TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aleksandr Gildi 201362

VEEBIPÕHINE EHITUSFÜÜSIKA TÖÖRIISTAKAST EHITUSINSENERIDELE

Bakalaureusetöö

Juhendaja: Kalle Tammemäe
Tehnikateaduste doktor

Tallinn 2024

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: Aleksandr Gildi

12.02.2024

Annotatsioon

[YOUR TEXT GOES HERE]

Lõputöö on kirjutatud [mis keeles] keeles ning sisaldab teksti [lehekülgede arv] leheküljel, [peatükkide arv] peatükki, [jooniste arv] joonist, [tabelite arv] tabelit.

Abstract

Building physics web toolbox for civil engineers

[YOUR TEXT GOES HERE]

The thesis is written in [language] and is [number of pages in main document] pages long, including [number] chapters, [number] figures and [number] tables.

Lühendite ja mõistete sõnastik

TODO:	**SORT ALPABETICALLY**
Demo-versioon	Tarkvara prooviversioon (*Demonstration version*)
MVP	Minimaalne elujõuline toode(*Minimum Viable Product*)
Miro	interaktiivne keskond milleks?(*ToDo*)
PDF	digitaalne formaat(*Portable Document Format*)
2D	TODO(*2 Dimensional*)
SPA	TODO(*Single Page Application*)
JavaScript	TODO(*TODO*)
TypeScript	TODO(*TODO*)
HTML	TODO(*Hyper Text Markup Language*)
CSS	TODO(*Cascade Style Sheet*)
front-end	TODO(*front-end*)
JSX	TODO(*JSX*)
props	TODO(*props*)
MVVM	TODO(*MVVM*)
SFC	TODO(*Single File Component*)
MVC	TODO(*Model View Controller*)
JSON	TODO(*JavaScript Object Notation*)
PHP	TODO(*PHP*)
REST	TODO(*Representational State Transfer*)
Oracle	TODO(*Oracle*)
JWT	TODO(*JavaScript Web Token*)
ORM	TODO(*Object Relation Mapper*)
root	TODO(*root*)
HTTP	TODO(*Hypertext Transfer Protocol*)
stateless	TODO(*stateless*)
Bootstrap	TODO(*stateless*)
UX/UI	TODO(*User Expirience/User Interface*)
popup	TODO(*User Expirience/User Interface*)
EFCore	TODO(*Entity Framework Core*)
reverse proxy	TODO(*reverse proxy*)
GUID	TODO(*reverse proxy*)

Sisukord

Joonistе loetelu

Tabelite loetelu

1. Sissejuhatus

Ehitusfüüsika on ehitusvaldkonna haru, mis käsitleb hoonet füüsikaliste nähtuste seisukohalt: soojus, niiskus, õhk, heli ja valgus, seetõttu võib väita, et ehitufüüsikaga puutub oma elus kokku igaüks. Ehitusfüüka valdkonna projekteerimise peamised eesmärgid on:

- optimeerida hoone kütte ning jahutuskulud
- tagada hoones soojuslikku mugavust, niiskustingimusi ja sisekliima kvaliteeti tervikuna
- välistada mikrobioloogilist kasvu konstruktsioonides
- välistada veest ja niiskusest tekkivaid probleeme
- tagada hoonepiirete õhupidavust
- parandada akustilist kvaliteeti

Ehitusfüüsikavaldkond on oluline, sest see suures osas määratleb hoonete sisekliima kvaliteeti, teiste sõnadega tagab inimestele kvaliteetset elukeskkonda. Valesti projekteeritud hooned võivad muuhulgas avaldada negatiivset mõju inimeste tervisele või olla isegi ohtlikud. Seevastu õigesti projekteeritud hoone tagab kasutajale mugavusetunnet ja ka hoiab raha kokku minimeerides hoone kasutuskulusid.

Ressursside kallinemise olukorras sai ehitusfüüsikast eriti tähtis inseneriteaduse haru, sest muuhulgas see käsitleb hoone soojusliku toimivuse probleemi. See tähendab, et õigesti projekteeritud hoone talvel tarbib vähem energiat küttele ning suvel vastupidi – jahutusele.

Ehitsfüüsikaga peab arvestama hoone elutsükli igal etapil - kavandamine, projekteerimine, ehitamine ja haldamine. Hoone kavandamisel määratakse planeeritavaid energiakulusid ja energiaklassi. Hoone projekteerimise faasis peavad ehitusfüüsikaga arvestama arhitektid, konstruktorid ja ka tehnosüsteemide projekteerijad, kes valivad õigete omadustega materjalid ning hindavad nende materjalide koosmõju konstruktsiooni toimimisele. Ehituse faasis peab ehitusfüüsikaga arvestama ehitusjuhid - kuigi ehitatakse tavaliselt projekti järgi, paraku peab ehituses ka operatiivselt võtta keerulisi otsuseid jooksvatest muudatustest keset ehitusprotsessi. Ja viimaseks peavad ehitusfüüsikat meeles hoidma ka hoone haldamisega tegelevad inimesed.

Probleemi teine külg on ehitusvaldkonna madal digitaliseerumise tase (ja konservatiivsus üldiselt). Viimastel aastatel on arendatud palju profesionaalseid tarkvarasid projekteerimise

ja ehitusjuhtimise tarbeks, kuid ehitusfüüsika valdkonna tarkvara arendused on olnud väga tagasihoidlikud. Turul on olemas mõned üksikud tooted, kuid need on liiga keerulised ja võrdlemisi ebamugava kasutajaliidesega - sellise tarkvara sihtgrupp on teadusvaldkond. Ehitusinseneride töö hõlmab väga palju erinevaid asju ning on tavaliselt ajaliselt väga piiratud, mistõttu keerulise kasutajaliidesega ja tööpõhimõttega tarkvara kasutamine ei ole parim variant.

