TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Anton Sauh 184986IADB

Teraviljakuivati jälgimissüsteemi arendus Siemens SIMATIC HMI baasil

Bakalaureusetöö

Juhendaja: Andres Käver
Magistrikraad
Jevgeni Družkov
Magistrikraad

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: Anton Sauh

04.01.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on välja töötada süsteem, mis on võimeline Siemens SIMATIC HMI PLC juhitavatelt viljakuivatitelt andmeid hankima, salvestama ja parsima, lisaks peaks süsteemil olema võimalus salvestatud andmeid visualiseerida.

Lõputöö analüüsi käigus uuriti olemasolevaid süsteeme ning arvukate faktide tõttu, nagu näiteks integreeritavus Siemensi SIMATIC HMI PLC-ga ja jälgitavate andmete hulk, otsustati kavandada ja juurutada MVP arenenum lahendus. Süsteemi töö põhineb andmetel, mida hangitakse PLC-st, mis ise ühendab arvukalt andureid üheks asjade interneti süsteemiks ning toimib teraviljakuivatisüsteemi tsentraliseeritud kontrolleri ja andmeväravana. Antud süsteemi kliendipoole eesmärk on anda vahetult viljakuivatusprotsessiga töötavatele operaatoritele tööriist, mis on võimeline neid abistama, minimeerides raisku mineva teravilja hulka.

Lisaks teravilja kuivatamise seiresüsteemi analüüsile ja arendamisele töötati välja automatiseeritud testimiskomplekt, mis koosneb ühiku- ja integratsioonitestidest, mis võimaldasid iteratiivse arendusprotsessi käigus regressiooni minimeerida.

Lõputöö on kirjutatud eesti keeles ja sisaldab 30 lehekülge teksti, 7 peatükki, 5 joonist ja 3 tabelit.

Abstract

Development of a Grain Dryer Monitoring System Based on Siemens SIMATIC HMI

The aim of this bachelor's thesis is to develop a system which is capable of fetching, storing and parsing the data from grain dryers based controlled by Siemens SIMATIC HMI PLC, in addition to that the system should have the ability to visualize the data stored.

During the analysis of the thesis, the existing systems were examined and due to the numerous facts, such as ability to be integrated with the Siemens SIMATIC HMI PLC and the amount of the data tracked, it was decided to design and implement an MVP of a more advanced solution. The operation of the system is based on the data, that is being fetched from the PLC, which itself connects numerous sensors into one IoT system and acts as a centralized controller and data gateway for the grain dryer system. The aim of the client side of the given system is to give operators working directly with the grain drying process a tool, which is capable of assisting them, minimizing the amount of grain going to waste.

In addition to the analysis and development of the grain drying monitoring system, automated testing suite was developed, which consists of unit and integration tests, which allowed to minimize the regression during iterative development process.

The thesis is written in Estonian language and contains 30 pages of text, 7 chapters, 5 figures and 3 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface, rakendusliides.*
CGI	*Common Gateway Interface, standard mis defineerib kuidas veebiserver suhtleb teiste programmidega.*
CPU	*Central Processing Unit, protsessor.*
CSV	*Comma separated values, andmete formaat.*
DDD	*Domain Driven Design, domeenipõhine disain.*
DSL	*Domain Specific Language, domeeni spetsiifiline keel.*
E2E	*End to End, lõpp-lõpuni.*
GB	*Gigabyte, gigabait.*
HTML	*Hypertext Markup Language, keel milles märgendatakse veebilehti.*
HTTP	*Hypertext Transfer Protocol, veebis kasutatav andmeedastusprotokoll.*
ID	*Unique identifier, unikaalne identifikaator.*
IoT	*Internet of Things, Asjade Internet.*
JSON	*JavaScript Object Notation, andmete formaat.*
MVC	*Model-View-Controller, tarkvara arhitektuurimuster.*
MVP	*Minimum Viable Product, minimaalne töötav toode.*
PLC	*Programmable logic controller, spetsialiseeritud tööstusarvuti kindla tööprotsessi juhtimiseks.*
RAM	*Random Access Memory, muutmälu.*
REST	*Representational State Transfer Protocol, tarkvaraarhitektuuri laad.*
SMS	*Short Message Service, lühisõnumiteenus.*
SSD	*Solid-state Drive, pooljuhtketas.*

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Teravilja kuivatamine on mitme astmeline protsess, milles peab jälgima erinevaid faktoreid igal etapil. Kuna teravilja õige niiskuse taseme vahemik on 11,5% kuni 13,5%, kuivatamise protsess peab olema hästi jälgitav ja töötaja peab omama võimalust erinevatele olukordadele reageerida. [1]

Osa põllumajandusettevõtteid on võtnud tööle uuemaid teraviljakuivati süsteeme, mis on digitaliseeritud, need annavad ettevõttele rohkem nähtavust kuivatusprotsessi kohta, kasutades temperatuuri, niiskuse, kiiruse ning teisi andureid, mis kõik on ühendatud ühte süsteemi. Nimetatud süsteemide puuduseks on suhteliselt kõrge soetusmaksumus, mistõttu iga ettevõte ei suuda digitaliseeritud süsteemi endale lubada. [2]

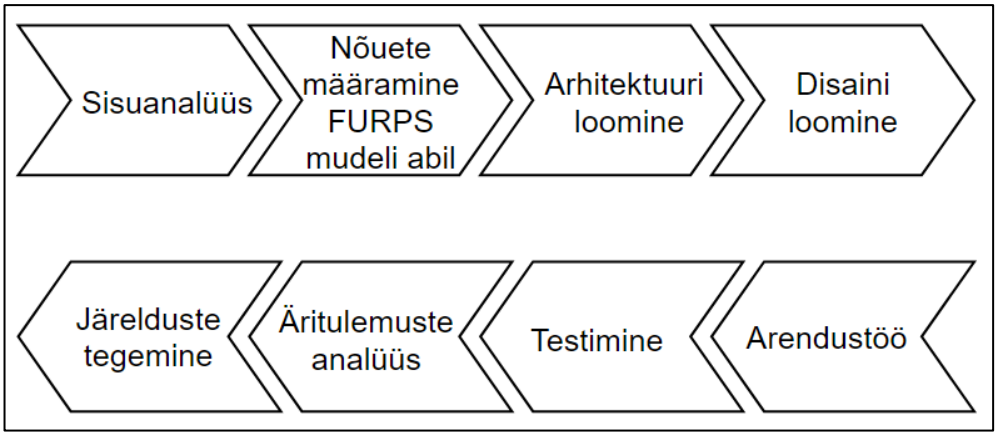
Uuemad teraviljakuivatid kasutavad kuivatusprogramme, mida operaator käivitab iga erineva vilja jaoks. Nimetatud programme laaditakse üles arvutisse, mis kontrollib kogu kuivati tööd. Selline kuivati digitaliseerimine annab võimaluse uute lahenduse otsimiseks ning arendamiseks, et muuta kuivatusprotsess efektiivsemaks.

Käesolevas lõputöös analüüsitakse probleemi, mille kohaselt puudub teraviljakuivati operaatoril võimalus mugavalt ning efektiivselt kuivatusprotsessi (kuivatusprogrammi töötamist) jälgida ning probleemidele reageerida. Teraviljakuivati ehitusliku arhitektuuri tõttu ei ole operaatoril võimalust kogu tööpäeva vältel jälgida süsteemi arvutist, kuna osa oma tööajast peab operaator veetma otseselt kuivatiga töötades. Lisaks sellele peab operaator suhteliselt lühikese ajaperioodi jooksul probleemile reageerima, et vili ei rikneks, seetõttu on tähtis arusaadava infograafika olemasolu. Tähtis on mainida, et iga teraviljakuivati on erinev ja iga operaator ei pruugi omada kogemust just selle kuivatiga, mis tekitab lisaraskusi tööpersonali planeerimisel. Autori erakogus olevate andmete kohaselt tekib nimetatud probleemi tõttu riknenud teravilja näol lisakulu, mille kogus võib ulatuda viie tonnini kuus.

Lõputöö eesmärgiks on arendada ja testida teraviljakuivati jälgimise tarkvara eesmärgiga pakkuda teraviljakuivati operaatoritele parimat võimalust kuivatuse protsessi jälgimiseks. See võimaldab operaatoril oma tööd kvaliteetsemalt teha ning vältida vigu, mis tekivad informatsiooni arusaamatuse tõttu või õigeaegse informatsiooni operaatorini mittejõudmisel.

1.1. Metoodika

Käesoleva lõputöös kasutatakse erinevaid andmekogumis- ja analüüsimeetodeid. Metoodika erinevaid etappe esitatakse joonisel 1. Andmekogumismeetodi näol on tegemist sisuanalüüsiga, mille alusel analüüsitakse erinevaid tekstiallikaid: tehnoloogiate käsiraamatud, teadusartiklid, raamatud, veebiallikad. Sisuanalüüsi tulemusel saadakse andmeid olemasolevate tehnoloogiate kohta, mille abil on võimalik lahendust paremini defineerida ja disainida. Nimetatud meetod sobib probleemi analüüsimiseks, kuna võimaldab samadel alustel erinevatest allikatest saadud informatsiooni omavahel võrrelda. [3]



Joonis 1. Lõputöö metoodika disain.

Andmeanalüüsi meetodite näol on tegemist:

1. Lahenduse nõuete analüüs kasutades FURPS mudelit [4].
2. Arhitektuuri ja disaini analüüs kasutades erinevaid projekteerimise meetodeid.
3. Ees- ja tagarakenduse arendamise tehnoloogia analüüs ja valimine.
4. Lahenduse testimiseks vajalikke meetodite ja tehnoloogiate analüüs ning valimine.
5. Äritulemuste analüüs ja arvutuste tegemine.
6. Lahenduse realiseeritud funktsionaalsuse analüüs ja järelduste tegemine.

