TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

Targa kasvuhoone prototüübi veebirakenduse loomine

Creating a web application for a smart greenhouse prototype

TELEMAATIKA JA ARUKATE SÜSTEEMIDE ÕPPEKAVA LÕPUTÖÖ

Üliõpilane: Kristina Mamai
Üliõpilaskood: 166129
Juhendaja: Natalja Maksimova, lektor

Kohtla-Järve, 2021

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"...." ..... 20.....

Autor: .....
/ allkiri /

Töö vastab rakenduskõrgharidusõppe lõputööle/magistritööle esitatud nõuetele
"...." ..... 20.....

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud
"...." ..... 20.....

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Kristina Mamai (sünnikuupäev: 13.08.1997)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

"Targa kasvuhoone prototüübi veebirakenduse loomine", mille juhendaja on Natalja Maksimova,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas 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 punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

TalTech Inseneriteaduskond Virumaa kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Kristina Mamai, 166129EDTR

Õppekava, peaeriala: EDTR17/18 – Telemaatika ja arukad süsteemid

Juhendaja(d): lektor, Natalja Maksimova, natalja.maksimova@taltech.ee

Lõputöö teema:

(eesti keeles) Targa kasvuhoone prototüübi veebirakenduse loomine (inglise keeles) Creating a web application for a smart greenhouse prototype

Lõputöö põhieesmärgid:

1. Targa kasvuhoone süsteemi loomine protsessi jälgimiseks ja kontrollimiseks

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Uurida, kuidas töötasid targad või automatiseeritud kasvuhooned tarkvara ja riistvara arhitektuuri osas.	15.02.2021
2.	Andurite uurimine, analüüsimine ja valimine kasvuhoone jaoks. Andmebaasi valimine ja loomine.	01.03.2021
3.	Prototüübi välja töötamine, mis võimaldab saata väärtusi andurist. Programmeerimise uurimine *node js* platvormile veebirakenduse loomiseks.	08.03.2021
4.	Targa kasvuhoone prototüübi välja töötamine, mis võimaldab saata väärtusi andurist ja saada käske kasvuhoones aktiveerimiseks (Backend)	29.03.2021
5.	Veebirakenduse loomine protsessi kaugjälgimiseks ja kaugjuhtimiseks targast kasvuhoonest	19.04.2021
6.	Targa kasvuhoone 3D-mudeli koostamine üldiseks visualiseerimiseks	01.05.2021

7.	Lõputöö vormistamine	10.05.2021

Töö keel: eesti keel **Lõputöö esitamise tähtaeg:** "28"mai 2021a

Üliõpilane: Kristina Mamai "....."..... 20.....a
/allkiri/

Juhendaja: Natalja Maksimova "....."..... 20.....a

SISUKORD

5

EESSÕNA 7

LÜHENDITE JA TÄHISTE LOETELU 8

SISSEJUHATUS 9

1 PROBLEEMI JA LAHENDAMISE KIRJELDUS 11

2 PROTOTÜÜBI ARENDAMINE 13

2.1 Ristvara komponendid 13

2.1.1 Andurid 13

2.1.2 Täiturid 14

2.1.3 Juhtimine 14

2.1.4 Komponentide hind 15

2.2 Andmebaas 16

2.2.1 Andmebaasi tabelid ja skeem 17

2.2.2 Andmete viimine Arduinost andmebaasi 21

3 VEEBIRAKENDUSE ARENDAMINE 23

3.1 Tarkvara valik 24

3.2 Backend 24

3.2.1 routes(app.get/app.post) 25

3.2.2 Andmebaasi API 25

3.3 Frontend 27

3.3.1 Kasutajad 28

3.3.2 Administraator 29

3.3.3 Bootstrap disain 30

3.3.4 Ajax päringud 32

3.3.5 Täiturite juhtimine 32

KOKKUVÕTE 34

SUMMARY 35

KASUTATUD KIRJANDUSE LOETELU 36

Programmijuht: Žanna Gratšjova "....."..... 20.....a

/allkiri/

EESSÕNA

Käesoleva lõputöö teema on põhinenud erialapraktika ülesandel. See projekt pakub välja prototüübi, mida saab muuta tavalise kasvuhoone nutikaks kasvuhooneks. Töö käigus oli loodud veebirakendus protsesside jälgimiseks targas kasvuhoones. Rakenduse loomiseks kasutati NodeJS platvormi ja ExpressJS raamistikku, JavaScripti programmeerimiskeel. Nutika kasvuhoone prototüüp põhineb Arduinol. Andmete salvestamiseks kasutati PostgreSQL andmebaas.

Võtmesõnad: prototüüp, veebirakendus , rakenduskõrgharidusõppe lõputöö

LÜHENDITE JA TÄHISTE LOETELU

Angular *JavaScript* veebirakenduste raamistik

API *Application Programming Interface*. Rakendusprogrammiliides

ARM Protsessor

Backend Tagaarendus

AD Andmebaas

bootstrap Tööriistakomplekt disaini jaoks

CSS *Cascading Style Sheets.* Kaskaadlaadistik, märgistuskeel

EJS *Express Java Script*

Frontend Eesarendus

HTML *Hypertext Markup Language*

HTTP *Hypertext Transfer Protocol*

JavaScript Programmeerimis keel

Json *JavaScript Object Notation*

MongoDB Andmebaasi süsteem

NodeJs Tarkvara platvorm

React *JavaScripti* raamistik

SQL *Structured Query Language.* Struktuur päringukeel

VueJs *JavaScripti* raamistik

Xml *Extensible Markup Language. L*aiendatav märgistuskeel

SISSEJUHATUS

Kaasaegses maailmas kasutatakse üha enam nutikaid asju, et vähendada inimese kulutatud energiahulk ja aeg. Targa kasvuhoone süsteemi abil võib muuta oma elu palju lihtsamaks. Automaatne kastmissüsteem kontrollib mulla niiskustaset, akna avamise süsteem hoiab soodsat temperatuuri, mis on taimede kasvu olulisteks teguriteks. Inimene ei saa alati hoida näitajaid normaalses vahemikus, aga tark süsteem saab seda maksimaalse täpsusega kohandada

See projekt valiti selle teema asjakohasuse ja huvi põhja. Alguses oli see praktiline töö esimesel kursusel, kus autor töötas õppestendiga, mis simuleeris nutikat kasvuhoonesüsteemi ja oli loodud ainult hariduslikel eesmärkidel. Selles süsteemis kasutati valmislahendusi. Mida rohkem autor selle projektiga tegeles, seda huvitavamaks see muutus. Seda projekti otsustati jätkata, et luua juhtimis- ja seiresüsteem, mis suudaks muuta tavapärase kasvuhoone nutikaks kasvuhooneks

Meie maailma ei saa ette kujutada ilma tehnoloogiata. Nutikas kodu pole enam paljude jaoks fantaasia olnud, nii et üha rohkem inimesi loob endale targad kodud, pakkudes seeläbi teatud olukordades teatud mugavust. Nutikad robotid toimetavad posti suurlinnadesse ja robottolmuimejad puhastavad meie kortereid ilma meie osaluseta. Kuid on üks nutisüsteemide haru, mis Eestis nii laialt levinud ei ole - see on "Tark Kasvuhoone". Pärast turu analüüsimist võib öelda, et Eestis pakutakse peamiselt nutikaid kasvuhooneid tootmiseks, kuid lihtsamaid süsteeme ja tavapäraseid tellitavaid kasvuhooneid nõudvaid süsteeme on vähe. Pakutud süsteemide hind on kõrge või ei ole määratud.

Selle töö põhieesmärgiks on luua välja targa kasvuhoone prototüübi ja prototüübi veebirakendust. Eesmärgi saavutamiseks jagati töö järgmisteks ülesanneteks:

- Uurida, kuidas töötavad targad või automatiseeritud kasvuhooned tarkvara ja riistvara arhitektuuri osas.
- Andurite uurimine, analüüsimine ja valimine kasvuhoone jaoks.
- Andmebaasi valimine ja loomine.
- Prototüübi välja töötamine, mis võimaldab saata väärtusi andurist.
- Programmeerimise uurimine *node js* platvormile veebirakenduse loomiseks.
- Targa kasvuhoone prototüübi välja töötamine, mis võimaldab saata väärtusi andurist ja saada käske kasvuhoones aktiveerimiseks (Backend)

- Veebirakenduse loomine protsessi kaugjälgimiseks ja kaugjuhtimiseks targast kasvuhoonest

Tehtud töö tulemusena tuleb veebirakendus, mille abil võib jälgida selliseid protsesse nagu õhku temperatuur ja niiskus, mulla niiskus, valgustustase, ja samuti juhtida akna avamise protsessi ja reguleerida mulla niiskust.

1 PROBLEEMI JA LAHENDAMISE KIRJELDUS

Paljudele inimestele pole aiandus mitte ainult töö, vaid on ka hobi. Taimede või köögivilja kasvamiseks on vaja pidevat kontrolli ja tähelepanu. Mulla temperatuuri ja niiskuse taseme hoidmine võtab palju aega. Selleks, et tõhustada juhtimist ning saavutada temperatuuri ja niiskuse suur täpsus on vaja ülejäänud protsessid automatiseerida. Tänu automatiseerimisele on vaja kasvuhoonet kontrollida palju harvemini.