Käesoleva töö eesmärk on välja töötada toodet, mis võimaldaks lahendada ehitusfüüsika valdkonna ülesandeid mugavalt ja operatiivselt. See võiks parandada olukorda, kus probleemide lahendamine jääb üldse erinevatel etapidel tegemata tarkvara või tarkvara kasutamise oskuste tõttu. See võiks olla ehitusinseneridele abivahendiks, mis ei vaja väga sügavat valdkonna tundmist, et teostada piisavas mahus arvutusi tagamaks ehitusprojekti või ehituse kvaliteeti ehitusfüüsika seisukohalt. Ehitusfüüsika valdkond on lai ning lahendusi on tarvis leida väga paljudele probleemidele. Käesoleva töö raames keskendutakse esialgu vaid ühe konkreetse probleemi lahendamisele, mis on ühtlasi ka kõige levinuim probleem - veeauru kondenseerumise riski hindamine ehituskonstruktsioonides.

2. Probleemi olemus

2.1 Probleemi uurimine

Ehitusfüüsika mõistes ehituskonstruktsioon kujutab endast erinevate füüsikaliste omadustega kihtidest koosnevat struktuuri, mis eraldab kaks erinevat keskkonda (näiteks: hoone sees olev õhk ja õues olev õhk). Seejuures kõige olulisemad materjalide omadused on soojuserijuhtivus λ [W/mK] ja veeaurutakistus, mis võib olla väljendatud mitmel viisil (neid viise on palju, aga käesolevas töös keskendutakse ainult järgmistele, kuna need on kõige rohkem kasutatud nii raamatutes, kui ka materjalitootjate dokumentatsioonis): μ - diffusioonitakistustegur (materjali omadus), Sd[m] - suhteline diffusioonitakistus (kindla paksusega toote omadus).

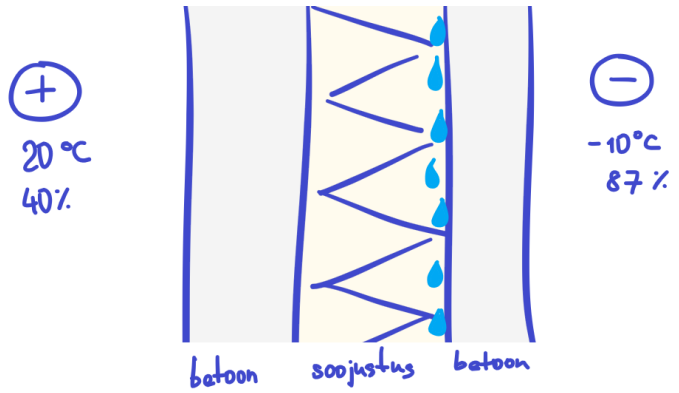

Figure 1. *Kihilise konstruktsiooni näide*

Teatud tingimustel võib tekkida olukord, kui konstruktsiooni sees on mingis punktis suhteline niiskus nii kõrge, et soodustab bioloogiliste kahjustuste või isegi kondensaadi tekkimist. Nimetatud olukord on ohtlik nii ehituskonstruktsioonile, mis pikaajalise niiskuse mõjul lagunevad, kui ka inimese tervisele, sest konstruktsioonide sees olev hallitus on õhku sattuvate bakterite allikaks. Seda, kuidas konstruktsioon töötab soojus- ja niiskuse leviku seisukohalt nimetatakse konstruktsiooni niiskustehniliseks toimivuseks. Arvutust, mille eesmärgiks on hinnata kondenseerumise riski konstruktsioonis, nimetatakse konstruktsiooni niiskustehnilise toimivuse analüüsiks.

Tegemist on klassikalise ehitusfüüsika ülesandega, mille lehandemiseks peab ette võtma järgmiseid samme:

- konstruktsiooni kihtide soojustakistuse ja konstruktsiooni summaarse soojustakistuse

arvtus

- temperatuuri jaotuse määramine kihtides sõltuvalt sise- ja väliskeskkonna temperatuuridest ning soojustakistuste väärtustest
- konstruktsiooni kihtide veeaurutakistuse ja konstruktsiooni summaarse veeaurutakistuse arvutus
- veeauru küllastusrõhu jaotuse määramine lähtuvalt temperatuuri jaotusest
- veeauru osarõhu jaotuse määramine kihides sõltuvalt sise- ja väliskeskkonna parameetritest ning veeaurutakistuse väärtustest
- tulemuste esitamine graafiliselt diagrammil
- arvutuste kordamine erinevate sise- ja väliskeskkonna parameetrite kombinatsioonidega

Ülesande käsitsi lahendades, koostatakse tabelit, mille ridadesse pannakse kirja kihid ja veergudesse arvutatakse väärtused. Kuigi arvutused ise ei ole väga keerulised (tegemist on tavaliste füüsika valemitega), paraku käsitsi arvutamine võtab tohutult palju aega. Microsoft Excel võimaldab teatud määral protessi automatiseerida, kuid siiski mõned tegevused (näiteks uute kihtide lisamine, või kihtide järjekorra muutmine) jäävad suures osas käsitööks, mis võtab palju aega ja ka soodustab vea tegemist. Pildil 2 on toodud sellise tabeli näide.