2. Ülevaade probleemist

Teraviljakuivati operaatoril on omal tööl vajalik jälgida mitmeid erinevaid faktoreid, mis koos mõjutavad tulemuseks saadud teravilja kvaliteeti. Neid faktoreid ei mõjuta mitte ainult teravilja niiskus enne kuivatamist, vaid ka keskkonnategurid nagu näiteks õhutemperatuur ja selle muutmine ööpäeva jooksul, vihm, udu.

Operaatoreid õpetatakse, et teatud tingimustel peab teraviljakuivati parameetreid reguleerima teatud viisil: millal peab kuivati temperatuuri muutma, millal sisse- ja väljatuleva teravilja kogust muutma või millal peab kuivatist välja tulnud teravilja tagasi kuivatisse suunama. Kuid selles olukorras on kriitiline operaatori töökogemus, mida näiteks uute operaatorite puhul ei pruugi olla ja ka mitmenädalane väljaõpetamine ei anna operaatorile teadmisi kõikidest olukordadest, mis tuleb tööprotsessis lahendada.

Uuemad teraviljakuivatid pakuvad lahendust läbi IoT süsteemide, kus kõik kuivati andurid on ühendatud ühe arvuti taha, kuid reeglina need süsteemid näitavad ainult reaalajas andmeid ja operaatoril puudub võimalus vaadata, mida teine spetsialist sarnasel juhul teinud kas kaks nädalat või aasta tagasi. [2]

Lisaks sellele, kui kuivatil on tekkinud tehniline probleem, ei ole operaatoril võimalust samal ajal jälgida teraviljakuivati parameetreid. Operaator näeb alles tagasi arvuti taha tulles, kas tema tegevused on probleemi lahendanud või teinud seda ainult suuremaks. Teraviljakuivati äri omaniku kogemusel on reeglina operaatorina töötanud inimesed, kes ei ole professionaalsed arvutikasutajad ehk nende jaoks peab olema lihtne ja kasutamismugav liides, mis võimaldab seda kasutamist kiiresti õpetada.

Eeltoodud probleemi lahendamiseks uurib autor erinevaid võimalusi, mis aitaksid operaatoritele anda rohkem informatsiooni teraviljakuivatamise protsessi kohta mugaval ja kasutussõbralikul kujul.

2.1. Probleemi praktiline käsitlus

Eeltoodud probleemi saab lahendada võttes kasutusele infosüsteemi, mis võimaldaks salvestada, töödelda ning visualiseerida teraviljakuivatiga seotud informatsiooni nagu näiteks kuivatamisruumi temperatuur, sissetuleva teravilja niiskus ja väljamineva teravilja niiskus. Oluliseks osaks on arusaadava infograafika olemasolu ja süsteemi kasutatavus, kuna lahenduse kasutamisega peavad hakkama saama inimesed madala arvutioskuste tasemega.

Lahenduse oluliseks osaks võib pidada ka ligipääsu süsteemile portatiivsetest seadmetest, milleks võiks olla kas nutitelefon või tahvelarvuti. Mainitud süsteemi omadus on seotud sellega, et osa oma tööajast veedavad operaatorid väljaspool arvutiruumi ja ligipääs andmetele oleks spetsialistidele heaks toeks teraviljakuivatiga otseselt tööd tehes.

Teraviljakuivati parameetrite muutmine operaatori poolt peab olema jälgitav, selleks et hiljem oleks võimalus operaatori tegevusi teatud olukordades analüüsida, hinnata ning neist järeldusi teha. Lisaks sellele saab siduda mitmeid erinevaid andmeid läbi sisendparameetrite muutmise: näiteks kui kuivati töövõimu suurendatakse, siis kui kiiresti temperatuur kuivati sees suureneb ning kuidas see mõjutab väljamineva vilja kvaliteeti.

Süsteemil peab olema teavituste süsteem näiteks läbi nutiseadme teavituste või läbi muu kanali, heaks näiteks on tekstisõnumid või automaatne helistamine operaatori numbrile. Nimetatud süsteem võimaldaks erinevaid häireid seadistada, mis minimiseerivad inimveo võimalust. Süsteem võiks anda võimaluse seadistada administraatoril või vanemspetsialistil teraviljakuivati seadistuse piirmäärasid. Näiteks olukorras, kui operaator paneb juhuslikult kuivati flööni taseme liiga kõrgeks, siis saab ta selle kohta teavituse ja kohe parandab oma vea.

Lisaks võib selle ajaloolise informatsiooni baasil tulevikus ehitada lahendusi ka suurema autonoomsusega, mis ise oskavad operaatorile soovitusi pakkuda ja tema tähelepanu juhtida olulistele teraviljakuivatusprotsessi aspektidele. Kui süsteem oskaks arvesse võtta teraviljakuivati praeguseid parameetreid ning ennustada, milline tuleb väljamineva vilja niiskus praeguste parameetritega, oleks operaatoril võimalus kohe reageerida ning parandusi teha.

2.2. Olemasolevate süsteemide analüüs

2.2.1. Dryer Master DM100

Dryer Master on Kanada ettevõte, mis pakub erinevaid lahendusi teraviljakuivatusprotsessi jälgimiseks ning automatiseerimiseks. Ettevõte on juba 30 aastat tegelenud eeltoodud probleemi lahendamisega. Üks nende uuematest lahendustest on Dryer Master DM100, mis oskab teraviljakuivati tööd kontrollida läbi andurite ning integratsiooni teraviljakuivatiga. [5]

Standardlahendus koosneb kontrollpaneelist ja ühest niiskusandurist. Kontrollpaneelil on sisseehitatud veebiserver, mis võimaldab andmeid edastada ka näiteks nutitelefoni või tahvelarvutisse. Kontrollpaneeli on sisse ehitatud mälukaart, kuhu andmed salvestatakse ja hiljem on võimalus ajaloolisi andmeid salvestada kasutades mälupulka, mis sisestatakse kontrollpaneeli USB pesasse. [6]

Süsteem põhimõte on selles, et see oskab automaatselt reguleerida tühjenemiskiirust kasutades 3 kasutaja poolt seadistatud malli. Kuivema sissetuleva teravilja puhul suurendab süsteem automaatselt tühjenemiskiirust ning vastupidi.

Süsteemil saab seadistada häireid ja kasutajaliidesest näha, milline häire on praegu aktiivne. Lisaks selle süsteem annab võimalust vaadata 3 erinevat graafi, kui on tähtis mainida et graafi ajaline periood on väiksem kui 24 tundi. Süsteemi oluline omadus seisneb selles, et seda saab integreerida suurema osa digitaliseeritud teraviljakuivatitega erinevatelt tootjatelt, mis on autori arvates selle süsteemi suurim eelis.

Autor leiab, et selle süsteemi puudused on:

- Ajalooliste andmete salvestamine sisseehitatud mälukaardile, millelt tuleb andmeid manuaalselt alla laadida kasutades .csv formaadis mälupulka, mida hiljem tuleb käsitsi arvutisse laadida.
- Süsteemil on piiratud kogus andureid, millest see saab andmeid koguda.
    - Süsteem annab võimaluse temperatuuriandmeid koguda ainult ühest andurist.
    - Süsteem annab võimaluse niiskuseandmeid koguda ainult ühest andurist.

2.2.2. Mathews Company Trax

Mathews Company on Ameerika Ühendriikide ettevõte, mis spetsialiseerub teraviljakuivatite tootmisele ning pakub tooteid ja teenuseid oma klientidele, kes on soetanud nende teraviljakuivati jälgimissüsteemi nimega Trax ja teine toode on Pinnacle 2020. [7] Pinnacle 2020 on süsteem, mis kujutab endast riistvara koos ekraaniga, mis füüsiliselt ühendatakse Mathews Company teraviljakuivatiga. [8] Kuna süsteem on tootja poolt arendatud spetsiaalselt oma seadmetele, siis on Pinnacle 2020 süsteemi oluline omadus see, et süsteem pakub kogu kontrolli ja ülevaate kuivatist ja selles toimuvatest protsessidest ning kogu olemasoleva reaalajas informatsiooni selle süsteemi kohta.

Trax on veebirakendus, millel on olemas ka mobiilivaade ning annab võimaluse kaugelt jälgida teraviljakuivati andmeid. Rakendus annab võimaluse vaadata järgmisi andmeid:

- Häired
- Temperatuuri ning niiskusandurite informatsioon
- Häired läbi sõnumite või emailide
- Viimase 3 tunni detailsed graafikud kogu teraviljakuivati parameetritega

Süsteemi suurim puudus on see, et seda on võimalik kasutada ainult Mathews Company teraviljakuivatites, mis on toodetud Ameerika Ühendriikides, ning see ei ole ühendatav teiste teraviljakuivati tootjate toodetega. Lisaks sellele pakub ettevõte tavaliselt oma klientidele Trax süsteemi ainult koos Pinnacle 2020 süsteemiga või on kliendil võimalus osta eraldi riistvara Pinnacle 2020 asemel, mille eest tuleb tasuda eraldi tasu ning aastane litsentsimakse Trax süsteemi kasutamiseks.

Autori arvates on Mathews Company poolt pakutav süsteem hea lisa nende terakuivatit omavatele ettevõtetele, sest süsteemil on olemas mugav infograafika ning funktsionaalsus, mis võimaldab mugavalt kuivatusprotsessi muuta.

2.2.3. SCADACcore EnviroLive Grain Drying Monitoring

ScadaScore on ettevõte, mis tegeleb kaugjuhtimis- ning jälgimissüsteemide arendusega erinevates ärivaldkondandes. [9] Eeltoodud süsteemi on võimalik integreerida kõikide sensorite ning teraviljakuivatussüsteemidega, kuna see toetab kõiki populaarsemaid

andmevahetusprotokolle. Lisaks sellele pakub ettevõte oma klientidele täisteenust ja annab võimaluse ka vanemaid teraviljakuivatisüsteeme digitaliseerida. Ettevõte aitab projekti loomisega, riistvara valikul ning ostmisega ja rakenduse seadistamisega kliendi vajaduste järgi.