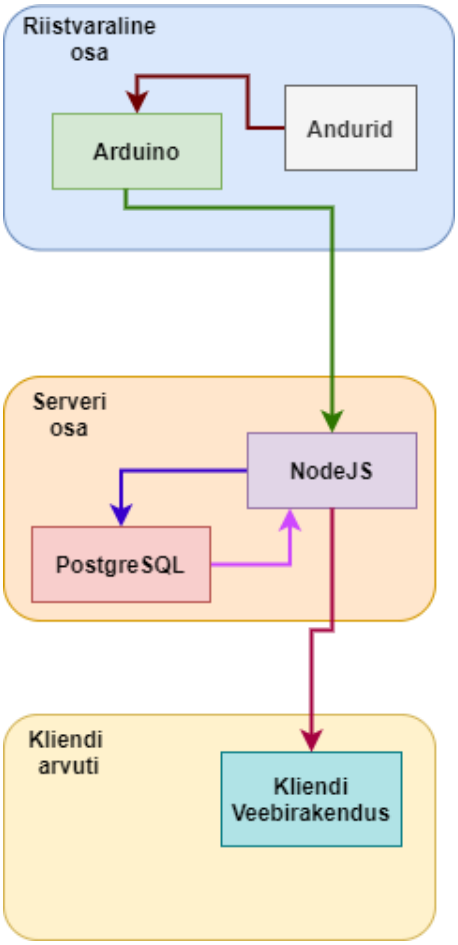
Tõstatatud probleemi lahendamiseks peaks tulevasel kasvuhoonel olema järgmine funktsionaal:

- kasvuhoone ventilatsioon õhutemperatuuri tõusmisel (ukse ja akna avamine sõltuvalt välis- ja sisetemperatuuri erinevusest);
- akna sulgemine;
- taimede automaatne kastmine sõltuvalt mulla niiskuse näitajast;
- automaatne valgustuse reguleerimine sõltuvalt valgustatuse näitajatest;
- mõõtmiste tulemuste kogumine ja regulaatorite juhtimine; mõõtmiste tulemuste saadud tulemuste kuvamine.

Kasvuhoone automatiseerimiseks tuleb integreerida andurid. Kõik andurid ühendatakse juhtimise komponentidega. Juhtimise komponendi kaudu võib juhtida protsessi ja luua internetiühendust.

Joonis1.1 Andmeedastuse näide

Arduino mikrokontroller võtab vastu andmed anduritelt ning saadab need http-päringu abil serverisse. Server võtab andmed vastu ja saadab need andmebaasi. Pärast seda saadetakse andmed veebirakendusse kasutaja isikliku võtme abil, andmed sorteeritakse iga kasutaja jaoks eraldi.



Joonis 1.2 Prototüübi skeem

See arhitektuur on paljude IoT-süsteemide standardlahendus, IoT (asjade Internet, Internet of Things) on seadmed, mis on ühendatud sama võrguga ja suhtlevad omavahel.[1] Näiteks nutikell ja telefon. Nutikella abiga on võimalik vastata kõnedele ja saata sõnumeid, kuid tingimusel, et need asuvad samas Interneti-võrgus või on omavahel Bluetoothi kaudu ühendatud. IoT kasutamise peamine tingimus on Interneti-ühenduse olemasolu.

Samuti kasutatakse nutikasvuhoone andmete jälgimiseks mobiilirakendusi, näiteks üks populaarsemaid ja lihtsamaid valmisrakendusi on Blynk https://blynk.io. See on valmis süsteem, kuid see ei sobi äriliseks kasutamiseks. Kuna Blynk nõuab äriliseks kasutamiseks suurt tasu, saab seda rakendust kasutada ainult prototüüpide jaoks.

2 PROTOTÜÜBI ARENDAMINE

Selles peatükis käsitletakse süsteemi elektroonilisi komponente, mis on vajalikud kasvuhoone funktsionaalsuse tagamiseks. Kasvuhoonele esitatud nõuded olid määratletud eespool ja kasvuhoone automatiseerimiseks on vaja:

- temperatuuriandur, mille andmete alusel toimub kasvuhoone uste avamine ja sulgemine, sest taimedel ei pea olema liiga jahe ega ka palav keskkond. Kasvuhoone sisese temperatuuri jälgimiseks on piisav ühe temperatuurianduri mõõtmistulemused.

- pinnase niiskusandur, mille andmete alusel lülitub veepump sisse taimede kastmiseks. Kuna mulla kuivatamine ei ole ühtlane protsess, millele mõjub päikese intensiivsus ja varjust, siis on vaja vähemalt 2 niiskusandurit erinevates kohtades mõõtmiseks;

- valgusandur, mille andmete alusel LED-lint lülitub sisse ja välja, tagades soovitud valgustatuse taseme. Valguse taseme reguleerimine on oluline taime toitainetega varustamisele. Kuna rohelised taimed saavad oma eluks vajalikud toitained ise sünteesida ja energiat selleks saavad päikselt [2]. Fotosünteetiliselt aktiivne kiirguse kättesaadavuse liigne limiteeritus pärsib taimede kasvu olenemata vegetatsiooni liigist [3]. Valgustatuse välise taseme jälgimiseks on piisav üks valgusandur.

2.1 Ristvara komponendid

2.1.1 Andurid

Andur on seade, mis muudab signaaliks väliskeskkonna näidud, nagu temperatuur, valgus, liikumine.

Elektrotehnika põhjal tehtud andureid nimetatakse elektroonilisteks anduriteks. Eraldi andur saab mõõta (kontrollida) korraga ühte või mitut füüsilist suurust.

Selles töös kasutatakse andureid mulla temperatuuri ja niiskuse, õhu temperatuuri ja niiskuse, valgustatuse taseme jälgimiseks, et neid andmeid kontrollida ja hoida normi piirides.

DHT11

Pinnase temperatuuri ja niiskuse andur. Selle anduri eeliseks on niiskuse mõõtmine vahemikus 0% kuni 100% ja temperatuuri mõõtmise vahemik 0 °C kuni 60 °C (mõõtetäpsus -0,5). Samuti on andur võimeline tegema ühe mõõtmise 2 sekundi jooksul [4]

Resistiivne niiskusandur

Seda kasutatakse mulla niiskuse mõõtmiseks. Seda tüüpi andurid on identsed.

Valgusandur MLG5516B

Valgusandur või teise nimega fototakisti. Kasutatakse valgustatuse taseme määramiseks. Peamised eelised on kiire reageerimine ja madal elektritarbimine [5]

2.1.2 Täiturid

Täiturid on ajamid, mis vastutavad nutikasvuhoone protsesside automatiseerimise eest.

Servomootor MG995-180

Servomootor sobib raskemate mehhanismide, näiteks uste ja akende avamiseks, tänu metallkäigukastile on ta omas klassis kõige võimsam servomootor. [6]

LED-lint

Kasutatakse nutikasvuhoones lisavalgustuse jaoks. Lülitub automaatselt sisse, kui valgustatuse tase on ei ole piisav.

Veepump

Kasutatakse taimede automaatseks kastmiseks. Töötab kahes režiimis: automaatne ja käsitsijuhtimine. Automaatse mulla niiskustaseme määramisega, kui niiskuse tase on alla normi - hakkab pump tööle ja kui andur näitab, et niiskuse tase on normaalne - lülitub see välja. Käsitsijuhtimise režiimis saate pumba ise sisse lülitada. [6]

2.1.3 Juhtimine

Mikrokontroller Arduino MEGA 2560

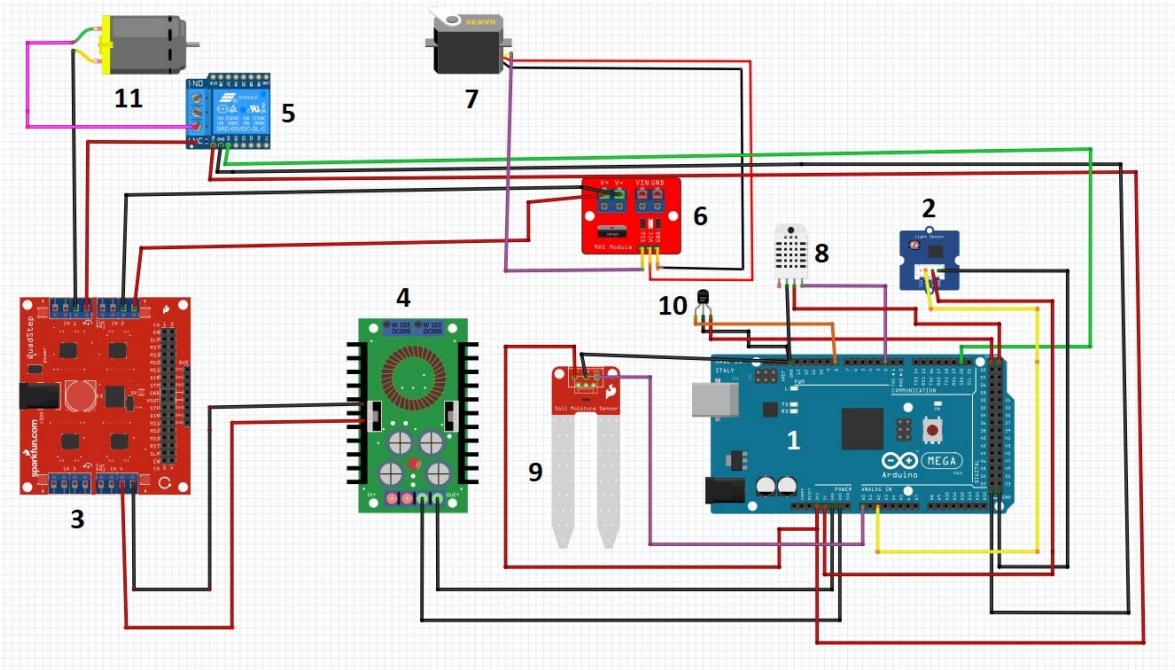
Mikrokontroller või teisiti nimetades süsteemi aju vastutab projekti protsesside ja komponentide juhtimise eest. Arduino MEGA 2560 mikrokontrolleri peamine eelis võrreldes teiste Arduino seeriatega on 54 digitaalse sisendi/väljundi ja 16 analoogse sisendi/väljundi olemasolu, mis on oluliseks faktiks suurte projektide koostamisel. Selles projektis nii paljusid ei kasutata, kuid kui taimi on rohkem, siis on lisaks vaja temperatuuri ja pinnase niiskuse andureid ning pumpasid. Võrreldes teiste tootjate mikrokontrolleritega on see valik kõige odavam ja praktilisem. Selliste projektide jaoks kasutatakse ka mikroprotsessorit RaspberryPi, mis erineb Arduinost sisendkontaktide arvu, väljundkontaktide ja programmeerimiskeele poolest. [7]

