulemusi rakenduses lihtsasti arusaadaval kujul näha.

3 Analüüs

Analüüsi tehes said loodud ka nõuded rakendusele, millega saab tehtud töö tulemusi valideerida. Lisaks sai uuritud, kuidas Maa-ameti informatsiooni kaardile kuvada ning võrreldud erinevaid kaardi teeke.

3.1 Funktsionaalsed nõuded

Rakendusele esitatud funktsionaalsed nõuded on järgnevad:

- Rakenduse kasutaja saab kaardi peal luua endale põllu. Põldu saab luua kasutades joonestamisfunktsionaalsust, millega on võimalik kasutajal kaardile hulknurki joonestada.

- Kasutaja saab muuta oma põllu üksikasju (nimi, kirjeldus, seal kasvavad taimed, kasutatav väetis). Lisaks on kasutajal võimalik enda loodud põlde kustutada.

- Kasutaja näeb kaardi peal oma tehtud mullaanalüüse (mobiilirakendusega tehtud). Mullakoostise analüüse on kasutajal võimalik seostada konkreetse põlluga.

- Kasutajal peab olema võimalik näha nimekirja oma põldudest, et tal oleks lõplik ülevaade.

- Iga põllu detailvaates peavad kajastuma antud põlluga seonduvate mullanalüüside tulemused (graafiku kujul).

- Kasutaja saab erinevate riikide puhul kasutada võimalikult täpseid kaardiandmeid. Antud töös realiseeritakse see Eesti Vabariigi puhul, kasutades Maa-ameti andmeid.

3.2 Andmete saamine Maa-ametilt

Antud lõputöö kontekstis, et oleks võimalik Eesti kohta saada võimalikult täpseid kaardiandmeid, tuleks kasutusele võtta Maa-ameti[1] andmed ning need inkorporeerida rakenduse baaskaardiga.

Maa-ametilt on võimalik andmeid saada avalike API-de kaudu. Üldjuhul on tegemist siis kas WMS, WFS või TMS teenustega. APIde kaudu on võimalik saada palju erinevaid kaardikihte, näiteks saab Maa-ameti WMS teenuse kaudu Eesti kontekstis põhikaardi kihi, ortofotode kihi ja palju muid kihte veel. Antud lõputöö puhul pakub eriti huvi võimalus saada kaardikiht, milles on kujutatud põllumajanduslikud maad.

Siinkohal tuleb mainida, et Maa-amet soovitab aluskaardi kuvamiseks vältida WMS teenuse kasutamist, kui kaardi rakenduses kasutatakse näiteks Google Maps[2]-i, OpenLayers[3]-it või Leaflet[4]-i, kuna see põhjustab Maa-ameti WMS serverile suurt koormust, pärides kaardipilte 256 x 256 piksli suuruste osadena, mitte ühte suurt kaardipilti. [3]

Uurides veel Maa-ameti poolt loodud näiteid ja API-de kirjeldusi, sai veel selgeks, et Maa-ameti teenuseid kasutades on vaja, et aluskaardil oleks õige kaardiprojektsioon. Nagu teada, siis Maa on kera kujuline, kuid samas on kaardid kahedimensionaalsed. Sellega kaasneb see, et kui kaardistada Maad, siis on võimatu täiesti täpselt kujutleda Maad kaardil. [4] Seega on loodud erinevaid kaardiprojektsioone ja koordinaatsüsteeme, mis on teatud alade kohta võimalikult täpsed, Eesti puhul on kasutusel tasapinnaline ristkoordinaatide süsteem L-EST97, mille EPSG kood on 3301. [5] See tagab selle, et kaart oleks võimalikult täpne Eesti kontekstis, kuid kuskil mujal võib see näiteks riikide kujusid moonutada või midagi taolist.

---

[1] Maa-amet https://maaamet.ee/

[2] Google Maps https://developers.google.com/maps

[3] OpenLayers https://openlayers.org/

[4] Leaflet https://leafletjs.com/

3.3 Kaardi kuvamise võimalused rakenduses

Kaartide kasutamine igasugustes rakendustes on aegamööda läinud üha populaarsemaks ja populaarsemaks. Paljudele tuleb vast esimese asjana sõna kaart peale ette Google Maps. Olemas on ka teisi raamistikke, millega saab oma rakendusse kaardi inkorporeerida. Antud peatükis vaadeldakse erinevaid variante ning nende tugevusi ning nõrkusi. Olgu öeldud, et Google Maps-i API-t ei siin kontekstis ei lahata, kuna selle kasutamine alates teatud päringute arvust muutub tasuliseks. Baaskaardiks on valitud OpenStreetMap[1], kuna tegemist ühe suurima ja põhjalikuma vabalt kasutatava kaardiga. Lisaks on siinkohal oluline, et oleks võimalik andmeid Maa-ametilt kaardil suhteliselt valutult kuvada.