Süsteemi integreerimiseks olemasoleva teraviljakuivatiga on vaja ainult osta ettevõtte poolt arendatud seade (ViaCell-100), mis on andmevahetuskiht kliendi teraviljakuivati süsteemi ning ScadaScore pilvelahenduse vahel. Süsteem kogub andmeid ning salvestab neid pilveserverites, milles on kõik ajaloolised andmed alati kättesaadavad ning kliendil on võimalus andmeid eksportida enda valitud meetodil ning kujul. Süsteemi oluliseks omaduseks võib pidada erinevaid infograafika võimalusi, süsteem saab ka ajalooliste andmete olemasolul teha graafiku peal trende, mis annavad operaatorile võimaluse ennustada tehtud konfiguratsiooni muudatuste tulemusi koheselt. Nagu kõik eespool analüüsitud süsteemid pakub ka EnviroLive võimalust häireid seadistada ning läbi erinevaid kanaleid informatsiooni edastada.

Autori arvates on kõikidest analüüsitud olemasolevatest lahendustest EnviroLive kõige suuremat funktsionaalsust pakkuv lahendus, mis võimaldab ettevõttel kogu oma tööperioodi kohta andmeid koguda ja analüüsida ning lisaks sellele pakub see operaatoritele kõikvõimalikke andmeid teraviljakuivatusprotsessi jälgimiseks. Süsteemi ainuspuudus on see, et see töötab ainult ühes suunas ehk andmete kogumine ja teraviljakuivati seadistuse muutmine kaugelt ei ole läbi sellise süsteemi võimalik.

2.3. Sisendparameetrite määramine

Antud uurimistöö raames käsitletud probleemi kirjelduse kohaselt on oluline välja tuua, et teraviljakuivatuse jälgimistarkvara peab olema sobiv väiksematele teraviljakuivatitele ning finantsiliselt mõistlik soetamiseks väiksematele ettevõtetele, kes omavad ühte teraviljakuivatit. Lisaks selle peab olema arendatud süsteem, mis ühilduks Siemens Simatic HMI süsteemiga. [10]

Võttes arvesse olemasolevaid ning eelpool toodud lahendusi on autor otsustanud teraviljakuivati jälgimissüsteemi arendada järgmiste põhimõtete alusel.

- Süsteem peab sisaldama minimaalset arvu riistavara, mida on integreerimiseks vaja. Nimetatud asjaolu aitab kulusid kokku hoida.

- Lahendus peab sisaldama võimalikult palju hädavajalikke funktsioone ja minimaalselt lisafunktsioone, mis ei ole antud süsteemis kriitilised. See aitab arendada kuluefektiivsemat lahendust, kuna sisuanalüüsist selgus, et lisafunktsioonide eest tuleb täiendavalt maksta.

- Süsteemi peab olema võimalik minimaalsete muudatustega integreerida erinevate tootjate poolt pakutud terakuivatitesse.

- Süsteem peab omama maksimaalselt lihtsat ja mugavat kasutajaliidest ning informatiivset infograafikat.

- Süsteem peab omama häirete seadistamise ning edastamise funktsionaalsust.

- Süsteem peab salvestama kõiki ajaloolisi andmeid ning kasutajal peab olema ligipääs nimetatud andmetele.

- Süsteem peab olema kasutatav ka väiksemate ekraanidega seadmetes, näiteks nutitelefonis või tahvelarvutis.

Eeltoodud põhimõtete alusel on autori arvamusel võimalik luua süsteem, mis aitab operaatoritel oma tööd efektiivsemalt teha ning ettevõttel tekib võimalus ajalooliste andmete baasil oma terakuivatusprotsessist analüüsi teha.

3. Analüüs

Käesoleva projekti nõuete paika panemisel arvestatakse kahe põhimõttega. Esiteks, lahendus peab töötama Siemens SIMATIC HMI süsteemiga ja võimaldama sellest andmeid koguda. Teiseks, arendusprotsessi skoobis on minimaalne arv kriitilisemaid funktsioone, mis annavad võimalust lõputöö raames arendatud tarkvara reaalmaailmas testida ning analüüsida selle efektiivsust ning -omadusi.

3.1. Siemens Simatic HMI integratsiooni analüüs

Käesoleva töö raames arendatakse projekti lähtuvalt olemasolevast teraviljakuivatisüsteemist, mis on ehitatud kasutades Siemens Simatic HMI PLC. PLC annab võimaluse integreerida erinevate tootjate poolt toodetud andureid. Mainitud PLC valis teraviljakuivati omanik selle kvaliteedi ning funktsionaalsuse tõttu. Nimetatud asjaolu on lõputöö üks olulisematest kitsendustest, mida autor ei saa mõjutada (nt valida teist PLC tehnoloogiat või selle simulatsiooni).

Ülalmainitud süsteem annab võimaluse andmete kogumiseks läbi HTTP serveri, mida saab PLC sees sisse lülitada ning kasutades CGI skripte andmeid anduritest ning süsteemist pärida. [11] Andmete kogumiseks ülalmainitud süsteemist on administraatoril vaja üles laadida HTTP serveri peale HTML lehekülg või mitu lehekülge, kus igal leheküljel on teatud struktuur ja andurite parameetreid kogutakse kasutades anduri identifitseerimistunnust (ID) käivitades spetsiaalne CGI skript.

Kuna lõputöös käsitletud süsteemil on rohkem kui 80 erinevat andurit ning parameetrit, sai tehtud otsus, et luuakse neli erinevat HTML lehekülge iga anduri tüübi jaoks. See on tehtud seoses sellega, et PLC ei oma suures mahus operatiivmälu, üks suur päring võib süsteemi üle koormata ning seejärel läheb PLC taaskäivitusele, mis rikub jooksva kuivatusprogrammi ning peatab teraviljakuivati töö.

HTML lehekülje andmed jagati järgmisteks gruppideks.

- Häired.
- Temperatuuri parameetrid.
- Niiskuse parameetrid.
- Süsteemsed parameetrid.

3.2. Tarkvara kvaliteedimudeli valik

Antud tarkvara eesmärk on pakkuda kasutajatele teatud funktsionaalsust mugaval viisil. Iga tarkvara peab vastama teatud nõuetele, et tagada kliendi rahulolu, ning teatud nõuetele vastamist võib nimetada kvaliteediks. Tarkvara kvaliteedi hindamiseks on loodud mitu erinevat mudelit, millest iga mudel kirjeldab erinevaid kvaliteediomadusi ning nende seoseid. Teatud kvaliteediomadused formuleerivad tarkvara arendusprotsessi ning iseloomustavad erinevaid vajadusi ning piiranguid, mis tähendab, et kvaliteedimudel tuleb valida projekti põhiselt ja selle vajadustest lähtudes. Lõputöö raames analüüsitakse kahte erinevat tarkvara kvaliteedimudelit, FURPS ja McCall, ning valitakse nende seast üks. [12]

3.2.1. FURPS mudel

FURPS mudel sai loodud aastal 1987, mudeli on loonud Robert Grady ning seda on edasi arendanud Rational Software. [12]

FURPS on abreviatuur:

- F (*Functionality*) – Funktsionaalsus. Defineerib funktsionaalsuse, süsteemi võimalused ning turvalisuse.
- U (*Usability*) – Kasutatavus. Kirjeldab lahti inimfaktorid, kasutajaliidese järjepidevuse, kasutajadokumentatsiooni ja koolitusmaterjale.
- R (*Reliability*) – Töökindlus. Defineerib rikete esinemise sagedust ning raskust, süsteemi taastuvust, täpsust ja rikete vahelist aega.

- P (*Performance*) – Jõudlus. Kirjeldab lisaks funktsionaalsetele nõuetele lisatingimusi nagu kiirus, tõhusus, täpsus, taastumisaeg ja ressursside kasutus.

- S (*Supportability*) – Toetatavus. Defineerib süsteemi testitavust, adaptiivsust, ühilduvust ja kasutatavust.

Eeltoodud mudelis kirjeldatakse suuremas osas kliendile orienteeritud infosüsteemi aspekti, st on võimalik väita, et ainuke punkt, mis otseselt ei puuduta kasutajat, on toetatavus (S), kuid see on oluline infosüsteemi aspekt. Autori arvates sobib antud mudel hästi lõputöö raames projekti nõuete määramiseks, kuna on piisavalt kompaktne ning katab peamisi tegureid, mis on MVP raames vajalikud.

3.2.2. McCall-i mudel

Antud mudelis grupeeritakse kvaliteediomadused kolme rühma, millest iga rühm kirjeldab tarkvara omadusi erinevates olukordades või arendusetappides. [12]

- *Product operation* ehk eesti keelde tõlgituna "toote toimimine" defineerib, kuidas tarkvara vastab oma spetsifikatsioonile.

- *Product revision* ehk "toote muutmine" kirjeldab lahti võimalust tulevikus tarkvara muuta kasutajate või kliendi nõuete järgi.

- *Product transition* ehk "toote üleminek" defineerib tarkvara adaptiivsust uude keskkonda.

Kokku 11 tegurit kirjeldavad süsteemi omadusi, millest osa on orienteeritud otseselt kliendiga seotud teguritele ning osa arenduse ja halduse teguritele. Mainitud mudeli eeliseks võib pidada asjaolu, et see kirjeldab lahti rohkem parameetreid kui näiteks FURPS mudel, kuid autori arvates on see antud töönraames liiga mahukas ning sobib paremini suuremate projektide analüüsimiseks.

3.3. Nõuete määramine

Antud projekti nõuete määramiseks sai valitud FURPS mudel, mis tähendab, et kirjeldatakse süsteemi kvaliteediomadusi antud mudeli parameetrite järgi.

3.3.1. Funktsionaalsus

Käesoleva lõputöö raames on oluline arendada süsteem, mis võimaldaks lihtsustada teraviljakuivati operaatorite tööd ning tõsta tööefektiivsust. Nimetatud põhimõttest lähtuvalt määrati süsteemile peamised funktsionaalsed nõuded.