Arvutustabel

	Kihi paksus	Soojus-erijuhtivus	Kihi soojus-takistus	Temp. muutmine	Temp. kihi piiril	Küllastus-rõhk	Veeauru-erijuhtivus	Veeauru-takistus	Rõhkude erinevus	Veeauru osarõgk
	d [mm]	λ_d [W/(mK)]	R [m^{2}K/W]	Δ t [^{0}C]	t [^{0}C]	p_{sat} [Pa]	δ_p[kg/msPa]	Z_p[m^{2}sPa/kg]	Δ P [Pa]	P [Pa]
Siseõhk										
Sisepind			0.13	0.5	23.0	2808				842.3
Krohv	5	0.570	0.01	0.0	22.5	2724	2.0E-11	2.5E+11	2.5E-01	842.3
krohv	5	0.57	0.01	0.0	22.5	2718	2.0E-11	2.5E+11	2.5E-01	842.1
Bauroc	150	0.11	1.36	5.3	22.4	2713	2.6E-11	5.7E+12	5.7E+00	841.8
Kile	1	0.17	0.01	0.0	17.2	1959	2.0E-15	5.1E+14	5.1E+02	836.2
Soojustus	200	0.035	5.71	22.0	17.2	1956	2.0E-10	1.0E+12	1.0E+00	326.1
Tsementkiudplaat	10	0.049	0.20	0.8	-4.8	407	3.7E-12	2.7E+12	2.7E+00	325.1
Krohv	5	1	0.01	0.0	-5.6	380	1.8E-11	2.7E+11	2.7E-01	322.4
Välispind			0.04	0.2	-5.6	380				322.1
Välisõhk					-5.8	375				322.1
Kokku			**7.5**					**5.22776E+14**		

Figure 2. *Arvutustabeli näide*

Analüüsi tulemused esitatakse graafiliselt diagrammi kujul, mille x telg on punkti asukoht konstruktsioonis ning y teljel on temperatuuri, veeauru küllastus- ja osarõhu väärtused vastavas punktis – näide on todud pildil 3.

Graafik annab väga head visuaalset ülevaadet konstruktsiooni kihtides toimuvale. Täpsemalt öeldes peab vaatama veeauru küllastus- ja osarõhkude jaotuste graafikuid. Veeauru osa- ja küllastusrõhu suhe on suhteline niiskus. Mida lähedam osarõhu graafiku joon küllastusrõhu graafiku joonele, seda kõrgem on suhteline niiskus. Punkt, milles need jooned ristuvad on suhteline niiskus 100%, mis tähendab kondensaadi tekkimist – sellist punkti

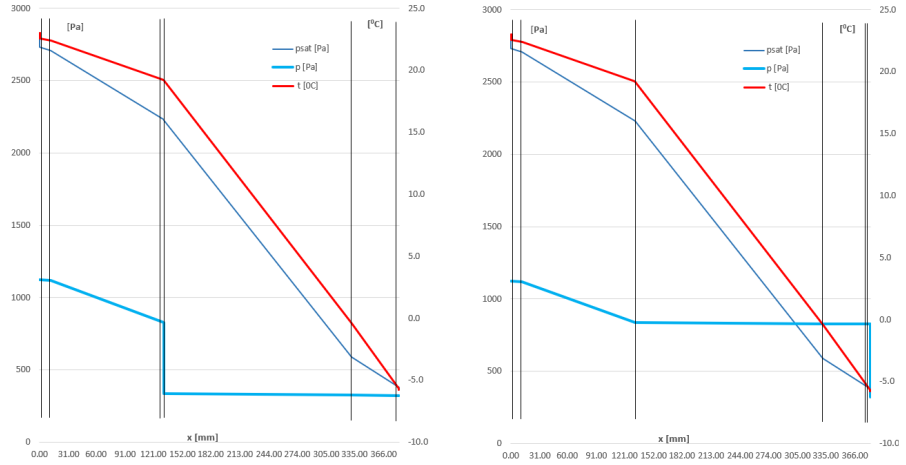

Figure 3. *Näide tulemuste esitamisest graafikul*

nimetatakse kastepunktiks. Näide on toodud pildil 3: vasakul on niiskustehniline olukord hea, kuna aurutõke asub õiges kohas, ning paremal paiknem konstruktsiooni külmemal pool veeaurupidav kiht, mistõttu läheb osarõhk kõrgeks ja ületab küllastusrõhu väärtust.

Kuigi probleem on üldjoontes lahendatav näiteks *Microsoft Excel* vahenditega, paraku pole see kõige mugavam viis mitmel põhjusel. Selline lähenemine vajab palju käsitööd kihtide lisamiseks või ümber paigutamiseks, mis on analüüsi lahutamata osa – proovitakse erinevaid materjale erinevates konstruktsiooni kohtades. Samuti materjalide andmeid on siiski vaja otsida ning hallata nende aktuaalsust ja usaldusväärsust – aeganõudev töö, mida võiks elimineerida kasutades valmistoodet.

2.2 Olemasolevad lahendused ja turu analüüs

Üks populaarsematest analoogsetest lahendustest, mis on inseneridel kasutusel Euroopas, sealhulgas ka Eestis, on Saksa päritoluga tarkvara **Ubakus**. Tegemist on kommertstarkvaraga, mis töötab veebirkenduse kujul. Tarvara *demo*-versioon on saadaval tasuta.