3.3.1 OpenLayers

Openlayers on avatud lähtekoodiga teek, millega on võimalik arendajal veebirakendusele lisada dünaamiline kaart ning seda täiendada. Kaardi põhi on realiseeritud OpenStreetMap-le tuginedes. Antud teek annab kasutajale suure vabaduse, kuidas kaarti realiseerida ning on nii-öelda hästi muljutav, vastavalt arendamisvajadustele. Lisaks on sellel teegil olemas tugi kaardiprojektsioonide jaoks. Antud teegil on olemas ka põhjalik dokumentatsioon, mille baasil saab arendada. OpenLayers on üle dekaadi vana, seega on sellel väga korralik ökosüsteem tekkinud aja pikku. Kahtlemata on sellel teegil üpriski suur õpikõver, kuid samas, on jällegi võimalik sellega palju korda saata.

3.3.2 Leaflet

Leaflet on samuti avatud lähtekoodiga JavaScripti teek, millega on arendajal võimalik dünaamilise kaarte veebirakendustele lisada. Võrreldes OpenLayers-iga on tegemist nii-öelda kergemini kasutatava teegiga, kuid samas ei paku Leaflet kõike, mida OpenLayers-iga on võimalik teha. Selle jaoks peab veel eraldi pistikprogramme kasutama. Kindlasti saab antud teegiga ka komplekssemaid kaardirakendusi luua, kuid eelkõige on see mõeldud lihtsakoelimaste kaartide loomiseks.

[1] OpenStreetMap https://www.openstreetmap.org/

Kokkuvõtteks langes lõputöö käigus valmiva rakenduse kaardi teegi valik OpenLayers-i kasuks. Kuigi ka Leaflet-iga saab erinevaid kujundeid kaardile joonestada on OpenLayers antud töö autori arvetes parem valik, kuna OpenLayers-iga on võimalik kaardiprojektsioone muuta, mis on selle töö raames vajalik, kui on tarvis kasutada Eesti Maa-ameti andmeid. Lisaks saab veel silmas pidada seda, et juhul, kui tulevikus on tarvis kaardirakendust täiustada mingisuguse kompleksse funktsiooniga, siis OpenLayers võimaldab seda vast lihtsamini realiseerida kui Leaflet.

4 Rakenduse realiseerimine

Töö loomisel sai lähtutud sellest, et rakenduse loomisel kasutatakse kaasaegseid tehnoloogiad, häid tavasid, ning et rakenduse arendusprotsess oleks võimalikult valutu, sealhulgas ka teistele arendajatele.

4.1 Arhitektuur

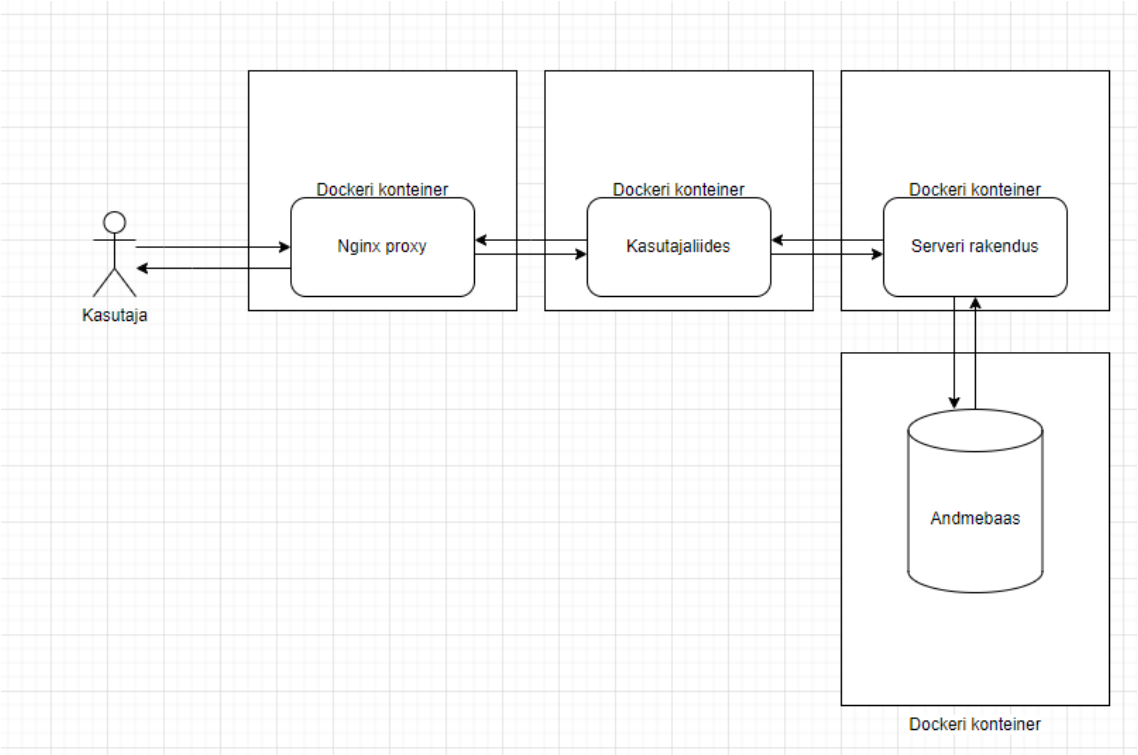
Antud lõputöö käigus valminud veebirakenduse arhitektuur on kolmekihiline (ingl k. m*odel-view-controller*), kus andmete muundamise loogika ning andmed ise on eraldi hoitud kasutajaliidese ja serverirakenduse vahel. Võib öelda, et kogu olemasolev süsteem, kaasaarvatud, antud töö käigus loodud rakendus, põhineb mikroteenuste arhitektuuril, kuna komponendid töötavad üksteisest eraldiseisvalt. Näiteks pärib lõputöö käigus tehtud rakendus mullaanalüüsi andmeid teisest Java serverirakendusest. Loodud rakenduse arhitektuuri visuaalset representatsiooni on võimalik näha Joonisel 3. Rakendus ise koosneb neljast komponendist:

- Kasutajaliides
- Serverirakendus
- Andmebaas
- Nginx[1] proksiserver

Siinkohal tuleb mainida, et andmebaasi töö käigus looma ei pidanud, see on juba realiseeritud teiste projektide käigus. Antud töö raames tuli seda vaid täiendada, lisades sinna uue skeemi ning tabelid loodava rakenduse jaoks. Nginx proksiserveril tuli ainult muuta konfiguratsioon enda vajadustele vastavaks.

Kuna andmebaas on ühine mitmel rakendusel, siis ei pidanud vajalikuks ka kasutajate registreerimise funktsionaalsust luua, kuna eelnevates projektides on see juba realiseeritud, samas sisselogimine oli ikkagi vaja realiseerida.

[1] Nginx https://www.nginx.com/



Joonis 3: Loodud rakenduse arhitektuur

4.2 Kasutajaliides

Antud töö käigus realiseeritud rakenduse kasutajaliides on oma olemuselt üheleherakendus (ingl k. *single-page application*). Üheleherakendused on oma olemuselt vägagi modulaarsed. Üldjuhul koosnevad need komponentidest, igaühel neist on oma roll, see oluliselt lihtsustab arendustööd. Samas on üldjuhul üheleherakendused kliendi poolel renderdatud,(ingl k. *client-side rendering*), mistõttu võib lehe laadimine algselt rohkem aega võtta. [6]

4.2.1 Kasutajaliidese raamistik

Kasutajaliidese loomiseks sai valitud Angular[1]-i raamistik, kuna tegemist on suhteliselt robustse raamistikuga, mida arendab Google. Angular-i üheks suurimaks tugevuseks on see, et see on kirjutatud kasutades TypeScript[2]-i ning selle kasutamine on Angulari raamistikku kasutades lausa rangelt soovituslik.

[1] Angular https://angular.io/

[2] TypeScript https://www.typescriptlang.org/

TypeScript on JavaScripti edasiarendus, mis lisab JavaScriptile moodulite süsteemi, klassid, liidesed ning staatilised tüübid. Põhirõhk TypeScript-il on arendamise võimalikult lihtsaks ja nii-öelda ohutuks muuta. Näiteks tüüpide olemasolu võimaldab staatiliselt juba koodis vigu leida. [7]

Üldjuhul koosneb rakendus komponentidest, need omakorda tavaliselt:

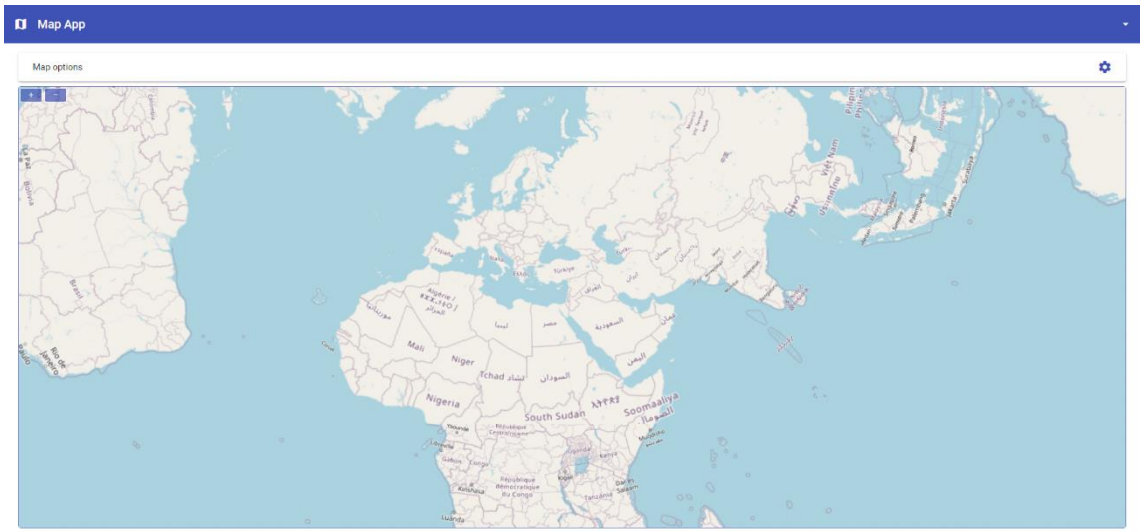
- HTML mallist
- TypeScript koodist, kus on komponendi funktsionaalsus realiseeritud.
- CSS stiilifailist, milles olevaid klasse saab antud komponent kasutada.

4.2.2 Kaardi kuvamine kasutajaliideses

Nagu eelnevalt on mitu korda mainitud, siis antud töö käigus loodava rakenduse üks põhifunktsionaalsusi on kaart, ning et kasutajal oleks võimalik sellele põlde joonistada.