Süsteem peab olema võimeline:

- Koguma andmeid Siemens Simatic HMI PLC-st
- Salvestama andmeid andmebaasi.
- Töötlema saadud andmeid ning koostama nendest ärikriitilist informatsiooni.
- Töötlema häireid ning teavitama kasutajat SMS-i kaudu kindlale numbrile.
- Töötlema süsteemsete parameetrite muudatusi ning salvestama muudatused eraldi.
- Pärima REST API kaudu kasutades JSON vormingut.
- Visualiseerima kasutajaliideses andmeid kuvades operaatorile:
    - Temperatuuri graafikut.
    - Niiskuse graafikut.
    - Tegevuste ajalugu.
    - Häirete ajalugu.

3.3.2. Kasutatavus

Süsteemi kasutatavus on oluline tegur, kuna süsteemi hakkavad kasutama mitte-IT taustaga isikud, kellel on madal arvutioskuste tase.

Kasutatavusega seotud nõuded on:

- Hästi loetavad graafikud.
- Kasutajaliides peab olema adapteeritud tahvelarvutite ekraanisuurusele.

Hetkel autor arvab, et lõputöö raames ei ole mõistlik kirjutada detailset dokumentatsiooni, kuna lõputöö projekti raames arendatakse MVP ning kui klient annab positiivset tagasisidet, siis järgmistel arenduse etappidel on arukas koostada detailsem dokumentatsioon.

3.3.3. Töökindlus

Antud projektis on töökindlus oluline faktor, kuna süsteemist saab oluline osa äriprotsessist ning seetõttu peavad olema täidetud järgmised nõuded.

- Süsteemi vea tõttu ei tohiks olla teraviljakuivati töö häiritud või peatatud.
- Süsteem peab olema sõltumatu PLC-st, mis tähendab seda, et isegi kui kuivatusprotsesses ei tööta, peab süsteemis olema võimalik ajaloolisi andmeid vaadata.
- Ees- või tagarakenduse vea korral, mille tõttu rakendus lülitub välja, peab süsteem automaatselt tegema rakendusele taaskäivituse.
- Süsteem peab olema võimeline töötama 24/7 ilma administraatori abita.

Tähtis on mainida, et antud süsteemi oluline omadus on võimekus andmeid koguda kogu teraviljakuivatamise protsessi jooksul, mis annab ettevõttele ja operaatoritele parema ülevaate ning annab võimaluse hooaja lõpus kogutud andmetele tuginedes teha järeldusi äriprotsesside arendamiseks.

3.3.4. Jõudlus

Tuginedes varasemale kogemusele antud arenduse etapis on autor seisukohal, et palju nõudeid jõudluse kohta ei ole arukas püstitada, kuid seonduvalt süsteemi omadustega võib välja tuua järgmised kvaliteedinõuded.

- Süsteemi minimaalsed riistvara parameetrid on:
    - 8GB RAM.
    - 2 CPU.
    - 256GB SSD.

- API päringute vastuse aeg on alla kahe sekundi.
- Süsteemi saab korraga kasutada viis inimest erinevatest seadmetest.

3.3.5. Toetatavus

Toetatavuse kohta määrati järgmised nõuded.

- Koodibaas peab olema testitud automaattestidega.
- Süsteemi peab olema võimalik testida ilma PLC-ta.
- Süsteemi liides peab olema ühildatav teiste PLC mudelitega.
- Süsteem peab töötama Linux ja Windows operatsioonisüsteemidega.

3.4. Arendustehnoloogia määramine

3.4.1. Tagarakenduse tehnoloogia määramine

Tagarakenduse arendamiseks on oluline valida programmeerimiskeel ning vajadusel raamistik, mis võimaldaksid täita järgmised ülesanded.

- HTTP päringute tegemine PLC-sse.
- Tagastatud HTML töötlemine.
- Andmete andmebaasi salvestamine.
- Andmete edastamine läbi JSON API kaudu.

Ülaltoodud ülesannete täitmiseks on autoril vaja otsustada, mis tehnoloogiaga saab kiiresti ning mugavalt HTTP serveri tööle panna ning selle API defineerida. Lisaks sellele peab olema võrdlemisi lihtne viis andmebaasiga integreerimiseks, andmete pärimiseks ning salvestamiseks.

Autori arvamuse kohaselt võib antud kontekstis vaadelda järgimisi tehnoloogilisi võimalusi ning hinnata autori oskustaset nendega töötamisel. [13] - [18]

Tehnoloogia	Õppimiskeerukus	Kogemus
Java + Spring Boot	Madal	Suur
NodeJs	Keskmine	Väike
C#	Kõrge	Väike
Kotlin + Ktor	Madal	Keskmine

Tabel 1. Tehnoloogiate võrdlus.

Autor otsustas valida tagarakenduse arendamiseks Java programmeerimiskeele ning raamistikuks Spring Boot, kuna autoril on nende tehnoloogiatega kõige rohkem kogemust ning seetõttu on võimalik optimeerida arendusprotsessile kuluvat aega.

3.4.2. Andmebaasi valik

Andmebaasiks on autor valinud PostgreSQL, kuna see on autorile tuttav tehnoloogia, mis on tänapäeva turul kõrgelt hinnatud ning lisaks sellele omab PostgreSQL vabavara litsentsi, head dokumentatsiooni ning piisavalt palju informatsiooni on võimalik pärida avalikest allikatest. [19] [20] PostgreSQL on lihtne integreerida Java ja Spring Boot-ga ning selline kombinatsioon on autori kogemuse järgi ennast tõestanud suuremahulistes IT süsteemides.

Plaanis on jagada kaheks etapiks andmete pärimine ja töötlemine. Nimelt esimesel etapil andmeid salvestatakse nn toorkujul ja siis töödeldatakse, et moodustada nendest ärikriitilist infot. Seega on võimalik kasutada kahte erinevat andmebaasi tehnoloogiat, kus toorandmed salvestatakse NoSQL andmebaasi ning töödeldud andmed hoitakse PostgreSQL-s, kuid antud töö raames selline lähenemine on liiga mahukas ja seetõttu piirdutakse ühe andmebaasi tehnoloogiaga. [21]

3.4.3. Eesrakenduse tehnoloogia valik

Praegusel turul on kasutusel suuremal määral kolm erinevat eesrakenduse tehnoloogiat, milleks on React, Angular ning Vue. [22]

Vue on hea ja kompaktne raamistik eesrakenduste arendamiseks, kuid sellega tööd alustades kulub suur osa ajast projekti seadistamisele. Raamistikul on teistega võrreldes

oluliselt vähem valmislahendusi, millega kiiresti mugavat kasutajaliidest valmis saada. [23]

Angular on suurem raamistik kui Vue ja React ning Angular põhineb MVC mudelil, seda raamistikku toetab ja arendab Google, ning see on üks raamistiku miinustest, kuna React ja Vue on vabavaralised ja nende toetus ning edasiarendus käib kiiremini. Lisaks eelnevale lahendatakse tavaliselt viimaste probleemid võrdlemisi kiiremini. [24]

Autor on antud töö raames valinud eesrakenduse tehnoloogiaks React raamistiku, kuna sellega saab kiiresti minimaalse funktsionaalsusega projekti valmis kirjutada ning Reactil on olemas mitu erinevat valmis disainiga teeki, mida saab kasutusele võtta. [25]

3.4.4. Arenduskeskkonna valik

Kuna antud lahendus peab töötama erinevate operatsioonisüsteemide peal, siis sai tehtud otsus projekti arendada kasutades Docker tehnoloogiat. Docker annab võimaluse projektide osadest ehk rakendustest teha pildid ehk Docker Image, mida saab lihtsasti virtualiseerida. Docker Image tihti nimetatakse ka Snapshotid-eks, pilt on üldises mõttes fail, mille sees on rakenduse kood, teegid ja teised vajalikud elemendid rakenduse tööle panemiseks. [26] [27]

Kuna tööle on vaja panna mitu rakendust korraga, siis on mugavam kasutada *docker-compose* tehnoloogiat [28]. Antud süsteem annab võimaluse lihtsamal viisil kirjeldada failina süsteemi tööle panemise malli ning annab võimaluse rakendused ühendada läbi virtuaalse privaatse võrgu. Autor arendab projekti MacOS operatsiooni süsteemil ning kasutades IntelliJIdea Ultimate Edition, millega saab mugavalt kirjutada Java ja React projekte [29] [30].

4. Prototüübi arhitektuur

Prototüüpi planeeritakse ühte konkreetsesse teraviljakuivatisse, mis asub maal ning kus interneti kvaliteet on nõrk, mistõttu sai tehtud otsus arendada prototüüp sellisena, et see töötaks sisevõrgus. Sisevõrguks peetakse antud kontekstis võrku, mis asub teraviljakuivati asukohal oleva ruuteri taga. Seoses sellega autentimine ning muud sellega seotud süsteemi aspektid on lõputöö raames arendatava projekti skoobist väljas. Niimoodi saab tagada parema prototüübi töökindluse ning kvaliteedi, kuna ei pea arvestama riske, mis kaasnevad avaliku välisvõrgu ühendusega.