Arduino Due

Kõige esimene plaat põhines 32-bitisel ARM-tuumaga mikrokontrolleril, millel on sarnaselt Arduino MEGA 2560-le 54 digitaalset sisendit/väljundit ja neli

analoogsisendit/väljundit vähem (12 analoogsisendit/väljundit). Kasutatakse laiendusplaadina pin-de arvu suurendamiseks. [6]

Joonisel 2.1 on toodud lülitusskeem kõigi komponentide ühendamise kohta ühtsesse vooluringi. Joonis oli tehtud Fritzingis



Joonis2.1 Ühendusskeem: (1) kontroller Arduino MEGA 2560; (2) Valgusandur; (3) Kruviklemmühendustega toite jaoturplaat Power Hub C1; (4) Reguleeritav alalisvoolu pingealaldi LM2596; (5) Releemoodul kahe lülitatava kontaktiga; (6) Jõutransistori MOS moodul; (7) Servomootor MG995-180; (8) DHT11; (9) Resistiivne niiskusandur; (10) LED-lint; (11) Pump

2.1.4 Komponentide hind

Prototüübi kogumaksumus moodustab 236,30 eurot. Autor arvestas komponentide ühendamiseks vajalike juhtmete, pistikplokkidega, toite süsteemi töötamiseks ja MOS moodul suure koormusega juhtimiseks . Tabelis 1 on toodud iga komponendi hind, kui see tellitakse Eestist (hind 1) ja madalaim hind teistest riikidest soetamisel (hind 2).

Tabel 2.1 Komponentide hind

Komponent	Hind 1	Hind 2	Kirjeldus
Kontroller Arduino MEGA 2560	40 €	10€	Andurite juhtimine
Arduino Due	26 €	12 €	Plaat töökiiruse suurendamiseks
Arduino Ethernet	35 €	5 €	Laiendusplaat Etherneti

Komponent	Hind 1	Hind 2	Kirjeldus
			külge lülitamiseks
Valgusandur MLG5516B	5 €	1 €	Loeb valgustatuse taset
Kruviklemmühendustega toite jaoturplaat Power Hub C1	5 €	4 €	Plaat toite jaotamiseks
Reguleeritav alalisvoolu pingealaldi LM2596	5 €	1 €	Stabiliseerib pinget
Releemoodul kahe lülitatava kontaktiga	3 €	2 €	Veepumba juhtimine
Jõutransistori MOS moodul	13 €	0.30 €	Pumba ja LED-lindi toide
Servomootor MG995-180	20 €	3 €	Akende avamine ja sulgemine
LED-lint	4 €	1 €	Lisavalgustus
Temperatuuri- ja niiskusandur DHT11	10 €	3 €	Õhutemperatuuri ja niiskuse mõõtmine
Resistiivne niiskusandur (x2)	5.30 €	2 €	Mulla niiskuse mõõtmine
Toiteplokk Adapter 36W (12V 3A)	15 €	3 €	Kasvuhoone toide
Veepump 12V kummist kinnitustega	26 €	4 €	Taimede kastmine
Pistikuga ühendus klemmliist Arduino	6 €	2 €	Arduino ühendamiseks anduritega
Ajam	18 €	10 €	Arduino ühendamiseks anduritega
Summa kokku:	**236.30 €**	**63.30 €**	

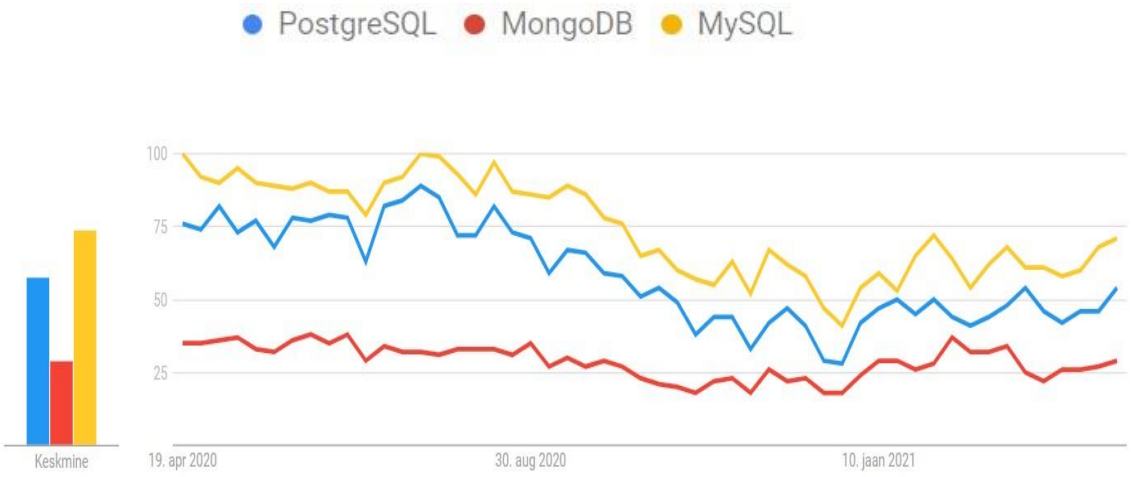
2.2 Andmebaas

Andmebaas on selle töö üks olulisemaid osi. Peamised kriteeriumid andmebaasi valimiseks on:

- ühilduvus NodeJS Expressi kasutamisel;

- suurte andmemahtudega töötamine; struktureeritud andmete hoidmine.

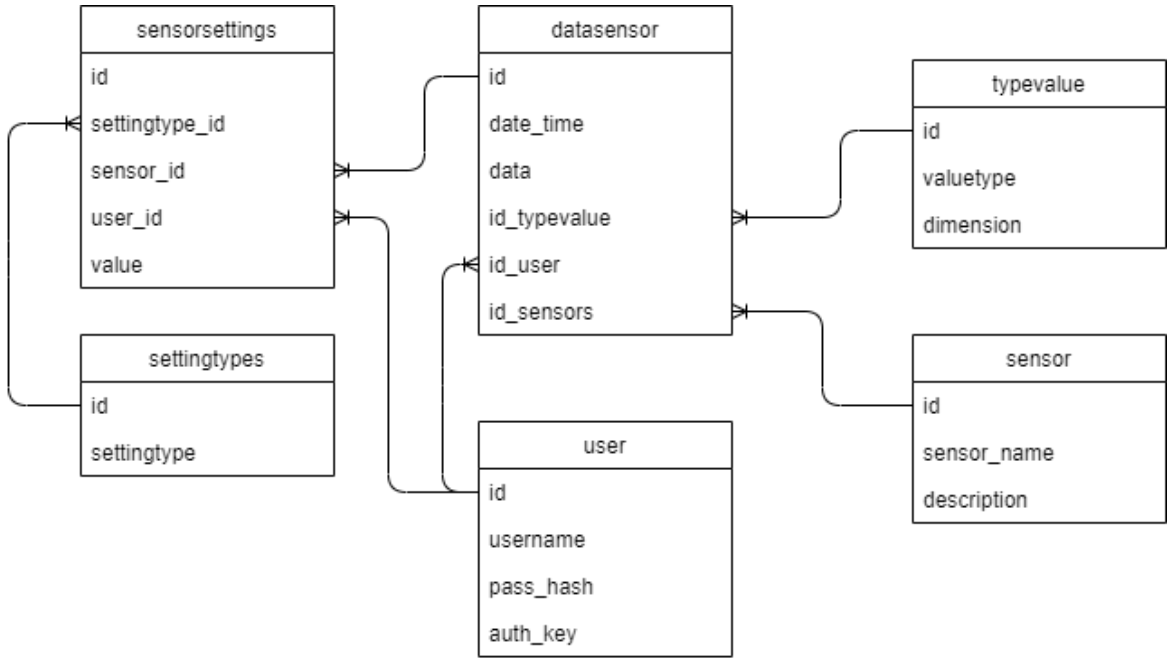
Võrreldes MongoDB ja MySQL-iga kasutatakse PostgreSQL-i suurte ja keeruliste analüütiliste protsesside jaoks. PostgreSQL on mitmete praktiliste funktsioonidega, nagu laiendatavus, pärilikkus ja multifunktsionaalsus, mis aitavad keerukat andmebaasi hallata. Võrreldes MongoDB-ga on PostgreSQL oma struktuuri ja laia kasutusvõimaluste poolest taas populaarsust kogumas. See on olnud saadaval juba pikka aega. Arendajatel on selle tehnoloogia jaoks rohkem võimalusi kui MongoDB-ga [8]. PostgreSQL-i eelis MySQL-i ees on see, et see toetab paljusid erinevaid andmetüüpe, sealhulgas JSON, hstore ja XML, samuti ei ole PostgreSQL-is vaja sümbolite komplekte ja jadasid teisendada UTF-8 kodeeringusse.