Ubakus võimaldab teostada konstruktsiooni niiskustehnilist analüüsi. Kasutajaliides võimaldab mudeldada mitmest kihist koosneva konstruktsiooni, valides igale kihile paksust ja materjali, millest kiht koosneb. Tugev eelis on see, et tarkvaraga saab analüüsida ka mittehomogeensete (mitemest erinevast materjalist, nt puitsõrestiksein) kihtidega konstruktsioone – pilt 4

Ehitusmaterjalide valik, mida on võimalik konstruktsiooni mudeldamisel kasutada, on piisavalt lai (aga tasuta versioonis piiratud). Tasulises versioonis on samuti võimalik ka oma materjalide lisamine ja kasutamine. Osa materjalidest on abstraktsed (näiteks: betoon, puit,

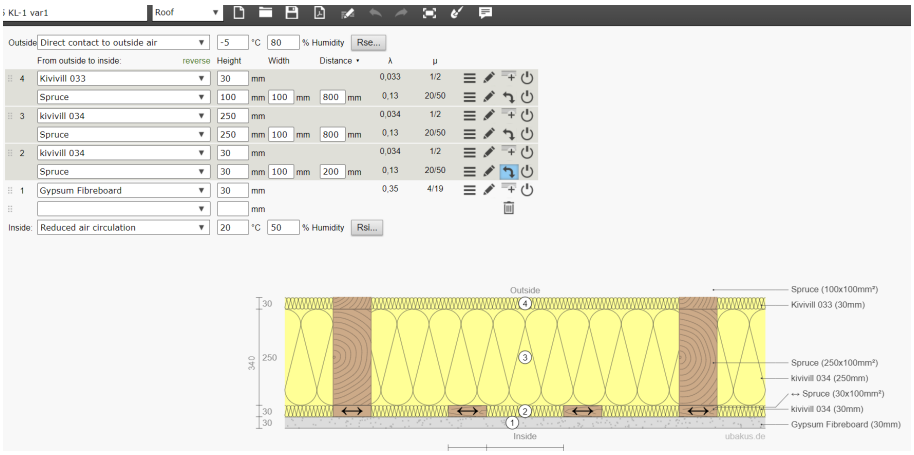

Figure 4. *Ubakus: kasutajaliides, ekraanitõmmis.*

mineraalvill), osa on reaalsed turustatavad tooted (näiteks: Isover soojusisolatsioonide valik) – pilt 5. Võib puuduseks pidada seda, et osa materjale (konkreetsed toted) on Saksamaal ja Kesk-Euroopas turustatavad materjalid, mistõttu selle tarkvara kasutades Eestis peab kas sisestama vajalikud kohalikud materjalid käsitsi, või arvestada Saksa analoogide kasutusest tuleneva arvutuste ebatäpsusega.

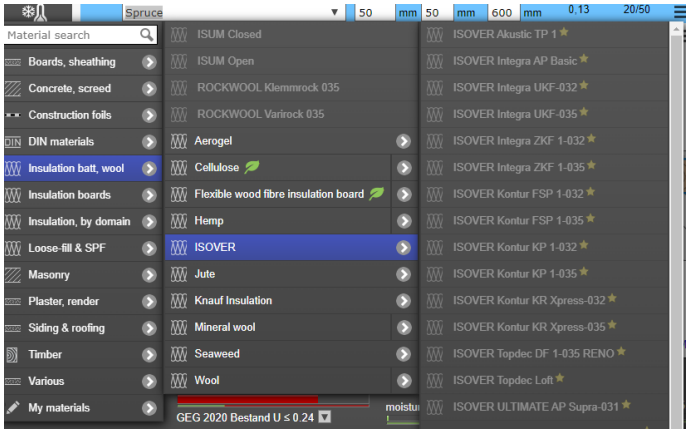

Figure 5. *Ubakus: ehitusmaterjalide valik baasis, ekraanitõmmis.*

Keskkonnatingimused valitakse manuaalselt sisestades õhutemperatuuri ja -niiskuse väärtused. Rakendus võimaldab arvutuste tulemust vaadata erineval viisil, alates lihtsamast 3D visualiseeringust kuni värvilise temperatuurikaardini. Tarkavara saab osa aastase tellimusega, mille maksumus on alates 50 kuni 120 eurot sõltuvalt valitud paketist. Objektiivselt vaadates on tarkvara hea nii funktsionaalsuse kui ka hinna seisukohalt. Lisaks sellele on ka kasutajaliides piisavalt mugav ja intuitiivselt arusaadav, et seda saaks kasutada ka inimene, kellel puuduvad sügavad teadmised valdkonnast. Nagu varem oli mainitud, takvara on suunatud eelkõige Kesk-Euroopa ja Canada turgudele, Balti ja Skandinaavia riigidele lokaliseerimine puudub. Kokkuvõttes antud lahendust võib kindlasti võtta arvesse toote funktsionaalsuse kavandamisel.

Physibel Glasta on üks analoogne lahendus veel, mis on samuti kommertstarkvara. Tegemist on samuti arvutile paigaldatava tarkvaraga, mille kasutajaliides on veidi keerulisem ja ka disain on oluliselt konservatiivsem (pilt 6).