4.1. Süsteemi arhitektuur

Antud süsteemi arhitektuuri võib kirjeldada kasutades kihilist arhitektuuri ehk *layered architecture*, kuid siin on tähtis mainida, et sellist tüüpi arhitektuuri võib kasutada nii terve infosüsteemi kirjeldamiseks kui ka iga rakenduse komponendi jaoks eraldi. Käesolevas lõputöös kirjeldatakse kogu süsteemi komponente ja nende seoseid. [31]

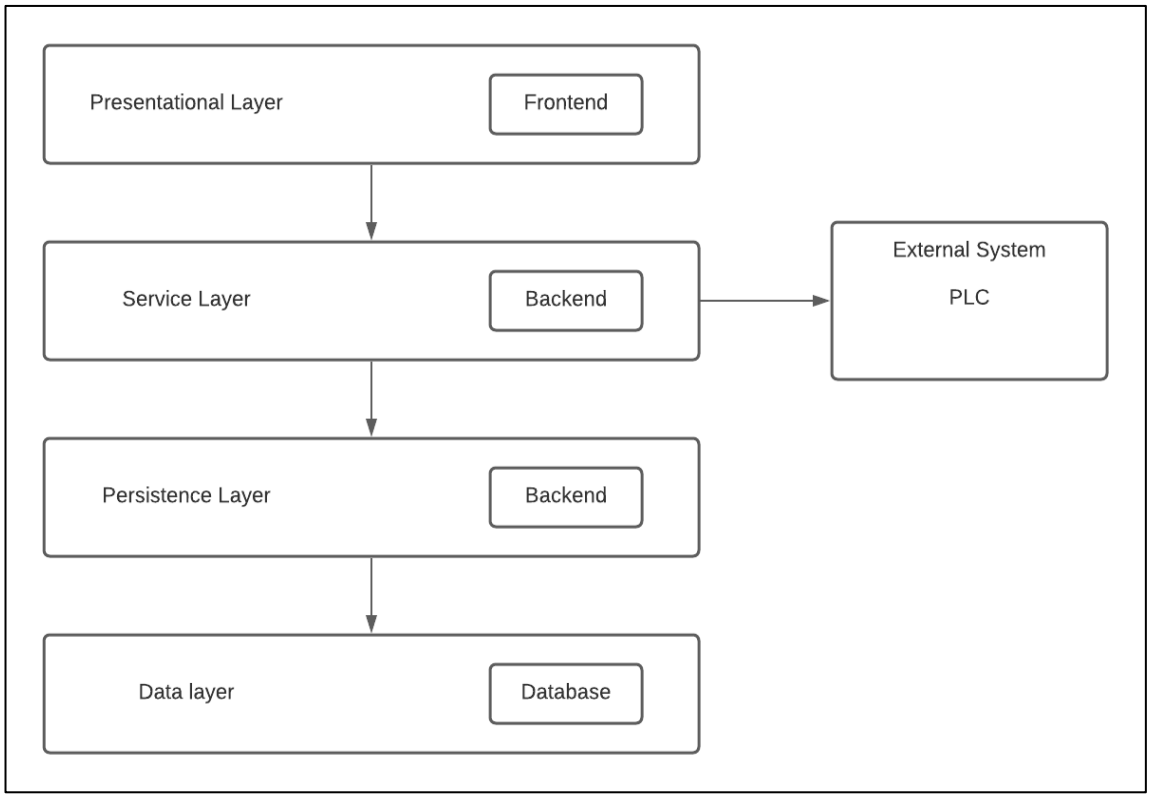
Kihiline arhitektuur on üks lihtsamatest arhitektuuri kirjeldamisviisidest, mis on laialt kasutatud väiksemate süsteemide ehitamiseks. Mainitud arhitektuuri mall ei defineeri, mitu kihti peab süsteemil olema, kuid arendaja ise teeb seda. Kihilise arhitektuuri malli prioriteediks on isolatsioon, mis aitab kapseldada iga kihi loogikat. Sellest mallist on hiljem lihtne üle minna teistele, näiteks mikroteenuste arhitektuuri peale, kui teha seda õigeaegselt. [32] Iga kiht pakub teistele kihtidele andmevahetuseks rakendusliidest API. Näiteks presentatsiooni kiht kasutab ärikihi JSON API-d, et sellega süsteemis andmeid vahetada.

Antud töö raames arendatavas süsteemis on neli peamist komponenti:

- Eesrakendus,
- Tagarakendus,
- Andmebaas,
- PLC.

Eesrakendusel peab olema võimalus teha päringuid tagarakendusse, kuid eesrakendusel on piiratud ligipääs PLC või andmebaasi komponentidele. Sellega seoses on võimalik väita, et süsteemil on olemas presentatsiooni kiht ehk *presentation layer*. Selle kihi ainuke funktsioon on konkreetse töö andmete visualiseerimine kasutajale.

Tagarakendus on süsteemi tuum, sellel peab olema ligipääs PLC-sse andmete pärimiseks ning andmebaasi andmete salvestamiseks ja pärimiseks. Lisaks sellele, tagarakenduse sees on ka kõikvõimalik äriloogika, mida näiteks kasutatakse andmete töötlemiseks. Sellest tulenevalt võib öelda, et tagarakenduses on kaks kihti - ärikiht ja püsivuskiht. Seda rakenduse arhitektuuri kirjeldatakse hiljem eraldi peatükis. Andmebaas on eraldi kiht, mille nimeks on andmekiht ehk *Data Layer*.



Joonis 2. Kihiline arhitektuur.

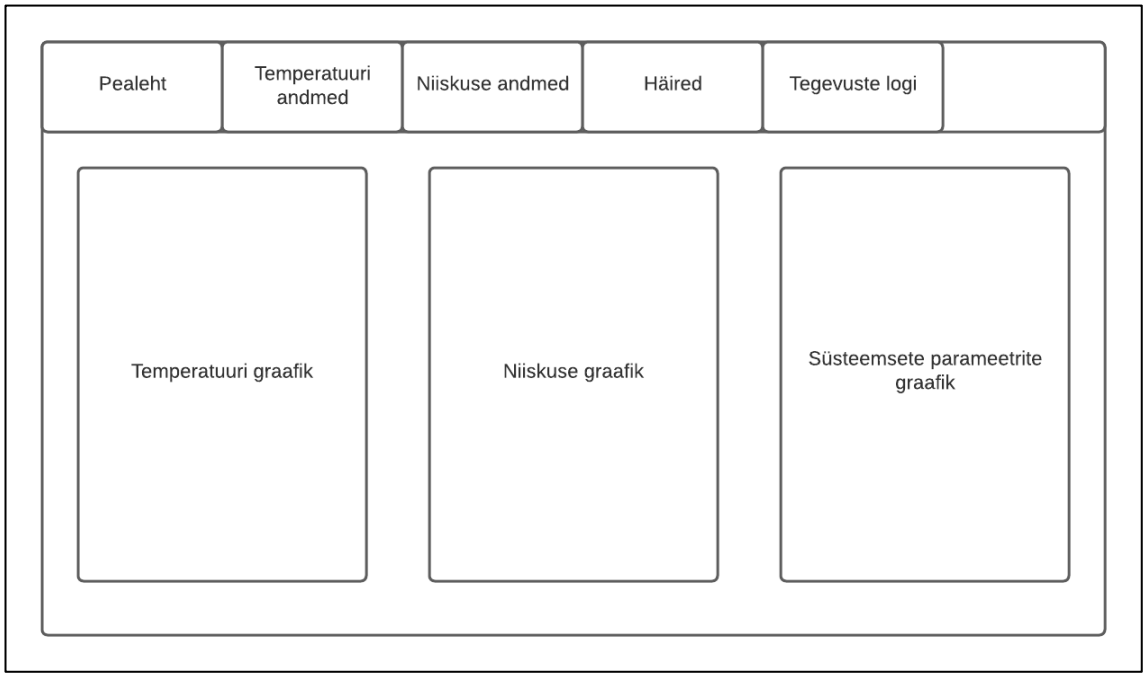
Joonisel number 2 on näidatud nooltega komponentide sõltuvuste suund. Lisaks sellele võib antud joonisel näha, et PLC on antud süsteemi raames välikomponent ehk *External System*, mis on eraldiseisev süsteem, millega süsteemi ärikiht (*Service Layer*) võib andmeid vahetada.

Antud mall annab võimaluse iga kihi kapseldatud testimiseks ning lisaks sellele ei kaasne äriloogika kihi muutumisega muudatused teistes kihtides (vähemalt olukorras, kus muudatus ei ole suuremahuline).

4.2. Eesrakenduse disain

Eesrakenduses kasutatakse arhitektuurset malli, millele tuginedes oli valitud raamistik ehk React ehitatud. React-i põhimõte on arendada süsteem kasutades taaskasutatavaid komponente. Sellega seoses tuleb antud rakenduse arenduses meeles pidada, et võimalusel tuleb komponente taaskasutada selleks, et minimeerida koodi duplikatsiooni. Viide eesrakenduse lähtekoodile asub lisas 2.

Joonisel number 3 on toodud plaaneritava kliendirakenduse esilehekülje disain. Jooniselt on võimalik näha, et süsteemis plaanitakse algselt kuvada viis ekraani ning joonisel on näidatud peaekraani ehk pealehe disain.



Joonis 3. Esirakenduse pealehekülje disain.

Pealehekülg on operaatori peamine töörist, kust ta näeb oma peamiseks tööks vajalikku informatsiooni - temperatuuri, niiskuse ning süsteemsete parameetrite graafikud. kirjeldatud komponentide paigaldamisviisi abil saab operaator korraga vaadata ja võrrelda kolme erinevat, kuid seotud graafikut.

Temperatuuri andmete leheküljelt võib operaator juba tabeli kujul temperatuuriandurite andmeid vajadusel vaadata. Niiskuse andmete leheküljelt saab operaator vajadusel samamoodi niiskuse andurite andmeid tabeli kujul vaadata. Häirete leheküljelt näeb operaator aktiivsete ning ajalooliste häirete logi: millal süsteem sai häirest teada ning millal süsteemi andmetel probleem sai lahendatud.

Tegevuste logi on oluline ekraan, mis annab operaatorile võimaluse vaadata, millal ja mis teraviljakuivati parameetrid olid muudetud. See on tihti oluline kui uus töötaja alustab oma vahetust ning vaatab järgi, mida eelmine operaator on vahetuse jooksul teinud ning lisaks sellele näeb, millised parameetrid on praegu seadistatud.

4.3. Tagarakenduse disain

Tagarakenduse kaks peamist funktsiooni on andmete pärimine PLC-st ning andmete edastamine esirakendusele läbi JSON API. Andmete pärimine PLC-st on eraldiseisev tagarakenduse loogika osa ning on vajalik seda kapseldada niimoodi, et kui hiljem otsustatakse üle minna mikroteenuste arhitektuuri peale, siis oleks võimalik seda võimalikult kiiresti ning mugavalt kliendi rakendusest (antud kontekstis tagarakenduse osa, mis pakub esirakendusele API teenust) eraldada ja teha sellest eraldi rakendus. Sellega seoses on otsustatud arendada andmete pärimise loogikat eraldi kapseldatud komponendina tagarakenduse sees (viite tagarakenduse lähtekoodile on võimalik leida lisas 2).