Joonis 2.2 PostgreeSQL, MongoDB ja MySQL võrdlus [9]

2.2.1 Andmebaasi tabelid ja skeem

Antud nutikasvuhoone jaoks kasutatakse andmebaasi kolmandat normaalset vormi. Igal andmekogumil on oma eraldi tabel, välja jäetakse ka eraldi tabelites olevad duplikaatrühmad ja iga seotud andmete komplekt tuvastatakse esmase kirje ID kaudu. Kõik andurite ja täiturite andmed hoitakse andmebaasis, samasuguselt andmebaasis hoitakse andurite ja täiturite nimed ja mõõtühikud. Kuna selline veebirakendus on kasutajate rakendus koostatakse samamoodi kasutaja kontod, andmete individualiseerimiseks koostatakse ka personaalne kood, igal kasutajal oma kood.



Joonis 2.3 Andmebaasi skeem

Andmete süstematiseerimiseks loodi 6 tabelit:

Tabel **Datasensor**(id, date_time, data, id_typevalue, id_user, id_sensors) -Täiturite ja andurite näite salvestamiseks (vt Joonis 2.4), kus

- id - iga rea isiklik esmane võti;
- date_time - andmete edastamise kellaaeg;
- data - andurite näidud;
- id_typevalue - anduri andmete tüübi võti;
- id_user – kasutaja võti;
- id- sensors - anduri võti.

	id [PK] integer	date_time timestamp without time zone	data double precision	id_typevalue integer	id_user integer	id_sensors integer
1	14	2021-04-13 16:00:29.816541	39.3	2	1	4
2	15	2021-04-13 16:00:36.706325	34.3	2	1	3
3	16	2021-04-13 16:00:57.505171	23.4	2	1	1
4	17	2021-04-13 16:01:02.823538	44	3	1	1
5	18	2021-04-13 16:01:50.512922	57.3	3	1	5
6	19	2021-04-13 16:01:57.447473	54	3	1	6
7	20	2021-04-13 16:02:16.784914	165	4	1	2
8	21	2021-04-13 16:03:58.81052	165	4	1	2
9	22	2021-04-13 16:09:26.530008	37	3	1	1

Joonis 2.4 Andmebaasi tabeli *datasensor* vaade

Tabel **Sensorsetting** (id, settingtype_id, user_id, sensor_id, value) - täiturite töö või tööta seisundi salvestamiseks (vt Joonis 2.5). Igal täitureil on kaks seisundid, kas töötab (1-värtus) või ei tööta (0-värtus) kus

- id - iga rea isiklik esmane võti;
- settingtype_id - võti andmetüübile ja elementidele, milles värv muutub
- user_id - kasutaja võti;
- sensor_id – täituri tüübi võti;
- value - täiturite näidud.(0- töö seisund või 1- või tööta seisund).

	id [PK] integer	settingtype_id integer	user_id integer	sensor_id integer	value text
1	15	1	1	12	0
2	14	1	1	11	0
3	13	1	1	10	0

Joonis 2.5 Andmebaasi table *sensorsetting*

Tabel **Settingtypes** (id, settingtype) - täiturite olekud, selline tabel hoiatab täiturite olekud, tabeli abil võib jälgina millal täitur hakkab tööle ja millal lõpetab, samuti on näha millal täitur ei töötanud(vt Joonis 2.6), kus

- id - iga rea isiklik esmane võti;
- settigtype - täiturite andmetüüp.(hakkab töötada, lõpetab töötada, ei tööta).

	id [PK] integer	settingtype text
1	2	work-start-value
2	3	work-stop-value
3	1	device-work-sta...

Joonis 2.6 Andmebaasi tabel *settingtype*

Tabel **Sensor** (id, sensorname, description) – andureid ja täitureid nimesid ja omadusi salvestamiseks (vt Joonis 2.7), kus

- id - iga rea isiklik esmane võti;
- sensorname - andurite nimetus;
- description – anduri ja täituri kirjeldus, milliseid andmeid see loeb.

	id [PK] integer	sensorname character varying (50)	description text
1	1	DHT11	air temperature
2	2	Light sensor	Light
3	3	DS18B20-1	Soil temperature
4	4	DS18B20-2	Soil temperature
5	5	LM393-1	Soil humidity
6	6	LM393-2	Soil humidity
7	10	Pump	On/Off
8	12	Servo	On/Off
9	11	Light LED Lint	On/Off

Joonis 2.7 Andmebaasi tabel *sensor*

Tabel **Typevalue** (id, valuetype, dimension) - andmete tüübi mõõtühikuid salvestamiseks, (vt Joonis 2.8), kus

- id - iga rea isiklik esmane võti; valuetype - anduri andmete tüüp;
- dimension – andmete tüübi mõõtühik.

	id [PK] integer	valuetype character varying (50)	dimension character varying (50)
1	2	temperature	C°
2	3	humidity	%
3	4	light	E

Joonis 2.8 Andmebaasi tabel *typevalue*

Tabel **User** (id, username, pass_hash, auth_key, is_super) – kasutaja andmeid salvestamiseks. Igal kasutajal oma konto, kus kasutaja võib näha andmeid oma targa kasvuhoonest (vt Joonis 2.9), kus

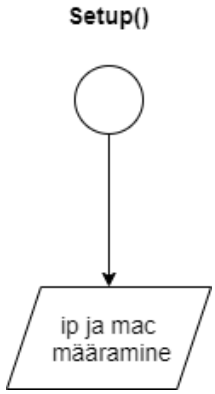
- id - iga rea isiklik esmane võti;
- useraname - kasutajanimi;
- pass_hash - kasutaja parooli räsi, paroole hoitakse krüptitud kujul;
- auth_key – kasutaja personaalne kood;
- is_super - kasutaja veebirakenduse administraatoriõigus.

	id [PK] integer	username character varying (50)	pass_hash character varying (50)	auth_key text	is_super boolean
1	1	t166129	######	exampleauth	false
2	2	admin	######	exampleauth2	true

Joonis 2.9 Andmebaasi table *user*

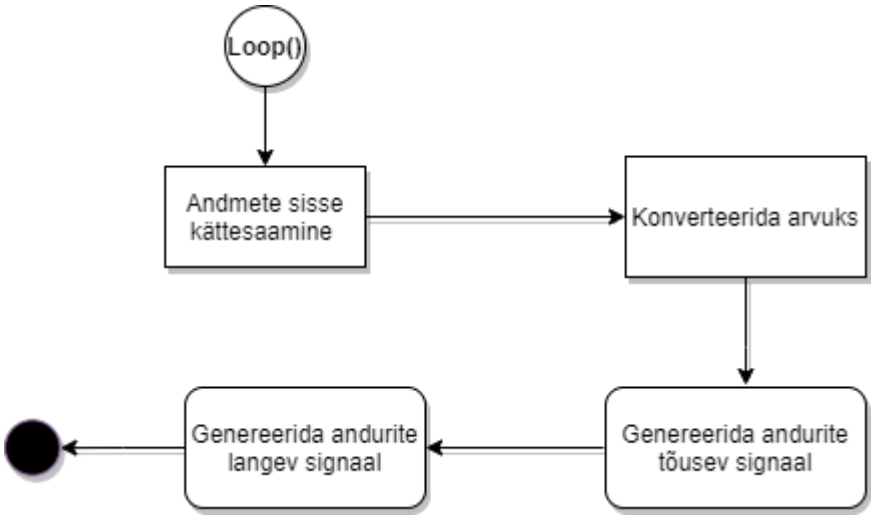
2.2.2 Andmete viimine Arduinost andmebaasi

Selleks, et Arduino hakkaks komponente juhtima ja serverisse andmeid edastama, tuleb see eelprogrammeerida. Arduino kood kirjutatakse C/C++ keeles ja koosneb 3 funktsioonist. Funktsioon **setup ()**, milles lähtestatakse serveri IP-aadress, nutikasvuhoone IP-aadress ja kõik komponendid, mille andmeid või olekuid loetakse.