Hea tava on komponendist eraldada suuremahulisem loogika, komponent on pigem mõeldud rohkem vaatena, seega sai loodud eraldi kaardi teenus, milles on realiseeritud suurem osa OpenLayers-i teegi loogikast.

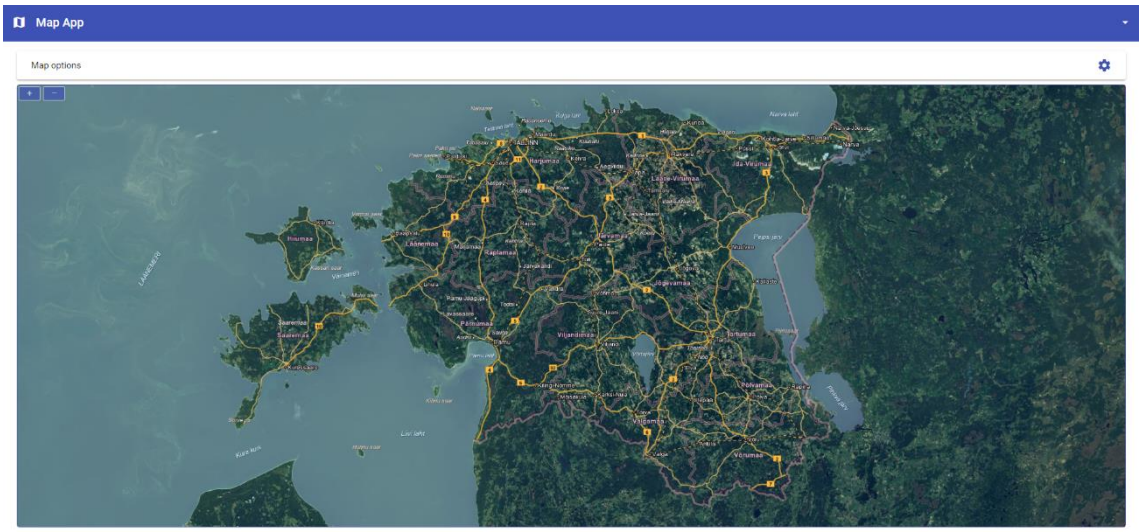
Üks problemaatilisem osa kasutajaliidese loomisel oli Maa-ameti kaardikihi kuvamine baaskaardile. Probleem seisnes selles, et kasutatav baaskaart, OpenStreetMaps-i kaart kasutab teist koordinaatsüsteemi, kui Maa-ameti kaardikiht. Nagu eelnevalt mainitud, siis Maa-ametilt saadavad kaardikihid kasutavad EPSG: 3301 projektsiooni, samas OpenStreetMaps-i baaskaart kasutab EPSG: 3857 projektsiooni. Joonisel 4 on võimalik näha, mis juhtub kui kasutada OpenStreetMaps-i kaarti EPSG: 3301 projektsiooniga.



Joonis 4: Rakenduse kasutajaliides maailmakaardiga kasutades EPSG:3301 projektsiooni

Nagu näha, siis Euroopa ala tundub enam vähem normaalne ning erilisi moonutusi ei ilmne, kuid probleemid algavad, mida kaugemale kaardil Eestist minna. Lisaks ka selline lahendus ei sobinud, et kaardil jätta projektsioon samaks, nagu baaskaardil, kuna siis ei laadinud Maa-ameti kaardikihid ära.

Lahenduseks antud probleemile tuli dünaamiliselt muuta kaardiprojektsiooni, vastavalt sellele, kas kasutaja nähtava kaardi sees on Eesti Vabariik või mitte. Täpsemalt tähendab see seda, et kasutaja peab kaardil olema piisava suurendusega, et suuremal osal ta kaardist oleks kuvatud Eesti. Joonisel 5 on näha ligikaudselt kaardi suurendust ning asukohta, kui kaardiprojektsioon muutub.



Joonis 5: Rakenduse kaart pärast projektsiooni muutumist EPSG:3301 peale.

Antud funktsionaalsus töötab ka vastupidi, kui kasutaja kaardil pole enam tervenisti Eesti Vabariik sees, siis muutub automaatselt kaardiprojektsioon tagasi EPSG:3857 peale, et maailma kaart oleks võimalikult optimaalne.

4.2.3 Kasutajaliidese kujundamine

Angular raamistikule on palju teeke, millega on võimalik rakendada valmis komponente, antud töö käigus langetati valik Angular Material[1]-i kasuks, kuna teek pakub suurel hulgal palju komponente, mida on arendaja mõistes suhteliselt lihtne implementeerida. Lisaks on paljud komponendid antud teegis päris suurel hulgal kohandatavad, seega on võimalik arendajal üpriski komplekssseid lahendusi välja töötada.

Kasutajaliidese komponentide küljendamisel pole kasutatud üldlevinud CSS-i vaid on kasutatud SASS[2]-i (ingl k *Syntactically Awesome Style Sheets*). Tegemist on CSS-i eeltöötlejaga, mis lõpuks muudab loodud koodi ikkagi CSS-iks. SASS-i süntaks on võrreldes CSS-iga lihtsam, arendaja ei pea sulgusid ja komasid kasutama. Lisaks veel võimaldab näiteks SASS muutujate kasutamist, midagi, mida tavalise CSS-iga teha ei saa. [8]

[1] Angular Material https://material.angular.io/

[2] SASS https://sass-lang.com/

4.3 Serverirakendus

Ehkki suurem osa rakenduse funktsionaalsusest on realiseeritud kasutajaliideses koos kaardiga, siis ikkagi on ka serverirakendusel oma roll, näiteks kasutajate autentimisel või andmete salvestamisel andmebaasi.