PLC-st andmete pärimise protsess on otsustatud teha mitmeetapiliseks, kus esimeses etapis kogutakse andmeid PLC-st ning salvestatakse neid andmebaasi PLC-st saadud kujul. Välja toodud lähenemine probleemile annab võimaluse andmeid salvestada sellisel kujul, kuidas PLC on neid edastanud. Teises etapis saab protsessi taas käivitada ja juba lahendatud probleemiga andmeid õigesti töödelda. Kuna iga etapp on eraldiseisev, siis on hiljem võimalik võrdlemisi lihtsalt ja mugavalt etapid ümber teha eraldiseisvateks mikroteenusteks.

Rakendus on otsustatud teha DDD (*Domain Driven Design*) malli järgi. See mall on ülesehitatud ideel, et tarkvara peab olema ehitatud äridomeeni baasil, mis on juba kujundatud mitmeaastase domeenieksperte kogemusega. Selle malli kasutusele võtmisel disainivad arendajad unikaalse ja domeeni-spetsiifilise DSL-i ehk *Domain*

Specific Language, mis annab võimaluse koodi kirjutamise protsessis kasutada äripõhilisi sõnu. Nimetatud lähenemine annab võimaluse arendajatele ja äriinimestele üksteisest paremini aru saada. [33] [34]

DDD malli järgi on võimalik domeenid rakenduses defineerida järgnevalt.

- Häired (*Alarms*).
- Süsteemsete parameetrite muutmise või tegevuste logi (*Activity Log*).
- Temperatuuriandurite andmed (*Temperature*).
- Niiskuseandurite andmed (*Moisture*).
- Teraviljakuivati programmi parameetrid (*Drying Parameters*).

Kliendirakenduse päringute töötlemiseks võetakse kasutusele JSON API, mida arendades on oluline kinni pidada standardist. Käesoleva töö raames võtab autor kasutusele REST standardi. Näiteks on toodud häirete ning tegevuste logi *endpoint*-d, kus esimene tagastab kogu nimekirja ning teine ühe teatud kirje. [35]

- /api/alarms
    - /api/alarms/{id}
- /api/activity-logs
    - /api/activity-logs/{id}

4.4. PLC HTML leheküljed

PLC integratsiooni jaoks tuleb arendada HTML leheküljed, kus on tehtud viited kasutades elemendi ID *tag*-i teatud anduri ID peale, ning kust saab välja kutsuda CGI skripti, mis teostaks PLC-s käsu ja mis omakorda tagastaks andurist andmeid ning paneks need HTMLi sisse. Joonisel 4 on kujutatud HTML malli, kus iga elemendi sees on märgitud anduri ID süsteemis ning kommentaarina on lisatud CGI käsk, mida PLC teostab, kui antud lehekülge päritakse.

```
<HTML>
	<HEAD>
		<title>MOISTURE_SAMPLE</title>
	</HEAD>
	<BODY>
		<div id=’D317’><!-- #exec cgi=”/get_device.fn D317” --></div>
		<div id=’D351’><!-- #exec cgi=”/get_device.fn D351” --></div>
		<div id=’D356’><!-- #exec cgi=”/get_device.fn D356” --></div>
	</BODY>
</HTML>
```

Joonis 4. PLC HTML koodinäide.

Vastuseks saame HTML lehekülje joonisel 5 toodud formaadis, kus iga elemendi sees on kirjutatud andurist saadud andmed ning ka on jäänud anduri ID. Antud näites tagastavad andurid andmeid sellises formaadis, et nendest protsentide saamiseks on vaja saadud tulemusi sajaga korrutada.

```
<HTML>
	<HEAD>
		<title>MOISTURE_SAMPLE</title>
	</HEAD>
	<BODY>
		<div id=’D317’>0.4534568<div>
		<div id=’D351’>0.3214567<div>
		<div id=’D356’>0.1034646<div>
	</BODY>
</HTML>
```

Joonis 5. Päringu tulemuseks saadud HTML-i näide.

4.5. Programmikoodi kirjutamise printsiibid

Rakenduste arendamisel on oluline kinni pidada teatud praktikatest, et koodi kvaliteet oleks kõrge ning vajadusel oleks uuel arendajal olemas võimalus tehtud lahendusest aru saada ning seda edasi arendada. Autori arvates on olulisteks printsiipideks *Clean Code* ehk puhta koodi printsiibid, mida Robert C. Martin oma raamatus on kirjeldanud [36]. Need on üle maailma austatud printsiibid, mida peetakse heaks tavaks koodi kirjutamisel. Lisaks sellele on olulised SOLID printsiibid. [37]

SOLID on abreviatuur, mida saab lahti kirjutada järgmiselt.

- *Single Responsibility Principle* – igal komponendil peab olema üks teatud vastutus ja ainult üks põhjus selle muutmiseks. Seda saab kasutada meetodite, klasside ning rakenduste põhiselt.

- *Open Closed Principle* – arendajal peab olema võimalus klassi uue funktsionaalsusega täiendada klassi sisu modifitseerimata.

- *Liskov Substitution Principle* - alamklassid tuleb arendada niimoodi, et objekti kasutaja jaoks ei oleks vahet, kui talle antakse alamklassi tüüpi objekt. Ehk alamklass ei pea tegema midagi, mida ülemklassist ei eeldaks.

- *Interface Segregation Principle* - liidese kasutaja ei tohiks sõltuda funktsionaalsusest, mida tal otseselt vaja ei lähe, sellisel juhul jagatakse liides väiksemateks liidesteks.

- *Dependency Inversion Principle* – tarkvarakomponendid peavad sõltuma abstraktsioonidest, mitte konkreetsetest realisatsioonidest.

5. Prototüübi testimine

Tarkvarakvaliteedi ning nõuetele vastavuse väljaselgitamiseks on oluliseks osaks arendatud prototüübi testimine. Tarkvara testimine annab võimaluse ettevõttele kaugemas perspektiivis kulusid kokku hoida ning minimeerida võimalikke probleemide tekitamist [38].

5.1. Automaattestimine

Käesolevas lõputöös arendatud prototüüpi testitakse kahte tüüpi automaattestidega: ühik- ning integratsiooni-testidega. Mainitud otsus sai tehtud testimispüramiidi põhjal, mis ütleb, et tarkvaras peab kõige rohkem olema ühikteste, vähem integratsiooniteste ning kõige vähem E2E (*End to End*) teste. [39] Kuna antud prototüübi arendamises oli oluline võimalikult efektiivselt kasutada arendusprotsessiks kulunud aega, otsustati, et E2E teste tehakse käsitsi ehk manuaalselt, kuna automaat E2E testide arendus on liiga ajamahukas ja ei mahu lõputöö mahupiirangutesse.

Ühiktestidega testitakse väiksemaid süsteemi elemente ehk klasse ning funktsioone, reeglina kui antud funktsioon või klass vajab mõnes teisest süsteemiosast vastust, siis seda vastust emuleeritakse kasutades *mock*-e. Java keeles kasutatakse selleks projektide arendamises kõige rohkem *Mockito* raamistikku. Ühiktestid annavad võimaluse testida nii, et isoleeritud koodiosa töötab korrektselt ning kuna ühiktestide kirjutamine on suhteliselt kiire ja võrdlemisi lihtne, on võimalik väiksema ajakuluga katta rohkem loogikat väiksemate testidega. See omakorda annab võimaluse integratsioonitestide mahu vähendamiseks, kuna nende kirjutamine on ajamahukas protsess. [40] [41]

Integratsioonitestidega kontrollitakse seda, et eraldi arendatud süsteemiühikud töötavad korrektselt kui need on omavahel ühendatud [42]. Antud töö raames on mugav integratsioonitestidega katta suuremad osad funktsionaalsusest, heaks näiteks võib tuua andmete pärimine REST lõpp-punkti kaudu, kus testitakse, et päringuga tagastatakse andmebaasist õiged andmed. Selliste testide kirjutamine võtab rohkem aega, kuna on vaja andmebaasi sisestada testi jaoks vajalikud andmed ning päringu test-objektid on

suuremad kui näiteks ühiktestide kirjutamisel. Lisaks sellele, tulemusena saadud andmete hulk on suurem.

5.2. E2E testid

Käesoleva lõputöö raames arendatud prototüübi E2E testid otsustati teha manuaalselt ehk teste kirjeldatakse dokumentidena *Confluence* keskkonnas, lähtudes oodatavatest lõpptulemustest ning süsteemi iseloomust ja neid teostatakse käsitsi, kasutades kasutajaliidest. Teste teostatakse vajadusel kooskõlas kliendiga, reeglina seda tehakse iga kahe nädala pärast. [43]

E2E testide jaoks arendati eraldi versioon tagarakenduses olevast HTTP kliendist, mis emuleerib PLC-d. HTTP päringuid ei saadeta PLC-sse, vaid tagastatakse teatud numbrivahemikus juhuslikke arve. Selline lahendus sai tehtud, kuna süsteemi ei ole koguaeg võimalik testida PLC-ga.

5.3. Testimisjärgsete puuduste parandamine

Tarkvaraarenduse protsessi jooksul olid iga teatud funktsionaalsuse kohta samal ajal selle realiseerimisega kirjutatud ka automaattestid, mis on aidanud autoril koheselt leida ning parandada süsteemi puudusi. Arendatud süsteemi üheks eeliseks on automaattestide olemasolu, millega kaeti suurem osa süsteemist. See annab võimaluse minimeerida regressiooni iteratiivses tarkvaraarenduse protsessis. [44]

E2E testide kirjutamise ning käivitamise protsessis leiti järgmised suuremad süsteemi puudused.