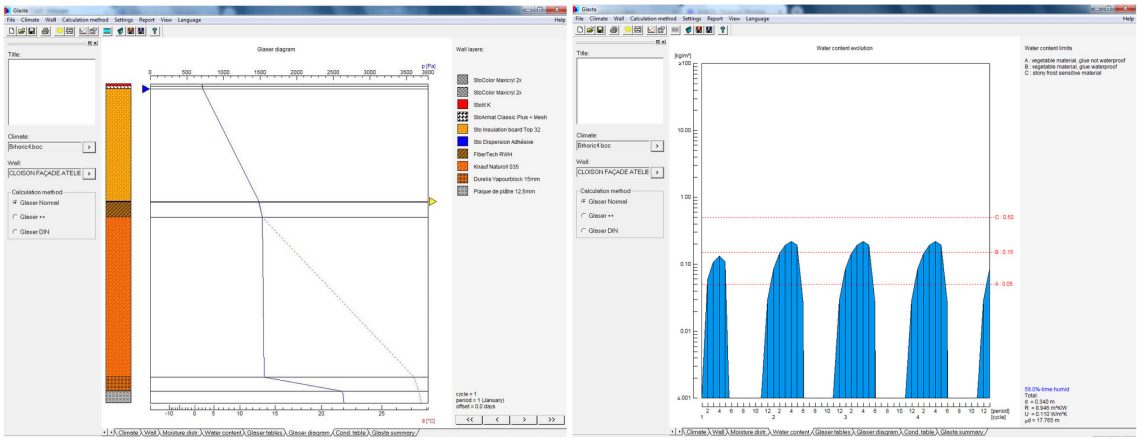

Figure 6. *Glasta: kasutajaliides, ekraanitõmmis.*

Selle tarkvara funktsionaalsuse tugev eelis on võimalus teostada analüüsi aasta lõikes - väga tihti kondenseerumise probleem esineb vaid teatud perioodil (tavaliselt külmal hooajal), ülejäänud ajal toimub kuivamine. See, et ühel või kahel talvisel kuul esineb konstruktsioonis kondenseerumise oht ei pruugi olla probleemiks, kui ülejäänud ajal jõuab konstruktsioon täielikult kuivada. Antud asjaolu Glasta tarkvara analüüsib ning tulemust esitatakse ka graafikul (pilt 6). Antud funktsionaalsus on äärmiselt oluline ja selle vajadusega peab toote planeerimisel arvestama. Tarkvara hind on suurusjärgus 500 eurot aastas, mis on päris kõrge, ning lisaks ka alla laadimise ja paigaldamise vajadus teeb antud lahendust ebamugavaks ja paljudel juhtudel ebaotstarbekaks.

Valdkonnas on olemas ka oma lipulaev – **Delphin** on professionaalne tarkvara, mille hind on suurusjärgus 1000-1500 eurot aastas. Eelisteks on väga lai funktsionaalsus ning ka täielik vabadus konstruktsiooni ja keskkonna mudeldamisel. Viimane on ühtlasi ka puuduseks, sest kliimatingimuste mudeldamine eeldab ilmatingimuste (sealhulgas ka andmed päikesekiirgusest, sademetest) andmebaasi olemasolu. Samuti vajab tarkvara ka kasutaja koolitust, mida tootja pakub ka pakub hinnaga 800 eurot. Eeltoodud asjaolud teevad antud tarkvara sobilikuks ja otstarbekaks vaid nendele, kellel ehitusfüüsika arvutused on põhitegevuseks.

3. Arenduse metoodika

Probleemi lahendamist alustatakse olemasolevate lahenduste otsimisest ja analüüsimisest. Iga lahenduse puhul tuuakse välja tugevad küljed ja puudused, arvestades planeeritava toote kontseptsioonist ja sihtgrupist. Samuti tehakse erinevate lahenduste hinnavõrdlust. Lähtuvalt lahenduste analüüsi tulemustest defineeritakse konkreetsed nõuded kavandatavale infosüsteemile, millest lähtutakse infosüsteemi tehniliste lahenduste projekteerimisel. Antud kohas määratakse ka toote MVP, mis oleks lõputöö mahu kohane.

Kui nõuded infosüsteemile on paika pandud, valitakse infosüsteemi ehitamise tehnoloogiaid – kasutajaliides, serveriosa ja andmebaas. Tehnoloogiate all mõeldakse konkreetsed programmeerimiskeeled ja raamistikud. Samuti lähtuvalt infosüsteemi nõuetest projekteeritakse kasutajaliidese disainilahendust.

Seejärel kavandatakse nii üldist infosüsteemi arhitektuuri (kuidas infosüsteemi osad omavahel töötavad, millised andmed kasutajaliidese ja serveriosa vahel liiguvad), kui ka arhitektuursed lahendust iga infosüsteemi osale eraldi (näiteks: serveriosa ja kasutajaliidese struktuur).

Seejärel planeeritakse arenduse protess. Kavandatav funktsionaalsus jagatakse kasutajalugudeks, mida gruppeeritakse featuurideks ja *epic*-uteks. Kasutajalugudest moodustatakse tehnilised ülesanded, mida võetakse aluseks koodu kirjutamisel.

TODO: kuidas selliseid asju õigesti kirjutada?

Kui MVP funktsionaalsus on saavutatud, siis toodet antakse mitmele sihtgrupi esindajatele testimiseks ja tagasiside saamiseks. Tagasiside alusel kavandatakse edasist arendusprotsessi.

4. Kavandatava veebirakenduse analüüs

4.1 Nõuete defineerimine

4.1.1 Funktsionaalsed nõuded

Funktsionaalsete nõuete määramisel lähtutakse erinevate osapoolte vajadusest - süsteemi kasutaja ja süsteemi administraator. **Süsteemi kasutajana tahan:**

- registreerida endale konto ja logida sisse
- logida sisse
- hallata oma konto andmeid
- tellida tasulist paketi
- vaadata enda poolt salvestatud materjalide nimekirja
- luua ja salvestada uus materjal
- redigeerida varem salvestatud materjal
- kustutada varem salvestatud materjal
- lisada uus kiht konstruktsiooni mudelisse
- valida uue kihi materjal
- sisestada uue kihi paksuse väärtust
- redigeerida olemasolevat kihti
- kustutada olemasolevat kihti
- vahetada kihtide järjekorda
- valida välistingimuste parameetrid
- valida sisetingimuste parameetrid
- valida konstruktsiooni tüüp
- vaadata tulemusi tabeli kujul (valikuliselt)
- vaadata tulemusi graafikul (valikuliselt)
- vaadata konstruktsiooni toimivuse mõõdikuid
- peale igat muutust kohe näha uusi tulemusi (arvulised väärtused)
- peale igat muutust kohe näha graafikute uuendamist
- näha konstruktsiooni skemaatilist joonist
- salvestada mudeldatud konstruktsiooni
- vaadata salvestatud konstruktsioonid
- kustutada salvestatud konstruktsioonid
- avada salvestatud konstruktsioonid kalkulaatoris