Serverirakendus on realiseeritud Java[1] programmeerimiskeeles, kuna antud programmeerimiskeelega on autoril kõige suurem kogemus. Rakendus on arendatud Java versiooniga 17.

4.3.1 Spring Boot ja Gradle

Serverirakenduse loomisel kasutati Spring Boot[2]-i. Tegemist on raamistikuga, mis oluliselt hõlbustab ja kiirendab serverirakenduste arendamist Java programmeerimiskeeles. Spring Boot rakenduse loomine käib üldjuhul spring initializr[3] veebilehe kaudu. Antud lehel on võimalik Springi projekti nimi määrata, valida, millist kooste tööriista kasutada (Maven[4], Gradle[5]) ning millistest välistest teekidest projekt sõltub, näiteks Spring Security[6].Antud veebileht genereerib projekti baasi zip failina, mida saab hõlpsalt arendustegevuse alustamiseks kasutada.

Gradle-it kasutatakse antud serverirakenduses väliste teekide haldamiseks ning rakenduse ehitamiseks. Gradle on avatud lähtekoodiga kooste automatiseerimise tööriist, mis baseerub ülesannetel (ingl k. *task*). Gradle modelleerib rakenduse ehitamisprotsesse suunatud atsükliliste graafidena. Kokkuvõttes tähendab see seda, et Gradle saab aru, kuidas erinevad ülesanded üksteisest sõltuvad ning käivitab neid vastavas järjekorras. [9]

[1] Java https://www.java.com/en/

[2] Spring Boot https://spring.io/projects/spring-boot

[3] spring initializr https://start.spring.io/

[4] Maven https://maven.apache.org/

[5] Gradle https://gradle.org/

[6] Spring Security https://spring.io/projects/spring-security

4.3.2 Serverirakenduse turvalisus

Serverirakenduse turvalisemaks tegemisel mängib suur rolli Spring Security teek. Rakenduse loomisest peale on Spring Security automaatselt konfigureeritud, kuid seda tuli muuta, et serverirakendus vastaks enda vajadustele.

Ehkki antud rakendus ei võimalda uute kasutajate registreerimist, see on juba realiseeritud eelnevas projektis, siis rakendus peab ikkagi isiku ära autentima. Isiku õnnestunud autentimisel tagastatakse HTTP vastuse päises ’*Authorization*’ väljal JWT, mille kasutajaliides salvestab lokaalsesse hoidlasse (ingl k. *local storage*). Serverirakenduse ainus autentimist mitte vajav lõppsõlm (ingl k *endpoint*) ongi see sama, mida kasutatakse kasutajaliidesesse sisse logimisel.

Lisaks veel on rakenduse loomisel lähtutud põhimõttest, et kasutajatel on võimalik ainult endaga seotud andmeid näha. Igal lõppsõlmel, mis on seotud kas andmete kuvamise, muutmise, loomise või kustutamisega, on kontroll peal, et kasutaja, kes näiteks soovib antud rakenduse kontekstis mingisugust põldu kustutada, on ka selle põllu ise enne loonud.

Serverirakendus kasutab veel CORS konfiguratsiooni, mistõttu brauserid saavad teha päringuid serverirakenduse pihta ainult kasutajaliidese aadressilt.

4.3.3 Andmebaasiga seonduv

Kõik andmebaasiga seonduv on realiseeritud serverirakenduses. Nagu eelnevalt mainitud, siis antud bakalaureusetöö raames ei pidanud looma uut andmebaasi, olemasolev PostgreSQL[1]-i baas oli juba olemas. Andmetega töötamiseks kasutab serverirakendus Hibernate[2]-i teeke ja JPA liidest.

Hibernate on avatud lähtekoodiga ORM tööriist, millega on võimalik hõlpsalt kaardistada näiteks Java objekt ümber domeeni mudeliks relatsioonilises andmebaasis. Hibernate-i kasutamine kahtlemata lihtsustab objektide salvestamist andmebaasi ning andmebaasist tuleva info ümberkaardistamist objektina, mis tõttu ka arendusele kuluv aeg tõenäoliselt väheneb. [10]

Andmebaasi muudatusete haldamiseks ning jälgimiseks on rakenduses kasutusel Liquibase[3]. Liquibase võimaldab arendajatel andmebaasimuudatusetel järge pidada. Liquibase-l on üks põhiline muudatusete (ingl k. *changeset*) fail, tegemist on siis XML, SQL, JSON või YAML failiga. Antud töö kontekstis on realiseeritud Liquibase kasutus nõnda, et on olemas üks peamine *master.yaml* muudatuste fail, milles on kirjeldatud teiste muudatuste (*changeset*) muudatused. Viimase puhul on tegemist SQL failidega, mis käivitatakse Liquibase poolt, kui need pole juba käivitatud. Liquibase peab selle üle järge tabelis nimega *databasechangelog*.