- Tagarakendusest tagastatavad temperatuurinäitude andmed ei kasutanud ajalist limiiti, mis oli edastatud esirakendusest. Seetõttu laaditi andmebaasist korraga mällu suur hulk andmeid ning rakendus ei saanud neid töödelda.

- Tagarakendusest tagastatavad niiskuse andmed olid vales formaadis. Seetõttu oli nendest ehitatud graafik puudulik, andmete ümardamisel kasutatud loogika ümardas komakohaga andmeid enne, kui nendest olid protsentide kujul andmed tehtud.

- PLC-st süsteemsete parameetrite andmete töötlemisel tekitas süsteem uue tegevuste logi kirje iga kord kui andmeid PLC-st päriti, seetõttu olid tagastatavad andmed kaheksa komakohaga arvud, kuid andmebaasi salvestati need kümme komakohaga. Sellest tulenevalt lisati lõppu nullid, mistõttu võrdlemine ei töötanud õigesti, kuna andmed on tekstiformaadis.

- Aktiivsete häirete tabelis olid koos aktiivsete häiretega toodud ka ajaloolised ehk juba lõppenud häired. Kirjeldatud süsteemi käitumine oli põhjustatud sellega, et andmebaasi päring oli valesti kirjutatud.

- PLC-st saadud andmete töötlemise funktsioon töötles andmeid uuesti süsteemi taaskäivitamisel. Probleem oli seotud sellega, et süsteemil puudus võimalus aru saada, millised andmed on juba töödeldud. Selle probleemi lahendamiseks sai arendatud eraldi loogika, mis lisas saadud andmetele järjekorranumbrid, mille järgi andmeid töödeldakse.

Lisaks ülaltoodud suurematele süsteemi puudustele parandati testide jooksul leitud ka väiksemad puudused ning osa süsteemi vigadest sai parandatud enne kui need olid tänu ühik-ja integratsioontestidele kliendile testimiseks antud.

6. Hinnang

Süsteemile hinnangu andmisel tugineb autor peatükis 3.3 määratud nõuetele, testide tulemustele ning kliendi poolt saadud tagasisidele.

6.1. Vastavus esitatud nõuetele

Peatükis 3.3.1 toodud nõutud funktsionaalsus sai antud uurimistöö raames arendatud ning testitud. Tulemuseks saadud funktsionaalsus vastab kliendi ootustele ning täidab nõutud funktsioone, kuid arenduse protsessis olid välja selgitatud mitmed parandused alguses määratud funktsionaalsusele. Eeltoodud väide viitab faktile, et arendatud prototüüp on aidanud ärile paremini oma ideedest aru saada ning tekitanud täiendava ärilise vaate, millega saab toodet edasi arendada ning seejärel toodet turul pakkuda.

Peatükis 3.3.2 kirjeldatud kasutatavuse nõuded said täidetud kliendi ning testijate arvamusele põhinedes, antud kontekstis on testijaks autor. Veebirakendust on võimalik mugavalt kasutada nii arvutiekraanil kui ka väiksematel tahvelarvutite ekraanidel, kuid tahvelarvutite ekraanidel on mugav rakendust kasutada ainult horisontaalses (*landscape*) vaates. Testijate poolt saadud tagasisides on mainitud, et vertikaalvaates graafikute peal olevad numbrid jäävad nähtamatuks.

Peatükis 3.3.3 kirjeldatud töökindluse nõuded said osaliselt täidetud, süsteem töötab autonoomselt kuni tekib teadmata olukord. Näiteks PLC-st saadud andmete formaat muutub, kuna tehnik on vahetanud ühe anduri teise tootja anduri vastu. Sellisel juhul peab arendaja tarkvaras muudatusi tegema, et süsteem jätkaks oma tööd õigesti. Heaks tulemuseks oleks võimalik lugeda olukorda, kui süsteem oleks teavitanud kasutajaid, et antud andurist ei ole enam võimalik andmeid töödelda.

Peatükis 3.3.4 kirjeldatud jõudluse nõuded said täidetud. Süsteem kasutab riistvaralisi ressursse efektiivselt ning ei nõua võimsamat riistvara.

Peatükis 3.3.5 kirjeldatud toetatavuse nõuded said täidetud, süsteemil on suurem osa funktsionaalsusest kaetud automaattestidega ning tähtsamaid kasutajalugusid võib testida käsitsi E2E testidega. Süsteemi on võimalik testida ilma PLC-ta. Lahendus töötab nii MacOS kui Windows süsteemidel, kuid serveritele peab olema installeeritud Docker ning

docker-compose. Süsteem on antud lõputöö kirjutamise hetkel ühilduv ainult Siemens Simatic HMI PLC-ga, kuid arendatud tarkvara annab võimaluse teha uut integratsiooni kogu süsteemi muutmata.

6.2. Efektiivsus

Antud lõputöö raames arendatud tarkvara efektiivsuse hindamiseks kasutab autor erakogus olevaid andmeid projekti erinevate osade maksumuse ning saadud finantstulemuste kohta.

Antud lahenduse integreerimiseks hetkel olevasse süsteemi on kliendil vaja soetada riistvara, mis võimaldaks antud tarkvara tööle panemist ning lisaks sellele peab klient kooskõlas arendajatega PLC-le integratsiooni arendama. Projekti kogukulu kontseptsiooni kontekstis on püsikuludeks igakuine arvuti rent ja haldus ning soetusmaksumusse kuulub ühekordne integratsiooni arendus. Nimetatud kulustruktuur on kirjeldatud tabelis 2.

Teenus	**Hind**
Arvuti rent ja haldus	100 EUR/kuu
Integratsiooni arendus	2500 EUR

Tabel 2. Süsteemi integreerimise tasu.

Lõputöö raames arendatud süsteem oli kasutusele võetud ühel teraviljakuivatil kaheks kuuks testimise eesmärgil. Nimetatud perioodi tulemusi võrreldi eelneva 2020. aasta tulemustega (vt tabel 3).

Parameeter	**2020. aasta näitaja**	**2021. aasta näitaja**
Rikutud vilja maht tonnides	9	7
Teraviljakuivati teostatud töö tundides	1205	1298

Tabel 3. Süsteemi rahaline efektiivsus.

Tabelis 3 olevad tulemused näitavad, et rikutud vilja maht on kahe tonni võrra vähenenud ning teraviljakuivati töötas 93 tundi rohkem. Viimane on osaliselt seotud sellega, et operaatoritel oli võimalik probleeme ennustada süsteemi abil ning kriitilisi vigu oli vähem, mille tõttu peatatakse teraviljakuivati töö. Eeltoodud andmeanalüüs näitab, et

süsteem on ettevõttele kasu toonud nii rikutud vilja koguse vähendamise kui ka tõstetud tööproduktiivsuse kaudu. Täpsemad projekti tasuvuse määrad loetakse ettevõtte ärisaladuseks, seega antud lõputöö raames toob autor ärilise efekti ja protsessi optimeerimise tabelis 3 esitatud andmete kaudu. Siinkohal on oluline mainida, et analüüsitud periood on kaks kuud ning täpsemad statistilised tulemused saadetakse järgmisel kuivatusperioodil.

6.3. Edasiarendus

Antud lõputöös käsitletud projekti edasiarendust võib teostada erinevates suundades ja vastavalt sellele, mis selgub kasutajatele kõige vajalikumaks pärast süsteemi pikemat kasutamist. Edasiarenduse vajadus võib tulla ka peale süsteemi integreerimist ja kasutamist ettevõtte partneritel, kes on näidanud huvi antud süsteemi vastu.

Üks võimalikest edasiarendustest on uute PLC integratsioonide arendus, milleks on vajalik välja selgitada, millised on sihtturul kõige levinumad PLC-d ning millised on võimalused nendega integreerimiseks.

Teine võimalik edasiarenduse suund on operaatorite töö korrigeerimine tehisintellekti abil, kuid selleks on vaja teha analüüs, mis hõlmab endas mitme aasta jooksul kogutud andmete töötlemist süsteemi töö kohta. Põhjalik testimine antud valdkonna spetsialistidega on samuti kriitilise tähtsusega. Põhiideeks on ajalooliste ning jooksvate andmete põhjal ennustuste tegemine ning ennustuste baasil operaatorile soovituste andmine vilja töötlemise protsesside optimeerimiseks.

Autori arvates on ka mitu võimalust olemasoleva operaatorisüsteemi edasiarendamiseks. Rakendusse võib lisada autentimise, mis võimaldaks iga operaatori andmeid tema töövahetusega siduda ning hiljem anda talle tagasisidet teostatud töö kohta.

Pilveteenuse arendus võib osutada ka perspektiivseks suunaks. Sellise süsteemi abil on võimalik suurematel ettevõtetel kõik oma teraviljakuivatitest saadud andmed ühendada ühte platvormi ning jälgida, kuidas iga teraviljakuivati oma tööd teostab. Lisaks eelnevale võib pilvesüsteem osutuda heaks analüüsi tööriistaks nii klientidele (teraviljakuivati omanikele) kui ka teistele ettevõtetele, kuna see annab ligipääsu suuremale andmehulgale, mida klientidega kokkuleppel võib kasutada tehisintellekti süsteemi arendamiseks.

7. Kokkuvõte

Käesoleva lõputöö eesmärk oli arendada ja testida teraviljakuivati jälgimissüsteemi, mida on võimalik Siemens SIMATIC HMI PLC-ga ühendada. Eesmärgi saavutamisel anti parem ülevaade kuivatamise protsessist ning arendati tööriist, mille abil operaatorid saavad oma tööd efektiivsemalt teha.

Lahendust arendati kasutades kihilist arhitektuuri. Rakenduse disaini kuuluvad ees- ja tagarakendus. Eesrakendust programmeeriti kasutades React raamistikku ja selle ülesandeks on visualiseerida kuivati kasutajale kuivati parameetreid erinevate graafikute kujul. Tagarakendust realiseeriti Java programmeerimiskeele abil ja selle peamine ülesanne on JSON API rakendusliidese kaudu teraviljakuivati andmete pärimine andmebaasist ja esitamine kasutajaliideses.