Joonis 2.10 Skeem *setup*()

Funktsioon Loop(), kus kirjutatakse andmete lugemise ja andmebaasi edastamise protsessi juhtimist. Andmebaasi edastamiseks on kirjutatud funktsioon *requestGetToGrServer*(float value, int valuetype, int sensor, String authkey)mille põhiülesandeks kontrollida greenhouseMonitoringServer(88- lokaalne serveri number) ja moodustada päringut andmete muutumatu algkujul säilitamiseks



Joonis 2.11 Skeem *loop*()

Täiturite töö sõltub andurite mõõtmistulemustest, siis ka kirjutatakse tingimusi veepumba ja led-lambi käivitamiseks. Katsete alusel oli leitud, et pilves päeval valgustuse tase on vahemikus 67% -69% ja mulla niiskuse minimaalne piir on 60%.

```
if (soil_hum_2 < 60)// kui niiskusanduri mõõtmistulemused < 60%
{
    digitalWrite(PUMP, LOW);// veepump on sisse lülitatud
    requestGetToLabWeb(1, "PumpState"); //seisund andmebaasis
  }
    else
{
  digitalWrite(PUMP, HIGH);// kui niiskusanduri mõõtmistulemused vähemalt 60%
  requestGetToLabWeb(0, "PumpState"); //veepump on välja lülitatud
}
```

Joonis 2.12 Veepumba automaatse käivitamise kood

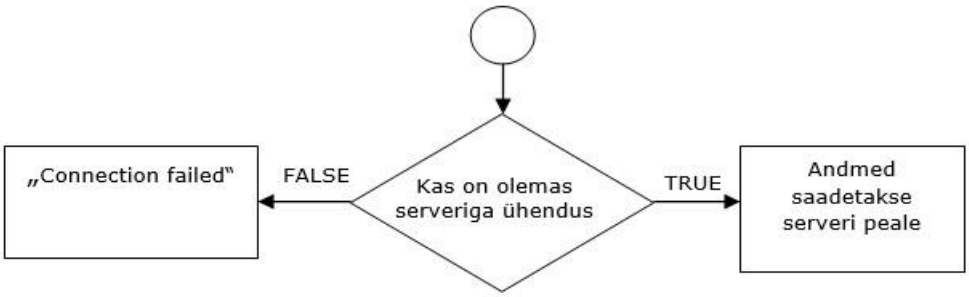
Mõõtmisobjekti mõjul muudab andur oma omadusi ja näitab pinge sõltuvust löögist. Selles etapis on vaja andurite andmete konverteerida protsendiks. Siin konverteerimist toimus järgmiste valemite järgi(joonis 2.12)

```
soil_hum_1 = analogRead(SOIL_SENSOR_1) / 1023.0 * 100.0;
soil_hum_2 = analogRead(SOIL_SENSOR_2) / 1023.0 * 100.0;
float light = analogRead(LIGHT_SENSOR) / 1023.0 * 100.0;
```

Joonis 2.13 Andmete konverteerimine

Funktsioon **requestGetToGrServer()** - andmete, andurite näitude saatmiseks kirjutatakse serverile GET-meetodi abil http-päring. GET-meetodit kasutatakse seetõttu, et see meetod on kiirem kui post päring ja sobib väikese mahuga andmete jaoks. Samuti temperatuuri, niiskuse jms andmed ei ole konfidentsiaalsed, mis vastab GET-meetodi kasutamisele.

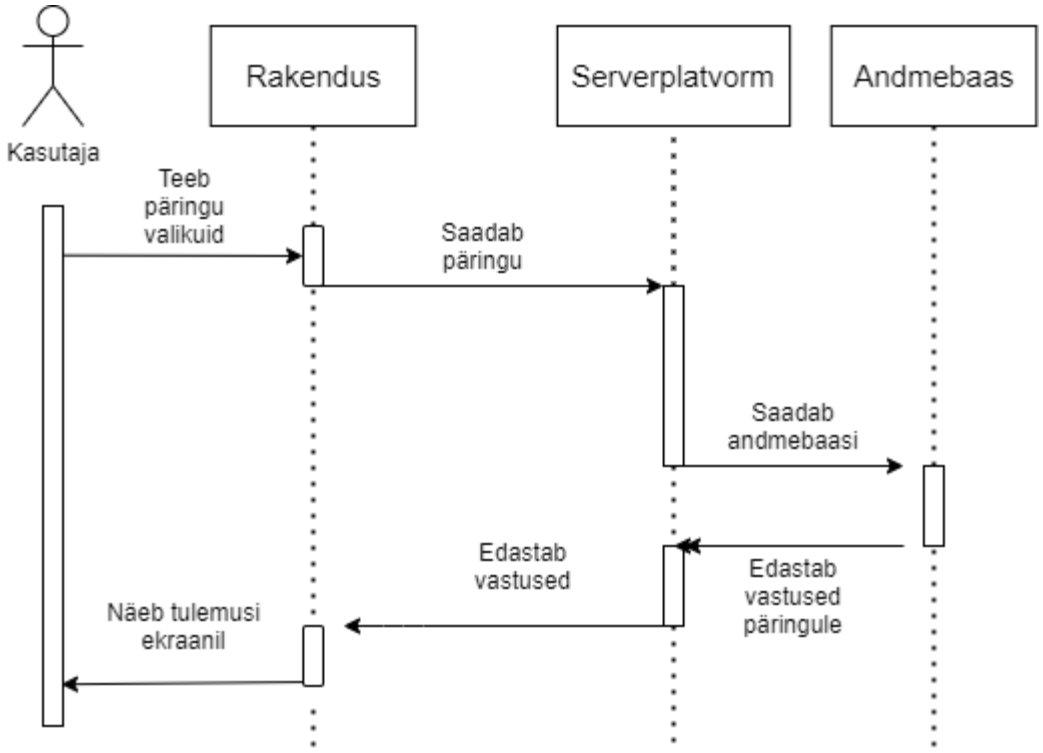
requestGetToGrServer()



Joonis2.14 Funktsiooni *requestGetToGrServer* skeem

3 VEEBIRAKENDUSE ARENDAMINE

Veebirakendus koosneb kahest osast: I - vastutab päringute vastuvõtmise, ülesannete täitmise ja vastuste esitamise eest, mida nimetatakse tagaarenduseks (ing. *backend*). II - vastutab kasutaja poolest rakenduse eest, mida nimetatakse eesarenduseks (ing. *frontend*). Edasi autor kasutab oma töös inglisekeelset terminaloogiat. Kasutaja, rakenduse, serverplatvormi (backend ja frontend) ja andmebaasi andmete liikumise printsiip on näidatud järgnevusdiagrammil (Joonis 3.1).



Joonis 3.1 Andmete liikumine

Selle töö puhul arvestati raamistiku valimisel ühilduvust andmebaasi kasutamisega. Kuna selle kohandatud rakenduse jaoks peab server vastama kiiresti ja õigeaegselt, siis peab server töötama suure kiirusega. Veebirakenduse loomiseks peab raamistik olema kasutajasõbralik ja mitmesuguste kasutatavate moodulitega nagu Ajax ja Bootstrap.

Kasutajale mõeldud veebirakendus peaks olema kõigepealt mugav ja lihtsasti kasutatav. Nii on see rakendus mõeldud kasvuhoonest saadud andmete visualiseerimiseks, andmed peavad tulema kiiresti ja õigeaegselt. Veebirakenduses peaksid andmed olema kuvatud arusaadavas ja mugavas vormis koos võimalusega hallata täiendavaid komponente.

Nagu on näidatud Joonisel 1.2, edastatakse andmed Arduinost serverisse, serverist andmebaasi ja andmebaasist veebirakendusse. Andmete edastamisel peavad olema turvatud isiklikud andmed.

3.1 Tarkvara valik

Veebirakenduse backend-i loomiseks kasutatakse NodeJs-t. Autor valis selle platvormi, kuna see on IoT veebirakenduste jaoks praegu kõige nõutum platvorm. NodeJs võimaldab luua reaalajas rakendusi. NodeJs platvorm kasutab JavaScript-i, mis kompileerib ja töötab väga kiiresti. JavaScripti skripte toetavad peaaegu kõik brauserid ja see on täielikult integreeritud saidi kõigi elementide (HTML ja CSS) ning serveripoolse (Backend) struktuuriga, see võimaldab arvutis veebisaite, kasutaja ei kasuta serverile päringuid, mis vähendab serveri koormust. [10] NodeJS on üks väheseid platvorme, mis pakub muljetavaldavat pakettide süsteemi, mis ühendab tuhandeid JavaScripti teeke ja tööriistu.

Express raamistikku kasutatakse andmebaasi haldamise prototüübi integreerimiseks. Autor valis selle seetõttu, et Expressi toetab Node.js; hästi dokumenteeritud; ühildub EJS-i malliga [11].

Rakenduse liidese loomiseks kasutab autor bootstrap-i, seda kasutatakse saidi kaartide ja navigeerimise loomisel. Autor valis bootstrap-i, kuna need on avatud ja populaarsed mallid. Liidese stiliseerimiseks ja kujundamiseks kasutatakse keeli HTML ja CSS, see on liidese loomise standardne valik.

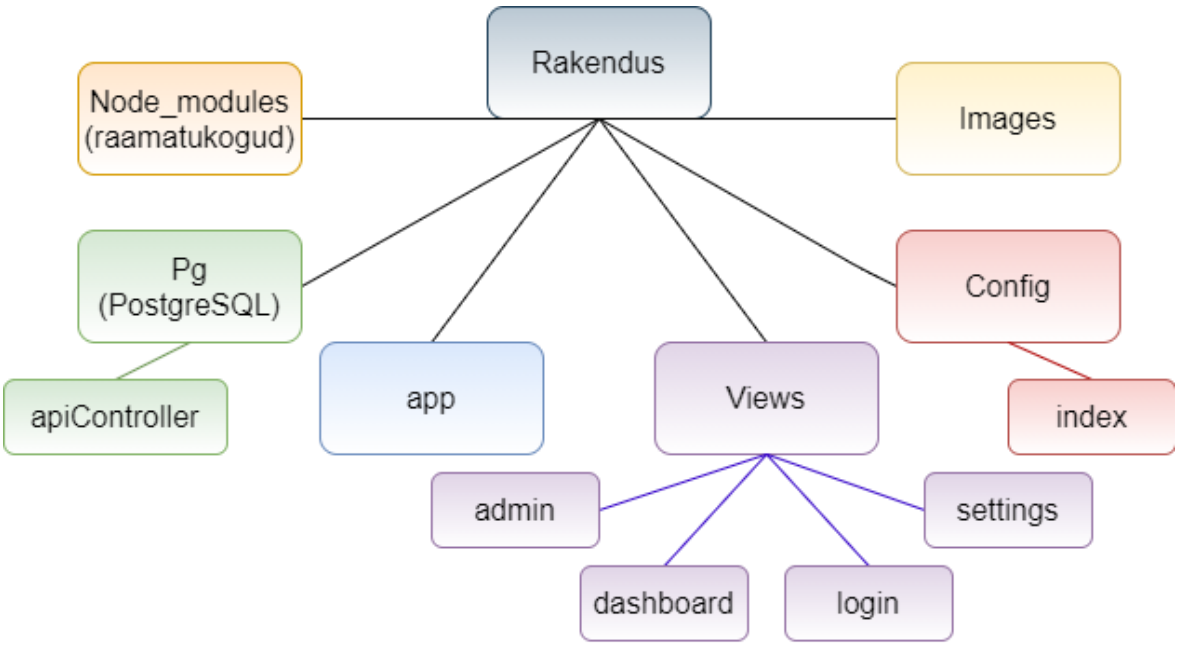
3.2 Backend

Backend on teenuse tarkvara- ja riistvaraline osa, seda võib nimetada kasutaja ja serveri osaga ühendamise protsessiks. Backend saadab päringud andmebaasist andmete saamiseks veebirakendusse. [12]