Nagu eelnevalt sai mainitud, kuna mitu rakendust kasutab sama andmebaasi, siis näiteks kasutajate leidmiseks ja sisselogimiseks sai tehtud eraldi vaade lõputöö käigus realiseeritava rakenduse andmebaasiskeemi. Uutele andmetüüpidele, näiteks põld, sai loodud uued tabelid rakenduse enda skeemi, samas andmeid proovide kohta päriti otse teisest skeemist. Joonisel 6 on näha näidet põllu tabelist kahes osas.

[1] PostgreSQL https://www.postgresql.org/

[2] Hibernate https://hibernate.org/

[3] Liquibase https://www.liquibase.org/

	id [PK] bigint	user_id bigint	coordinate_sys character varying (50)	name character varying (50)
1	1	761	EPSG:3857	Läti põld
2	2	761	EPSG:3301	Männiku põld
3	3	761	EPSG:3301	New field
4	4	761	EPSG:3857	aafrika põld
5	5	761	EPSG:3301	Liikuri põld

geo_json jsonb
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[{"latitude": 7773910.349790208, "longtitude": 2864778.1418648968}, {"latitude": 7774261.22292756, "lo
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[{"latitude": 6577582.850209422, "longtitude": 540780.7658111834}, {"latitude": 6577541.791835294, "lo
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[{"latitude": 6564731.081752546, "longtitude": 569189.9159640598}, {"latitude": 6577234.313915039, "lo
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[{"latitude": 761606.6519431202, "longtitude": -3390962.9503181763}, {"latitude": 3699232.3094380135,
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[{"latitude": 6589937.071737543, "longtitude": 546531.2605462043}, {"latitude": 6590219.9827741, "longt

Joonis 6: Näide salvestatud põldudest andmebaasis (loetavuse huvides on joonis poolitatud)

Üks huvitav asjaolu kerkis esile Hibernate-i teegi kasutamisega. Kuna põllu geograafilised andmed on salvestatud GeoJSON-i formaadis, et seda oleks võimalik lihtsalt pärast andmebaasist pärimist taasesitada kasutajaliideses, siis tuli geo_json väljale tabelis *field* panna *jsonb* tüüp. Hibernate-i teek ise automaatselt ei suutnud geo_jsoni objekti õigesti andmebaasi salvestada, seega tuli kasutada veel ühte teeki (Hibernate Types 52), mis lisas Hibernate-le tüübi toe juurde.

4.3.4 Serverirakenduse testimine

Serverirakenduse testimiseks on loodud ühik ja integratsioonitestid, valideerimaks, et rakendus funktsioneerib nagu plaanitud. Ühiktestide kirjutamiseks on kasutatud JUnit[1]

[1] JUnit https://junit.org/junit5/

teeki, ühte levinumatest testimisteekidest. Lisaks on kasutatud Mockito[1]-t, millega ühiktestides jäljendada näiteks mingeid teenuseid, millest testi all olev klass sõltub.

Integratsioonitestide pool on lahendatud järgnevalt:

Testcontainers[2]i teegiga on luuakse Docker[3]i konteiner, milles luuakse rakenduse andmebaas valitud SQL skriptidega, mida Liquibase jooksutab. Andmebaasimootoriks on nagu ka pärisrakenduses PostgreSQL. See tagab võimalikult autentse testimiskeskkonna. Variant oleks ka mingisugust mälupõhist andmebaasi kasutada, et oleks testid jookseksid võimalikult kiirelt, kuid siis võib tekkida anomaaliaid erinevate SQL-i dialektide vahel. Variant on ka see, et integratsiooni teste harvem jooksutada, näiteks rakenduse ehitamisel antud lõputöö kontekstis Gitlab[4]-i CI/CD-ga.

4.4 Docker ja pidevkooste

4.4.1 Docker

Docker on avatud lähtekoodiga platvorm, mille eesmärk on hõlbustada rakenduste arendamist ja tarnimist. Docker-i peamiseks nii-öelda osaks võib kutsuda konteinerit. Tegemist on kerge virtuaalmasinaga. Konteineri ehitamisel pakitakse sinna rakendus koos kõigi vajalike sõltuvustega. Loogika seisneb selles, et rakendust saab alati samasugustes tingimustes jooksutada, Docker-i konteiner on isoleeritud jooksutava masina operatsioonisüsteemist, seega on võimalik konteineriseeritud rakendusi suhteliselt kerge vaevaga erinevate masinate peal jooksutada. [11]