Rakendust arendati kasutades TDD lähenemist. Süsteemile on kirjutatud ühiktestid ning rakendust on testitud E2E ja integratsioonitestidega. Testimisprotsessi kaasati klient tagades seekaudu testitavate stsenaariumite asjakohasus.

Projekti investeeringu maksumuseks on 2600 EUR. Kulustruktuuri soetusmaksumuseks on integratsioonitasu väärtuses 2500 EUR ja püsikuludeks 100 EUR kuus arvuti rendi ning halduse eest. Pärast kuivati testimist ja lahenduse integreerimist parandati äritulemust kahes komponendis. Esiteks, rikutud vilja mahtu vähendati rohkem kui 20%. Teiseks, teraviljakuivati töötas peaaegu 10% kauem tänu lahenduse poolt genereeritud eelhoiatustele, mille tagajärjel muutis kasutaja kuivati seadeid ja vältis vigu.

Projekti võib lugeda õnnestunuks, kuna loodud rakendusega on võimalik teraviljakuivatusprotsessi jälgida ning operaatoritel on võimalus kasutada tööriista, mis annab võimaluse mugavalt jälgida protsessiga seotud andmeid ning vajadusel protsessi korrigeerida.

Kasutatud allikad

[1] A. Inginen, *AS Tartu Agro Teraviljakeskuse tehnoloogia rekonstruktsioon,* Tartu: Eesti Maaülikool, Tehnikainsituut, 2014.

[2] A. Ylä-Soini, *Automatization of a grain dryer,* SeAMK School of Technology, Automation Technology, 2015.

[3] C. Roberts, „A conceptual framework for quantative text analysis," *Kluwer Academic Publishers,* 2000.

[4] J. W. J. A. William Jr, „Comparing structured and unstructured methodologies in firmware development," *Hewlett-Packard Journal,* 1989.

[5] Dryer Master Inc., „Dryer Master," Dryer Master Inc., 2021. [Võrgumaterjal]. Available: www.dryermaster.com. [Kasutatud 2 December 2021].

[6] Dryer Master Inc., „Dryer Master," 31 July 2020. [Võrgumaterjal]. Available: https://www.dryermaster.com/userContent/documents/DM100%20Manual%20-%202020.pdf. [Kasutatud 2 December 2021].

[7] Mathews Company, „Learn About Us," [Võrgumaterjal]. Available: https://www.mathewscompany.com/learn-about-us.html. [Kasutatud 2 December 2021].

[8] Mathews Company, 12 September 2016. [Võrgumaterjal]. Available: https://www.mathewscompany.com/uploads/1/1/5/5/115558437/p2020_broch_12-9-16_web.pdf. [Kasutatud 2 December 2021].

[9] EnviroLive, by SCADACore, „Home," 2021. [Võrgumaterjal]. Available: https://envirolive.scadacore.com/. [Kasutatud 2 December 2021].

[10] Siemens, „SIMATIC HMI Basic Panels," [Võrgumaterjal]. Available: https://new.siemens.com/global/en/products/automation/simatic-hmi/panels/basic-panels.html. [Kasutatud 2 December 2021].

[11] R. E. M. Nancy J. Yeager, Web Server Technology, Burlington, Massachusetts: Morgan Kaufmann, 1996.

[12] M. H. S. M. A. J. A. B. Al-Badareen, „Software Quality Models: A Comparative Study," %1 *Software Engineering and Computer Systems: Second International Conference*, Kuantan, 2011.

[13] Oracle, „Java,“ Oracle, 2021. [Võrgumaterjal]. Available: www.oracle.com/java/. [Kasutatud 2 December 2021].

[14] JetBrains s.r.o., „Ktor Documentation,“ JetBrains s.r.o., 2021. [Võrgumaterjal]. Available: ktor.io/docs/welcome.html. [Kasutatud 2 December 2021].

[15] The PostgreSQL Global Development Group, „PostgreSQL,“ The PostgreSQL Global Development Group, 2021. [Võrgumaterjal]. Available: www.postgresql.org/. [Kasutatud 2 December 2021].

[16] AltexSoft, „Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others,“ AltexSoft, October 2021. [Võrgumaterjal]. Available: www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/. [Kasutatud 2 December 2021].

[17] MongoDB, Inc, „What is NoSQL?,“ MongoDB, Inc, [Võrgumaterjal]. Available: www.mongodb.com/nosql-explained. [Kasutatud 2 December 2021].

[18] J. Hibbard, „The 5 Most Popular Front-end Frameworks Compared,“ 13 April 2021. [Võrgumaterjal]. Available: www.sitepoint.com/most-popular-frontend-frameworks-compared/. [Kasutatud 2 December 2021].

[19] E. You, „Comparison with other Frameworks,“ 2021. [Võrgumaterjal]. Available: vuejs.org/v2/guide/comparison.html. [Kasutatud 2 December 2021].

[20] K. Maaoui, „why Angular is your best choice for your next projects ?,“ 15 October 2019. [Võrgumaterjal]. Available: medium.com/@maaouikimo/why-angular-is-your-best-choice-for-you-next-projects-9d754fb18f91. [Kasutatud 2 December 2021].

[21] K. White, „Top React component libraries (2021 edition),“ 28 May 2021. [Võrgumaterjal]. Available: retool.com/blog/react-component-libraries/. [Kasutatud 2 December 2021].

[22] Docker Inc, „Homepage,“ Docker Inc, [Võrgumaterjal]. Available: https://www.docker.com/. [Kasutatud 2 December 2021].

[23] phoenixNAP Global IT Services, „Docker Image vs Container: The Major Differences,“ phoenixNAP Global IT Services, 31 October 2019. [Võrgumaterjal]. Available: phoenixnap.com/kb/docker-image-vs-container. [Kasutatud 2 December 2021].

[24] Docker Inc, „Overview of Docker Compose,“ Docker Inc, [Võrgumaterjal]. Available: docs.docker.com/compose/. [Kasutatud 2 December 2021].

[25] Apple Inc, „MacOS Monterey,“ Apple Inc, 2021. [Võrgumaterjal]. Available: www.apple.com/macos/monterey/. [Kasutatud 2 December 2021].

[26] JetBrains s.r.o., „IntelliJ IDEA,“ JetBrains s.r.o., 2021. [Võrgumaterjal]. Available: www.jetbrains.com/idea/. [Kasutatud 2 December 2021].

[27] M. Richards, „Layered Architecture,“ O'Reilly, [Võrgumaterjal]. Available: www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html. [Kasutatud 2 December 2021].

[28] C. Richardson, „Pattern: Microservice Architecture,“ [Võrgumaterjal]. Available: microservices.io/patterns/microservices.html. [Kasutatud 2 December 2021].

[29] S. Miteva, „The Concept of Domain-Driven Design Explained,“ 3 July 2020. [Võrgumaterjal]. Available: medium.com/microtica/the-concept-of-domain-driven-design-explained-3184c0fd7c3f. [Kasutatud 2 December 2021].

[30] M. Fowler, „Domain Specific Languages,“ 2010. [Võrgumaterjal]. Available: martinfowler.com/books/dsl.html. [Kasutatud 2 December 2021].

[31] R. T. Fielding, „Architectural Styles and the Design of Network-based Software Architectures,“ Univercity of California, Irvine, 2000.

[32] R. C. Martin, Clean Code, London: Pearson Education, 2009.

[33] S. Millington, „A Solid Guide to SOLID Principles,“ 18 May 2021. [Võrgumaterjal]. Available: www.baeldung.com/solid-principles. [Kasutatud 2 December 2021].

[34] IBM, „Why software testing is important,“ IBM, [Võrgumaterjal]. Available: https://www.ibm.com/topics/software-testing. [Kasutatud 2 December 2021].

[35] S. Bose, „Testing Pyramid : How to jumpstart Test Automation,“ BrowserStack, 2020 January 21. [Võrgumaterjal]. Available: www.browserstack.com/guide/testing-pyramid-for-test-automation. [Kasutatud 2 December 2021].

[36] S. Kolodiy, „Unit Tests, How to Write Testable Code and Why it Matters,“ TopTal LLC, [Võrgumaterjal]. Available: www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters. [Kasutatud 2 December 2021].

[37] S. Faber, „Mockito Framework Site,“ [Võrgumaterjal]. Available: site.mockito.org/. [Kasutatud 2 December 2021].

[38] M. Fowler, „IntegrationTest,“ 16 January 2018. [Võrgumaterjal]. Available: martinfowler.com/bliki/IntegrationTest.html. [Kasutatud 2 December 2021].

[39] Atlassian, „Confluence,“ Atlassian, 2021. [Võrgumaterjal]. Available: www.atlassian.com/software/confluence. [Kasutatud 2 December 2021].

[40] D. Raphael-Renne, „What is regression in software development?,“ 18 August 2021. [Võrgumaterjal]. Available: www.gratasoftware.com/what-is-regression-in-software-development/. [Kasutatud 2 December 2021].

[41] VMware, Inc, „Spring Boot,“ VMware, Inc, 2021. [Võrgumaterjal]. Available: spring.io/projects/spring-boot. [Kasutatud 2 December 2021].

[42] Microsoft Inc, „A tour of the C# language,“ Microsoft Inc, 2021. [Võrgumaterjal]. Available: docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 2 December 2021].

[43] JetBrains, „Kotlin Programming Language,“ JetBrains, [Võrgumaterjal]. Available: https://kotlinlang.org/. [Kasutatud 2 December 2021].

[44] Joyent Inc, „About NodeJs,“ Joyent Inc, [Võrgumaterjal]. Available: nodejs.org/en/about/. [Kasutatud 2 December 2021].

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

Mina, Anton Sauh

Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose, mille juhendaja on Andres Käver ja Jevgeni Družkov reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni; üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

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.

Lisa 2 – Viited lähtekoodile

Viide tagarakenduse lähtekoodile:

https://github.com/antonsauh/thesis_backend

Viide eesrakenduse lähtekoodile:

https://github.com/antonsauh/thesis_frontend