- muuta kiht mittehomogeenseks
- mittehomogeensele kihile lisada alamkihid
- valida alamkihtide materjalid
- sisestada alamkihtide paksuse väärtust
- näha konstruktsiooni skemaatilist joonist
- näha skemaatilise joonise peal graafikut
- näha skemaatilise joonise peal värvilist temperatuurikaarti
- näha dünaamilist analüüsi aasta lõikes
- valida kliimaandmeid dünaamilise analüüsi jaoks
- genereerida analüüsi aruanne PDF formaadis

Infosüsteemi administraatorina soovin:

- vaadata kasutajate nimekirja
- hallata kasutajaid
- seadistada tellimust vormistanud kasutajale vastavad õigused
- hallata kasutajate andmeid
- vaadata süsteemis salvestatud vaikimisi materjalide nimekirja
- salvestada uus materjal
- määrata materjali ligipääsu taset
- redigeerida varem salvestatud materjal
- kustutada varem salvestatud materjal
- hallata uusi materjali kategooriaid
- hallata uusi materjalide tootjaid
- hallata keskkonna seadistuse valikuid
- lisada kliimaandmeid failina

Funktsionaalsest nõuetest on kokku pandud kasutajalood, mis omakorda jagatud featurideks. Protsessi visualiseerimiseks on kasutatud Miro interaktiivne keskkond. Kliendi kasutajalood on jagatud kuueks featuuriks (pilt 7):

- infosüsteemi kasutamine
- ehitusmaterjalide andmebaasi haldamine
- konstruktsiooni mudeldamine
- arvutuse lisatingimuse seadistamine
- analüüsi tulemuste esitamine
- salvestatud konstruktsioonide haldamine

Administraatori kasutajalood on jagatud kolmeks featuuriks (pilt 8):

- infossteemi kasutajate haldamine
- ehitusmaterjalide avaliku andmebaasi haldamine
- lisaandmete baasi haldamine

Samuti olid kasutajalood kategoriseeritud prioriteedi järgi. Kõrgema prioriteediga kasutajalood moodustavad MVP funktsionaalsust, mida arendatakse käesoleva lõputöö käigus. Osa funktsionaalsusest, mis on kirjeldatud madala prioriteedi kasutajalugudega, jääb käesoleva lõputöö skoobist välja. Suures osas see puudutab tulemuste esitamise viise: mudeldatud konstruktsiooni skemaatilise 2D joonise genereerimine ning selle peale graafikute või värvikaartide pealekandmine eeldab eraldi teeki kirjutamist. Isegi kui värvikaartide puhul õnnestuks leida valmislahendust, siis selle adapteerimine siiski tähendab suurt töömahtu. Samuti on MVP skoobist välja jäetud mittehonogeensete kihtidega konstruktsioonide arvutus. Andmemudelid kohe tuleb projekteerida nii, et tulevikus see oleks võimalik implementeerida ilma suurte muutusteta , kuid arvutuste ja kasutajaliidese lihtsustamise mõttes jääb see osa esimese iteratsiooni skoobist välja.

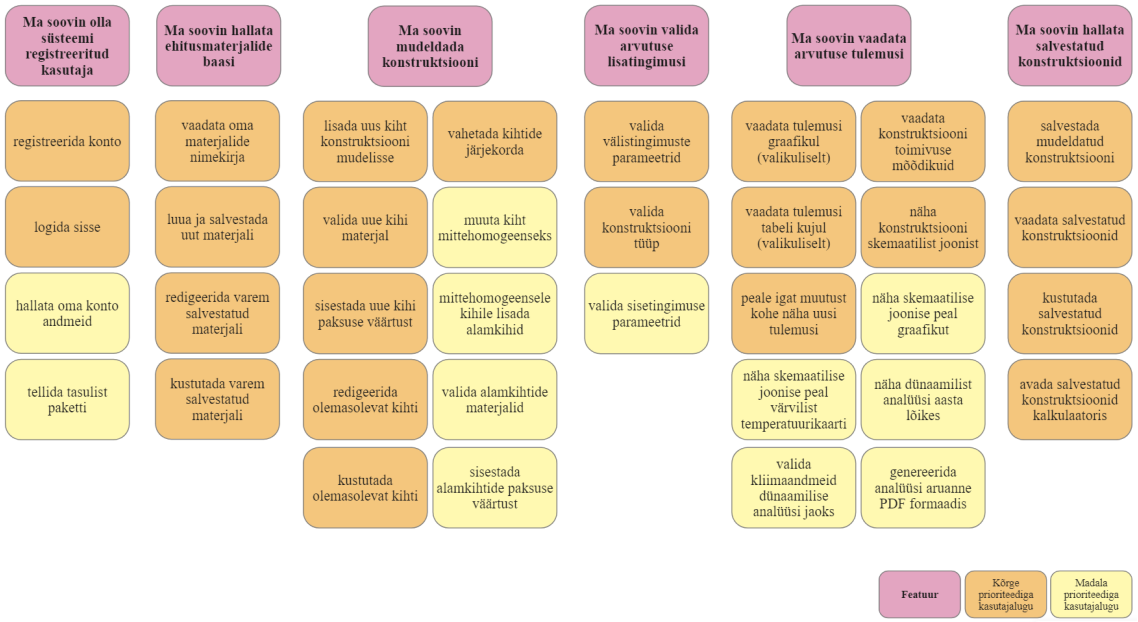

Figure 7. *Kliendi kasutajalood*

4.1.2 Mittefunktsionaalsed nõuded

Lisaks osas 4.1.1 kirjeldatud funktsionaalsetele nõuetele, peab süsteem vastama järgmistele nõuetele:

- kasutajaliides peab olema kiire – kasutaja peab nägema arvutuse tulemuste uuendamist kohe peale lähteandmete (kihid, parameetrid) muutmist, pikk ooteaeg või lehe ümberlaadimine tulemuste näitamiseks ei ole aktsepteeritav;

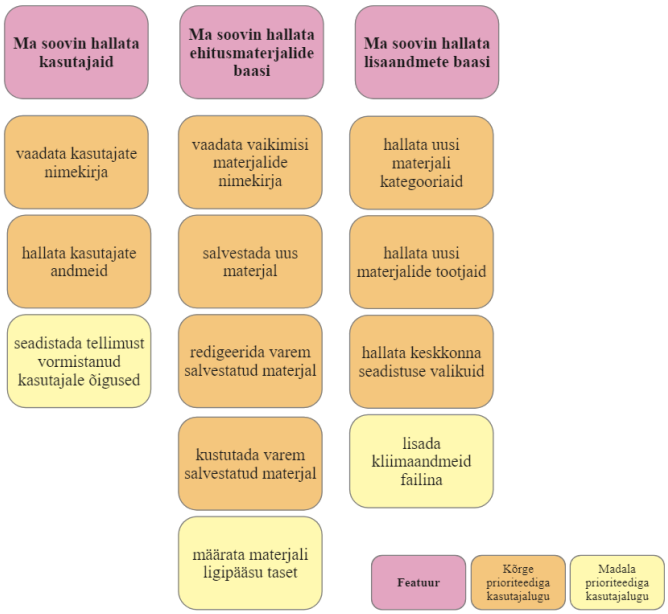

Figure 8. *administraatori kasutajalood*

- kasutajaliides peab olema kasutajasõbralik – kasutaja peab intuitiivselt aru saama, mida ta peaks tegema, et jõuda tulemuseni, vajadusel peab ta olema juhitud *popup*-tüüpi infoplokidega;
- kasutajaliides peab olema kaasaaegne, st välimus ja veebilehe struktuur peavad vastama kaasaaegsetele UX/UI printsiibidele.

4.2 Tehnoloogiate valik

Tehnoloogiate valimisel lähtutakse kaasaaegsetest veebirakendamise ehitamise pritsiibidest, arvestatakse rakenduse loogika keerukust, võimalike kasutajate hulka, säilitavate andmete mahtusid ja infosüsteemi edasise arengu perspektiive.

Veebirakendusel peab olema selgelt eristatav serveriosa ja kasutajaliides. Vajadusel saab tulevikus implementeerida ka teised kasutajaliidesed, mis töötavad sama serveriosaga (näiteks: mobiilrakendus). See tähendab, et infosüsteemi arhitektuur peab olema REST arhitektuurse stiili nõuete kohane. See eeldab ka seda, et andmevahetus kasutajaliidese ja serveriosa vahel peab toimuma HTTP päringutega kasutades JSON (JavaScript Object Notation) formaati. JSON on JavaScript-i põhine andmevahetuse formaat, mis representeerib JavaScript-i andmeobjektid tekstilisel kujul, mida on võimalik saata HTTP päringutega üle veebi[1]. Lisaks sellele on oluline, et kõik päringud on omavahel sõltumatud, see tähendab, et ühe päringu raames serveriosas alustatakse ja lõpetatakse kõik päringuga seotud protsessid, päringute vahel serveril puudub kliendiga seotud olek (*stateless* protokoll)[1].

Selleks, et süsteem saaks võimalikult pikemat aega töötama ilma tehnoloogiate uuenduse vajaduseta, peab kasutusele võtma võimalikult uued, aga samas ka stabiilsed ja pikema toega lahenduste versioonid.

4.2.1 Kasutajaliides

Kasutajaliides implementeeritakse üheleherakendusena (SPA). SPA tehnoloogia võimaldab minimeerida andmevahetuse mahtusid: serverilt küsitakse ja vastavalt kliendile saadetakse ainult need andmed, mida on hetkel tarvis. Arendatava infosüsteemi kontekstis see on oluline, sest kõiki arvutusi teostatakse serveril ning kliendile saadetakse andmed, mis on vajalikud tulemuste näitamiseks kasutajale. Iga uus tegevus kasutajaliideses, mis mõjutab tulemusi (uue kihi lisamine, kihtide järjekorra muutmine, arvutuse parameetrite muutmine jms.), tähendab uut päringut serverile. Samuti SPA tehnoloogia võimaldab muuta lehe sisu dünaamilisel viisil – uuendatakse vaid lehe teatud komponent ilma kogu lehekülje ümberlaadimise vajaduseta. See on ka oluline, kuna tulemuste esitamist peab uuendama dünaamiliselt kohe peale arvutuse lähteandmete muutmist.