[1] Mockito https://site.mockito.org/

[2] Testcontainers https://www.testcontainers.org/

[3] Docker https://www.docker.com/

[4] Gitlab https://about.gitlab.com/

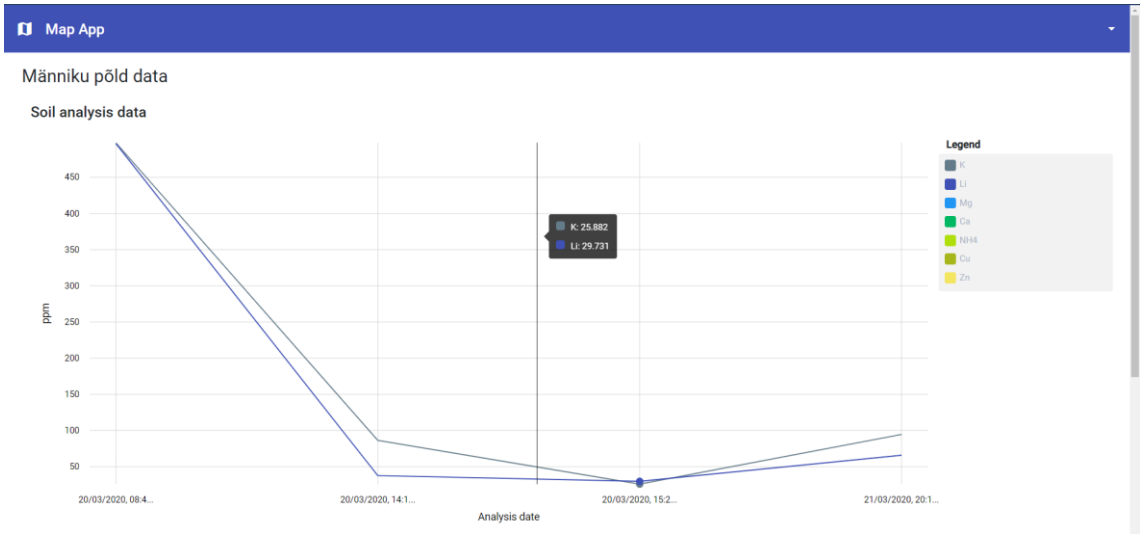
Antud lõputöö raames on loodud serverirakendus kui ka kasutajaliides, mõlemad, Docker-i konteineritesse pandud. Lisaks nendele on lisatud ka eraldi konteineris Nginx proksiserver. Antud konteinerite jaoks on loodud eraldi *Dockerfile* skriptid, mille põhjal konteinereid luuakse. Kogu loodava rakenduse käivitamise hõlbustamiseks on ka eraldi *docker-compose.yml* skript loodud. Docker-compose skriptiga on võimalik kõik kolm konteinerit korraga käivitada, kasutades ühte käsurea käsku: *docker-compose up*, mitte kolme erinevat koos mitmete parameetritega.

4.4.2 Pidevkooste

Pidevkooste osas on antud rakenduse loomisel kasutatud Gitlab-i CI/CD võimalusi. Gitlab-i CI/CD seadistamine on suhteliselt lihtne. Projekti põhikausta tuleb luua skripti fail nimega *gitlab-ci.yml*. Faili on võimalik läbi Gitlab-i kasutajaliidese luua, või siis ükskõik millises tekstiredaktoris. Esmalt tuleb failis ära kirjeldada etapid, näiteks ehitamine (ingl k. *build*). Siis saab täpsemalt iga etapi lahti kirjutada, see tähendab, missuguseid skripte antud etapis peaks Gitlab-i jooksutaja (ingl k. *runner*) käivitama. Kuna antud töö raames on juba nii kasutajaliidesele kui ka serverirakendusele eraldi *Dockerfile*-id loodud, siis piisab ainult näiteks järgnevast käsust, et serverirakendus ehitada ning testida: „*cd back && docker build . -t map-app-back“* . Kui testid ebaõnnestuvad, siis Docker-i konteineri ehitamine ebaõnnestub samuti, mistõttu terve rakenduse ehitamine ebaõnnestub. Juhul kui kõik testid on läbitud ning rakenduse ehitamine õnnestub, siis järgnev samm on rakenduse automaatne paigaldamine (ingl k. *deployment*), seda juhul kui muudatused on tehtud projekti peamisesse Git-i harusse (*main*).

5 Töö tulemuste analüüs

Tehtud töö käigus said kõik püstitatud eesmärgid täidetud, loodi rakendus millega kasutaja saab enda põldusid kaardile märkida ning põllul tehtud mullakoostise mõõtmisi analüüsida. Mullakoostise analüüsi tulemuste graafikut on võimalik näha joonisel 7.



Joonis 7: Mullaanalüüsi tulemuste graafik ühe põllu kohta (4 analüüsi)

Samas tuleb tõdeda, et arendus oleks võinud olla struktureeritum, nagu tiimis arendades, kus arendusi tehakse näiteks eraldi *feature* harudes ning pärast selle valmimist lisatakse (ingl k. *merge*) peamisesse harusse, millest tehakse tarne. Võib-olla oleks seda probleemi leevendanud see, kui CI/CD oleks ühe esimese asjana antud rakenduse puhul käima saanud.

Ilmselt oleks võinud ise ka olla rohkem proaktiivsem ja rohkem teha, siis oleks vast edasiste arenduste nimekiri lühem, kuid sellest hoolimata sai kokkuvõttes üpriski hea rakenduse loodud, nagu ülevalpool sai mainitud, siis püstitatud eesmärgid said täidetud.

5.1 Edasised arendused

Ehkki antud bakalaureusetöös sätestatud eesmärgid said täidetud, saaks antud rakendust paljugi paremaks teha.

- Üks variant oleks lisada rakendusele masinõpe, mis aitaks kasutajal teha väetamise plaane või siis üleüldiseid soovitusi põlluharimise kohta, kasutades selleks olemasolevaid andmeid (mullaanalüüsi tulemused, põllul kasvavad taimed, kasutatav väetis).