Esimene etapp on kasutaja autoriseerimine, kontrollitatakse , kas sisestatud parool ja kasutajanimi vastavad andmebaasi andmetele, kui jah siis kasutaja logib sisse ja näeb andmeid oma targa kasvuhoonest

Backend programmikood koosneb 3 põhiosast: andmebaasiga ühendamine, päringute loomine ja tulemuste avamine veebiserveril. Programmikood on jagatud 6 failiks.

- **node_modules** – raamatukogud;
- **apiController.js**- kus on ettevalmistatud API saadetud json päringuteks;
- **app.js**- on rakendusse sisenemispunkt;
- **images**- kus on hoitakse pildid;
- **index**- fail, kirjutatud ühenduse konfigureerimist baasiga ; **views**- kus hoitakse kasutaja küljed.



Joonis 3.2 Rakenduse failid

3.2.1 routes(app.get/app.post)

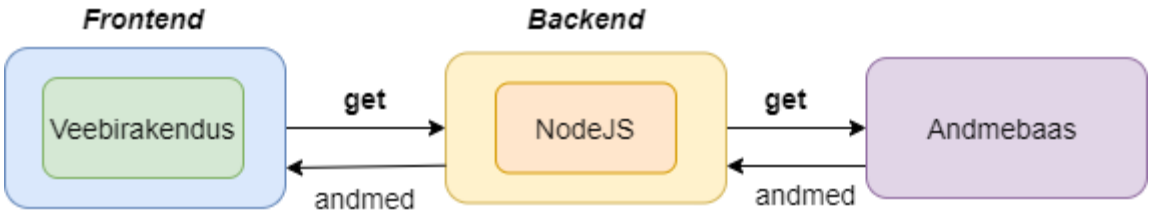
Marsruutimine on veebirakenduse vastus kliendi päringule. Selline päring nagu andmete saamine nutikasvuhoonest (get päring) või kasutaja volitamine veebirakenduse sisestamise kohta (post päring).

Get-meetod küsib andmeid. Selle meetodi abil päringutega saab andmeid ainult hankida. Tingimused get-päringute hankimiseks, päringuid saab vahemällu salvestada, neid saab järjehoidjatesse lisada ja see ei ole vastuolus veebilehe turvalisusega, päringuid kasutatakse ainult andmete hankimiseks. Meetod post saadab andmed määratud ressurssi töötlemiseks, post päringute kasutustingimused, need on konfidentsiaalsed andmed, mida ei saa vahemällu salvestada ja need andmed ei saa olla järjehoidjad. Post päringute andmeid ei ole piiratud märkide arvuga, erinevalt getpäringute andmetest.

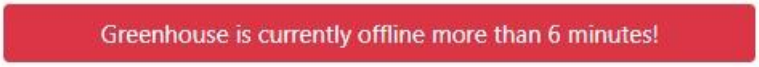
3.2.2 Andmebaasi API

3.2.2.1 Targa kasvuhoone andmed saamine

Andmete hankimiseks andmebaasist saadetakse veebilehe rakendusele get-päring. Andmed saadetakse veebirakendusse iga 2 minutid. Kui veebirakendusse 6 minuti jooksul uued andmeid ei saabu siis ilmub teade kasutaja teadmiseks



Joonis 3.3 Näide get päring



Joonis 3.4 Teade

Serveri osas frontend ja backend programmeeritud protsessid suhtlevad omavahel backend'is loodud API kaudu- see on päringud millised saadetakse andmete saamiseks või edastamiseks. Töö käigul oli loodud kokku 6 API, mis annavad meie andmeteks liigipääse ja tagastavad järgmist infot:

1)get päring arduinost serverile andmete edastamiseks

2)get päring serverist andmebaasile andmete edastamiseks

3)get päring veebirakendust andmebaasile andmete saamiseks

4)post paring kasutajakonto kontrollimiseks

5)post päring veebirakendust andmebaasile täiturite oleku muutumiseks

6)post päring andmebaasist arduinole täiturite oleku muutumiseks

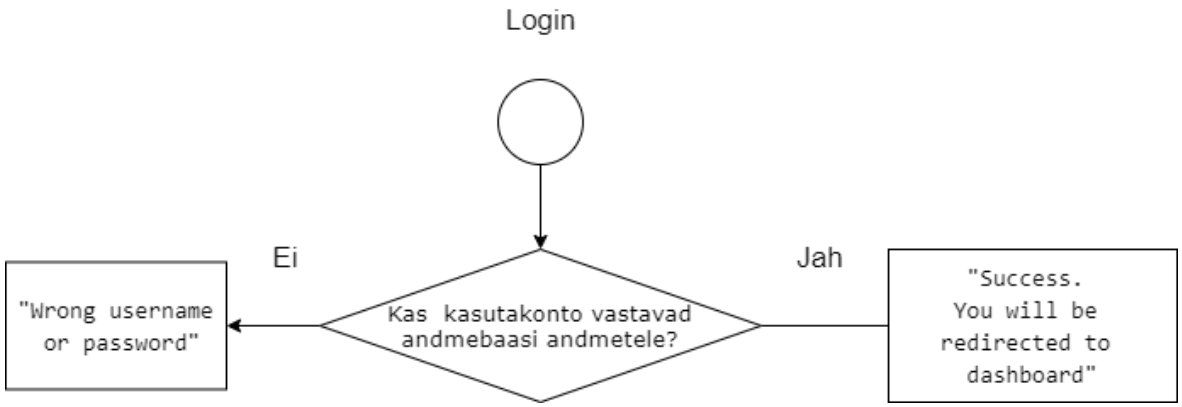
Näiteks joonisel 3.5, kui server on Arduinolt andmed kätte saanud, suunatakse andmed GET päringu abil PostgreSQL andmebaasi.

```
app.get("/api/send", function (req, res) {

    var sensor = req.query.sensor;
    var value = req.query.value;
    var authkey = req.query.authkey;
    var user = req.query.user;
    var type = req.query.type;
    var today = new Date();
    var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();
    var time = today.getHours()+':'+today.getMinutes()+':'+today.getSeconds();
    var datetime = date + ' ' + time;


    db.any(`INSERT INTO datasensor (date_time, id_user, id_sensors, data, id_typevalue)\
    values ('${datetime}',${user},${sensor},${value},'${type}')`)
      .then(function (data) {
        res.json({
          status: "success",
          data: data,
        });
      })
      .catch((err) => {
        res.json({
          description: "Cannot insert. May be values are not provided",
          error: err,
        });
      });
});
```

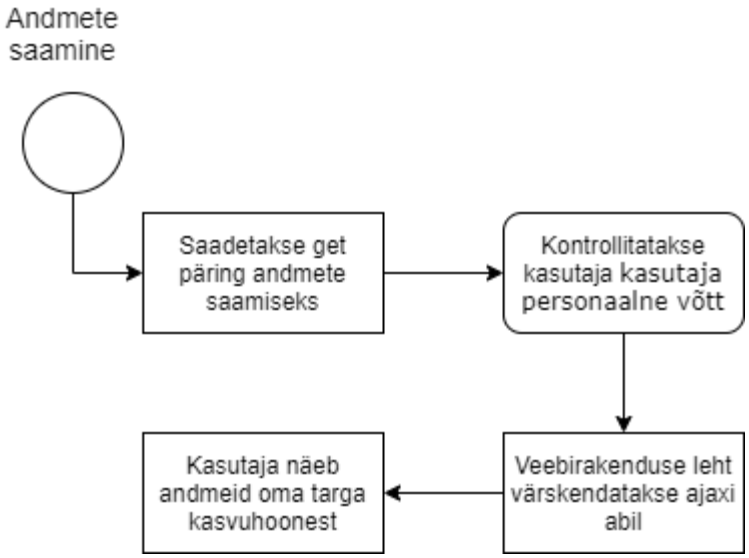
Joonis 3.5 Koodi näide get päring

3.3 Frontend

Frontend on veebirakenduse kasutajaliidese kliendipool. Veebirakenduse kogu funktsionaalne külg on seotud Frontend'iga: veebikaartide navigeerimine, pildid ja kujundus. Frontend'i keskmes on programmeerimiskeeled, näiteks HTML ja CSS, nad vastutavad elementid, stiili ja JavaScripti kuvamise eest, mis loob kasutaja interaktsiooni veebirakendusega [13].