- Lisaks veel oleks võimalik rohkem andmeid lisada iga põllu kohta, näiteks oleks võimalik informatsiooni ilmastiku kohta pärida. See informatsioon võiks rakendusel aidata näiteks paremaid väetamiseplaane luua.

- Kaarti saaks veel täiustada, erinevate riikide Maa-ameti analoogidelt võiks samamoodi nagu Eesti puhul on tehtud, täpsemaid kaardiandmeid saada.

- Kõigele lisaks võib veel kaarti täiendada näiteks satelliitide andmetega (Sentinel). Sealt oleks võimalik saada informatsiooni näiteks vegetatsiooni kohta, mida võiks kuidagi rakendada.

6 Kokkuvõte

Antud lõputöö eesmärgiks oli luua veebirakendus, millega kasutajal on võimalik luua kaardi peal enda põlde. Põldudele pidi olema võimalik lisada mullaanalüüsi tulemusi, mida kasutaja oli eelnevalt võtnud. Lisaks pidi rakendus andma kasutajale ülevaate põllu mullakoostise muutumisest ajas.

Sõnastatud tulemuse saamiseks, sai uuritud nõudeid, millele loodav rakendus peaks vastama, lisaks sai uuritud olemasolevaid täppisväetamise lahendusi, et oleks hea ülevaade sarnastest lahendustest ning et oleks mingil määral arusaadav milliseid lahendusi võiks kasutada ja milliseid mitte.

Kuna rakenduse peamiseks funktsionaalsuseks on kaart ja selle modifitseerimine, siis sai veel uuritud milliseid kaarditeeke oleks mõistlik kasutada. Lisaks veel, kuna üheks nõudeks oli täpsete kaardiandmete kasutamine, siis sai realiseeritud see, et kaardiandmed Eesti Vabariigi territooriumi kohal tulevad Maa-ametilt.

Töö tulemusena valmis kolmekihilisel arhitektuuril baseeruv rakendus, mis koosneb serverirakendusest, kasutajaliidesest ja proksiserverist. Valminud rakendust on võimalik lihtsasti paigaldada, kuna on olemas vajalikud skriptid, et rakendus Docker-i konteinerites käima panna. Lisaks kasutab rakendus CI/CD-d, seda läbi Gitlab-i.

Edasiarendusena võiks rakendusele lisada vast tehisintellekt, mis analüüsiks antud mullakoostise mõõtmistulemusi ning põllul kasvavaid taimi ja selle informatsiooni põhjal looks kasutajale väetamisplaani või annaks üldisemaid soovitusi.

Kasutatud kirjandus

[1] G. A. B. H. P. Good, „Fertilizing Nature: A Tragedy of Excess in the Commons,“ *PLOS Biology,* nr 8, 2011.

[2] T. Polansek ja A. Mano, „As sanctions bite Russia, fertilizer shortage imperils world food supply,“ *REUTERS,* 24 märts 2022.

[3] Maa-amet, „Geoportaal: Maa-amet: Teenused,“ 25 märts 2021. [Võrgumaterjal]. Available: https://geoportaal.maaamet.ee/est/Teenused/WMSWFS-teenused/TMS-WMS-C-ja-WMTS-teenused-p481.html. [Kasutatud 6 mai 2022].

[4] K. A. Mulchay ja K. C. Clarke, „Symbolization of map projection distortion: a review.,“ *Cartography and geographic information science,* nr 3, pp. 167-182, 2001.

[5] Maa-amet, „Geoportaal: Ruumiandmed: Eesti geodeetiline süsteem: L-EST97,“ 14 juuni 2021. [Võrgumaterjal]. Available: https://geoportaal.maaamet.ee/est/Ruumiandmed/Geodeetilised-andmed/Eesti-geodeetiline-susteem/L-EST97-koordinaatsusteem-p173.html. [Kasutatud 6 mai 2022].

[6] V. Solovei, O. Olshevska ja B. Y, „The difference between developing single page application and traditional web application based on mechatronics robot laboratory onaft application.,“ *Automation of technological and business processes,* nr 10, 2018.

[7] G. Bierman, M. Abadi ja M. Torgersen, „Understanding typescript,“ %1 *European Conference on Object-Oriented Programming,* Berliin, 2014.

[8] M. D ja T. N, „An empirical study on the use of CSS preprocessors,“ %1 *IEEE 23rd international conference on Software Analysis, Evolution, and Reengineering (SANER),* 2016.

[9] Gradle Inc., „Gradle: Docs: Introduction: What is Gradle?,“ Gradle Inc., 2021. [Võrgumaterjal]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html. [Kasutatud 6 mai 2022].

[10] „Home: Open source strategy: Programming: Hibernate,“ [Võrgumaterjal]. Available: https://www.theserverside.com/definition/Hibernate. [Kasutatud 6 mai 2022].

[11] B. Rad, H. Bhatti ja M. Ahmadi, „An introduction to docker and analysis of its performance,“ *International Journal of Computer Science and Network Security (IJCSNS),* kd. 3, nr 17, p. 228, 2017.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Johann Veispak

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Täppisväetamise kasutajaliides“, mille juhendaja on Evelin Halling.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

06.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.