Joonis3.6 Autoriseerimine



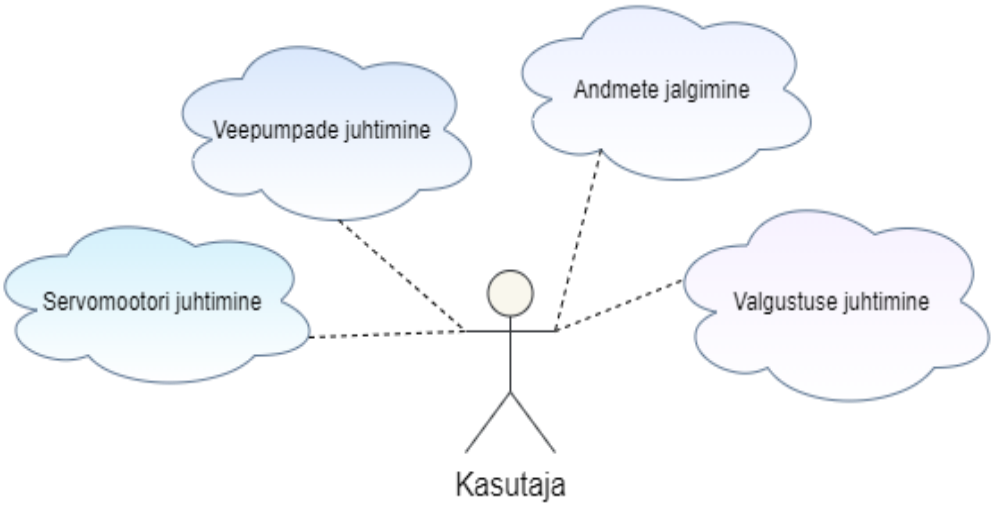
Joonis3.7 Andmete saamine

3.3.1 Kasutajad

Kõik kasutajad on salvestatud andmebaasi. Kasutajanime, parooli ja kasutaja autentimisvõtme loob administraator iga kasutaja jaoks. Nutikasvuhoones andmete isikupärastamiseks on vaja autentimisvõtit.

Autoriseerimine toimub järgmiselt, kasutaja sisestab oma kasutajanime ja parooli ning seejärel klõpsab nuppu "login", seejärel saadetakse serverile post päring, et kontrollida, kas sisestatud parool ja kasutajanimi vastavad andmebaasi andmetele. Kui parool ja kasutajanimi sobivad, kuvatakse teade ja kasutaja logib sisse. Kui ei, siis kuvatakse teade, et sisselogimist ei toimunud

Joonis 3.8 Veebirakenduse sisse autoriseerimine



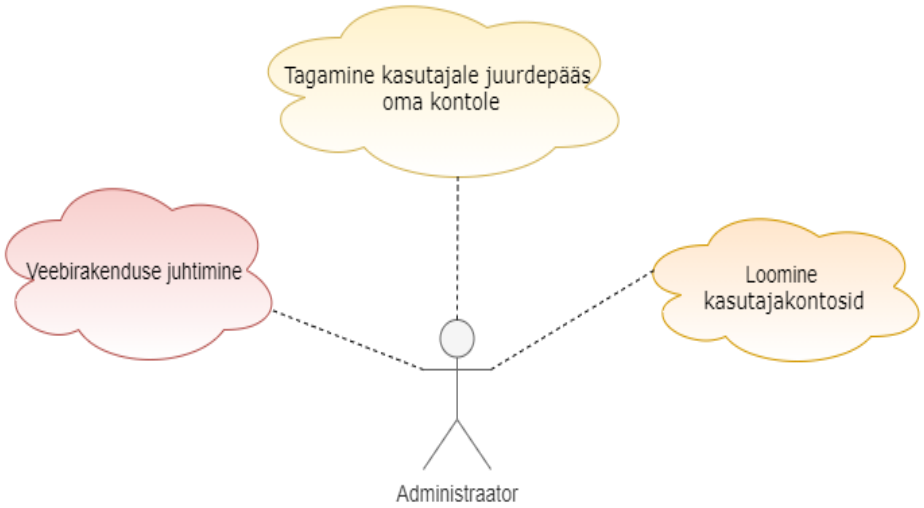
Joonis 3.9 Kasutaja võimalused

3.3.2 Administraator

Veebiadministraator peab olema selleks, et tagada kasutajale juurdepääs oma kontole, selleks luuakse eraldi leht, millele pääseb juurde ainult administraator.
Kasutaja autoriseerimisel kontrollitakse veebi haldamise õigust.

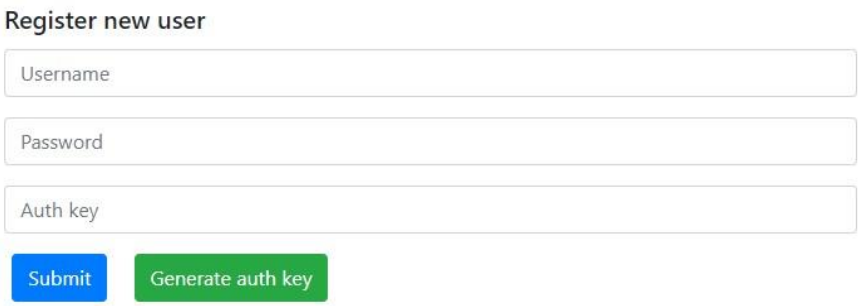
```
if (data["data"][0]["is_super"]){
    window.location = "/admin"
}else{ window.location = "/dashboard"; }
```

Joonis 3.10 Koodi näide administraatori initsialiseerimine



Joonis 3.11 Administraatori võimalused

Kui kasutajal on õigus veebi hallata, avaneb pärast autoriseerimist administraatori leht, kus administraator saab luua kasutajakontosid.



Joonis 3.12 Kasutaja nimi ja salasõna loomine

```
app.post("/api/createnewuser", function (req,res){
  if (req.session.issuper){
    username = req.body.username;
    password = req.body.password;
    auth_key = req.body.auth;
    is_super = false;
    db.any(`INSERT INTO public.user (username, pass_hash, auth_key, is_super)
     values ('${username}','${password}','${auth_key}','${is_super}')`)
      .then(function (data) {
                res.json({
                  status: "success",
                   data: data,
        });
      })
      .catch((err) => {          res.json({
          description: "Cannot insert. May be values are not provided",
           error: err,
        });
      });
  }

    });
```

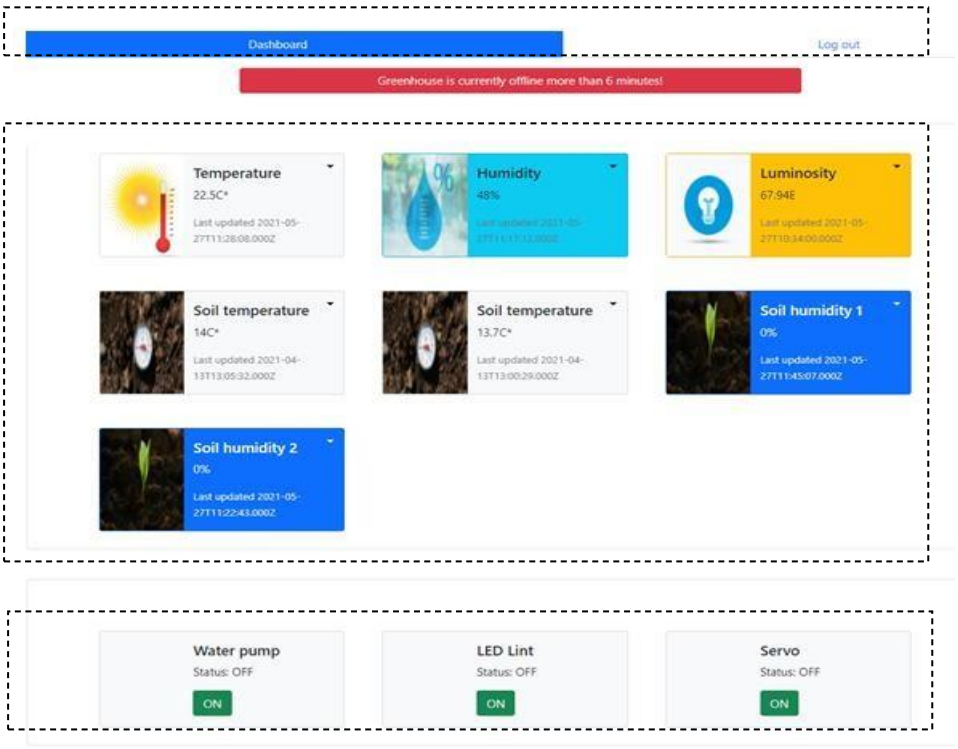
Joonis 3.13 Koodi näide kasutajanime ja salasõna loomine

3.3.3 Bootstrap disain

Bootstrap on tööriistakomplekt veebirakenduse loomiseks. Avatud lähtekoodiga tööriistakomplekt sisaldab veebivormide, nuppude, siltide ja navigeerimisplokkide malle.

Veebirakenduse disaini oli tehtud haarliku stiilil ja minimeeritud, et kasutajale veebirakendus oleks mugav ja lihtsasti kasutatav.

Veebirakenduse aken on jagatud kolmeks osaks (vt Joonis 3.14): 1) navigeerimiskaart, 2) andurite mõõtmistulemuste visualiseerimine ja 3) täiturite juhtimine (käsitsi lüliti).



Joonis 3.14 Veebirakendus

Joonistel 3.15 – 3.17 on veebirakenduse navigeerimikaardi ja anduri tekstivälja koodide näited.

```
<div class="card">
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
      <a class="nav-link active" aria-current="page" href="/dashboard">Dashboard</a>
  </li>
      <a class="nav-link" id="user_logout" href="">Log out</a>
  </li>
      </ul>
</div>
```

Joonis 3.15 Koodi näide bootstrapi navigeerimiskaardi loomine

```
<h5 class="card-title">Humidity</h5>
<p id="hum-value" class="card-text">Unknown</p>
<p class="card-text">
<small id="hum-upd-date" class="text-muted">Last updated never
</small></p>
```

Joonis 3.16 Koodi näide bootstrapi kardi loomine

Joonis 3.17 Pinnase temperatuuri visualiseerimise plokk

3.3.4 Ajax päringud

AJAX on lühend, mis tähendab Asynchronous JavaScript and XML (Asünkroonne JavaScript ja XML) [14]. Asünkroonne päring on selline päring, mis täidetakse taustarežiimis. Kui asünkroonne päring saadetakse, võimaldab brauseri leht kasutajal lehega edasi töötada, ei sega kasutaja veebirakendusega suhtlemist ega pane teda ootama.[15] Selles töös kasutatakse ajax-päringut nii, et uued andmed nutikasvuhoonest pidevalt liiguvad kasutaja veebilehele ja kasutaja ei pea lehte ise värskendama.

Ajaxit kasutatakse ka kasutajate autoriseerimiseks. Kui kasutaja on autoriseeritud, ilmub teade eduka autoriseerimise kohta, kui ei, siis ilmub teade, et sisenemine ebaõnnestus.

```
$.ajax({                                                          1
    url:'/api/login',
    type: 'POST',
    data: {"username":user,"pass":pass},
    success: function(data) {
      if (data["status"] == "success"){                           2
          alert("Success. You will be redirected to dashboard");
          if (data["data"][0]["is_super"]){                       3
              window.location = "/admin"
          }else{ window.location = "/dashboard"; }
      }else{
          alert("Wrong username or password");
      }
    },
    error  : function() {
      console.log('error');
      }
    });
```

Joonis 3.18 Koodi näide ajax kasutamine. (1)Saadetakse post pärig kasutaja nimi ja salasõnu kontrollimiseks(2) Kui parool ja kasutajanimi sobivad, kasutaja näeb teade ja logib sisse, kui ei, siis ilmub teade "Vale nimi või salasõnu"(3)Kontrollitakse veebi haldamise õigust, kui kasutajal on õigust, siis avatakse administraatori lehekülge, kui ei ole siis tavaline kasutaja lehekülge

Lisaks eelnevale antud töös kasutatakse ajax-t andmebaasist uute andmete saatmiseks veebirakendusse, nii et kasutaja ei peaks ise lehte uuesti taaskäivitama.

3.3.5 Täiturite juhtimine

Veebirakenduse kaudu samuti kasutaja võid juhtida täitureid: muuta valgus, avada aknaid ja kasta taimi (vt Joonis 3.19). Kui täituri olek on muutunud siis sadetakse post

päring andmebaasile täituri värtusega: 0- töö seisund või 1- või tööta seisund. Ning andmebaasis muutakse olek saadetakse post päring ja arduinole ka muutakse olek (vt Joonis 3.20).

Joonis 3.19 Täiturite juhendamine

```
$('.devicebtn').on('click', function () {
    $.ajax({
        url: '/api/changedevicestatus',
        type: 'POST',
        data: {id: $(this).attr('id').replace('device-','')},
        dataType : 'json',
        success: function(data) {
            updateDeviceStatuses();
              },
            error: function() {
              console.log('Can not turn device.');
              }

    });
});
```

Joonis 3.20 Koodi näide

KOKKUVÕTE

Lõputöö tulemusena loodi veebirakendus targa kasvuhoone prototüübi jaoks.Selle rakenduse abil saab jälgida kasvuhoone õhu temperatuuri ja niiskust, mulla niiskust ja valguse taset. Samuti saab juhtida täitureid: muuta valgust LED-ribaga, avada aknad servomootoriga ja kasta taimi veepumbaga.) Veebirakenduse andmeid värskendatakse automaatselt ning need saabuvad kiiresti ja õigeaegselt.

Lõputöö valmimise käigus oli uuritud erinevaid veebirakenduse realiseerimise võimalusi. Samuti uuriti, kuidas töötada andmebaasi ja selliste tööriistadega nagu Ajax ja Bootstrap. Selles töös on palju tööd seotud programmeerimiskeele *JavaScript*i õppimisega.

Lõputöö eesmärgi saavutamiseks kasutati *NodeJS, ExpressJs, PostgreSQL, JavaScript* Nende tehnoloogiate valik põhines võrdlusel teiste sarnaste probleemilahendustehnoloogiatega

Selle töö tulemuseks on veebirakendus nutika kasvuhoone andmete jälgimiseks NodeJS platvormil, kasutades Expressi. Rakendus kasutab andmebaasi PostgreSQL, mis salvestab kõik nutikasvuhoone andurinäidud Tehtud tööd võib nimetada edukaks, kuna veebirakendus töötab õigesti. Kõik protsessid võib jälgida ja võib juhtida täiturite

Tulevikus võiks tööd täiendada suurte nutikate kasvuhoonete täiustatud või täiendavate anduritega või vahendada mikrokontrollerit, selle asemel integreerida juhtmeta mikrokontrollerit, mis on tänapäeval eelistatavam Samamoodi saab veebirakendusse lisada kasutajatele uusi funktsioone.

SUMMARY

Creating a web application for a smart greenhouse prototype

The goal of this thesis was to create a web application for a smart greenhouse prototype. Using this application, it is possible to track data such as air temperature, humidity, soil moisture and luminosity directly from the greenhouse. It is also possible to control additional devices: for example, the user can change the illumination level using LED, open and close the window using a servo motor, and water plants with a water pump. The web application data is updated automatically and arrives in a timely manner.

In the course of this work there were studied various possibilities of implementing a web application, as well as various ways of working with databases. Technologies such as ajax and bootstrap were studied to implement the proposal. Moreover, a lot of work has been done to study the programming language *JavaScript.*

NodeJS, ExpressJs, PostgreSQL and *JavaScript* were used to achieve the goal of this thesis. The choice of these technologies was based on their comparison with other technologies for solving similar objectives.

The result of this work is a web application for tracking smart greenhouse data on the NodeJS platform using Express. The PostgreSQL database is used in the application, and all sensor readings from the smart greenhouse are stored in the database.

In the future, this work can be supplemented with improved sensors, or additional sensors for large greenhouses. It is also possible to replace the controller with wireless connection, which is more preferable nowadays, and to add new features to the application for the convenience of users.

KASUTATUD KIRJANDUSE LOETELU

1. Alexander S. Gillis, internet of things (IoT). [Online] https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT (30.01.2021)

2. [Triin Marandi, Fotosüntees, loengumaterjal, Tartu Ülikool, 2012 https://kodu.ut.ee/~triinm/bioloogia/fotosntees.html[Online] (20.03.2021)

3. Merilin Saarma, Valguse kvantiteedi ja kvaliteedi roll taime kasvu ning herbivooria tolerantsuse regulatsioonis, magistritöö, Tartu, 2013. [Online] http://dspace.ut.ee/bitstream/handle/10062/31469/Magistritoo_Merili_Saarma.pdf, lk. 47 (20.03.2021)

4. DHT11 Sensor and Its Working. [Online] https://www.elprocus.com/a-brief-ondht11-sensor/ (16.05.2021)

5. What is a light sensor? Types, Uses, Arduino Guide. [Online] https://www.seeedstudio.com/blog/2020/01/08/what-is-a-light-sensor-typesuses-arduino-guide/ (16.05.2021)

6. List of the Best 21 Arduino Modules. [Online] https://randomnerdtutorials.com/21-arduino-modules-you-can-buy-for-lessthan-2/ (26.02.2021)
7. What is an Arduino? [Online] https://learn.sparkfun.com/tutorials/what-is-anarduino/all (26.02.2021)

8. ADVANTAGES OF POSTGRESQL. [Online] https://www.cybertecpostgresql.com/en/postgresql-overview/advantages-of-postgresql/ (15.02.2021)
9. Google Trend. [Online] https://trends.google.com/trends/explore?geo=EE&q=%2Fm%2F05z_r2n,%2Fm%2F05ynw,%2Fm%2F04y3k (19.04.2021)

10. Ihor Feoktistov 7 Advantages of Node.js for Startups. [Online] https://relevant.software/blog/7-benefits-of-node-js-for-startups/#:~:text=js%20really%20shines%20in%20building,user%20real%2Dtime%20data%20situations (27.02.2021)

11. Express.js. [Online] https://www.tutorialsteacher.com/nodejs/expressjs (15.02.2021)

12. Explanation: What is a backend and why do I need one? [Online] http://mariechatfield.com/tutorials/explanations/backend.html (14.05.2021)

13. What Is a Front-End Developer? [Online] https://frontendmasters.com/books/front-end-handbook/2018/what-is-a-FD.html (14.05.2021)

14. jQuery Ajax. [Online] https://www.tutorialrepublic.com/jquery-tutorial/jqueryajax.php (21.04.2012)

15. How to Use jQuery's ajax() Function for Asynchronous HTTP Requests. [Online] https://www.sitepoint.com/use-jquerys-ajax-function/ (30.01.2021)

16. Advantages and Disadvantages of JavaScript. [Online] https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/ (27.02.2021)