Üheleherakenduste implementeerimiseks kasutatakse JavaScript programmeerimiskeelt veebilehe dünaamilise loogika juhtimiseks. Soovitavalt kasutada JavaScript keele laiendust – TypeScript, mis muudab JavaScript-i tugevalt ja staatiliselt tüübitud keeleks. Lehtede struktuuri ehitamiseks kasutatakse HTML (või raamistikust sõltuv laiendatud HTML-i süntaks). Kujundust teostatakse CSS stiilireeglitega ja lihtsustamise mõttes võetakse kasutusele ka vastavad teegid nt Bootstrap.

Kuigi on võimalik implementeerida loogikat kasutades puhtat JavaScript koodi, tänapäeval seda tehakse harva. On olemas erinevad lahendused, mis oluliselt lihtsustavad rakenduse ehitamise protsessi, kuid nõuavad ka spetsiifilisi teadmisi. Raamistiku kasutuselevõtt olulisel määral vähendab koodi kirjutamist, kuna raamistik ise haldab loogikat, mis on seotud näiteks *Routing*-uga, turvalisusega, komponentide genereerimise ja uuendamisega. Spetsiifiliste asjade jaoks kasutatakse eraldi pluginaid ja teeke. Näiteks päringute saatmise ja serveri vastuse töötluseks kasutatakse *Axios* – teek, mida saab kasutada erinevate raamistikutega.

Üheleherakenduse implementeerimiseks kõige sobilikud JavaScript raamistikud on *React*, *Vue.js* ja *Angular*. Kõikidel raamistikutel on oma eripärad alates projekti arhitektuurist kuni koodi süntaksini.

React on laialt levinud *front-end* teek, mis kasutab JavaScript programmeerimiskeelt. Lehe šablooni kujundamiseks kasutatakse JSX (JavaScript XML). JSX on JavaScript-i laiendus,

mis võimaldab sisestada HTML koodi JavaScript-i programmi. Rakendus koosneb React-elementidest, mille oleku haldamisega teek tegeleb ise. Elemendid on taaskasutatavad ning nendele antakse andmeid edasi andmeobjektide kujul (*props*). Kuna lahendus on populaarne – selle kasutamise kohta on kogutud palju teavet ja kogemust veebis, mistõttu probleemide tuvastamine ja lahenduste leidmine on piisavalt lihtne. Lisaks sellele eksisteerib palju pluginaid ja teeke, mida saab React raamistikuga ühendada funktsionaalsuse laiendamiseks.

Vue.js on MVVM (Model-View-ViewModel) tüüpi raamistik. Lehe šablooni kujundamiseks kasutatakse HTML, mis siseldab Vue-spetsiifilist süntaksi, mille abil juhib raamistik lehe logikat. Vue rakendus koosneb SFC komponentidest, igas komponndis on eraldi defineritud lehe šabloon, skript ja stiil. Vue.js raamistik on samuti laialt levinud ja selle kohta on võimalik Internetist piisavalt infot leida.

Angular on MVC (Model-View-Controller) tüüpi raamistik. Angular-i projekt struktuurselt koosneb moodulitest, komponentidest ja teenustest. Angular-is kasutatakse lehe šabloonides sarnaselt Vue raamistikule HTML koodi Angular-spetsiifilise süntaksiga.

Kõik ülaltoodud raamistikud toetavad ka *TypeScript*-i kasutamist. Oma funktsionaalsuse seisukohalt kõik toodud raamistikud võimaldavad implementeerida kavandatavat funktsionaalsust (*routing*, oleku juhtimine, komponentide dünaamiline uuendamine), seega määravaks asjaoluks on arendamisega tegeleva programmeerija eelistused. Kuigi töötamise kiirus on raamistikutel erinev, planeeritava rakenduse suurusjärgu kontekstis see faktor ei ole kriitiline.

Toodud põhjendustel valitakse kasutajaliidese tehnoloogiaks React-i. Programmeerimiskeeleks peab valima TypeScript, mis on erinevalt JavaScript-ist võimaldab teha tüübikirjeldust, tänu millele on programmi käitumine ettearvatavam, vigade tõenäosus väiksem ja kood on üldiselt kvaliteetsem. React on populaarne lahendus, seetõttu eksisteerib palju teeke, pluginaid ja leindusi, mida tõneäoliselt saab kasutusele võtta. Kasutada peab React viimane versioon, mis on käesoleva töö koostamise hetkel v18.2.

4.2.2 Serveriosa

Serveril töötav *backend* rakendus tegeleb kasutajaliidese päringute töötlusega ja andmete saatmisega. Samuti *backend* osa suhtleb andmebaasiga, küsides ja redigeerides andmeid. Rakenduse serveriosa on võimalik implementeerida kasutades järgmiseid programmeerimiskeeli:

- *PHP* – popupaarne ja ka võrdlemisi lihtne programmeerimiskeel (avaldatud 1995. aastal), mille otstarve oli kohe alguselt suunatud veebilehtede ehitamiseks. Kuigi esialgu PHP kontseptsioon oli selline, et HTML-kood genereeriti serveril ja saadeti veebilehitsejale iga kord uuesti näitamiseks (monoliitne arhitektuur), siis viimasel ajal kastutakse PHP ka REST-tüüpi veebirakendustes, kus serveril töötav PHP programm saadab andmeid kasutajaliidese rakendusele JSON (või muul) kujul. Tugevaks eeliseks on see, et suur osa veebimajutust pakkuvaid teenuseid toetavad täna PHP keelt vaikimisi, mistõttu rakenduse paigaldamise protsess sellisel juhul on oluliselt lihtsam (koondub programmi failide kopeerimisele serverile).
- *Java* – objektorienteeritud programmeerimiskeel (avaldatud 1995. aastal), mille arendamisega tegeleb Oracle. Keel sobib suuremate REST-tüüpi veebirakenduste ehitamiseks, kuid selle kasutusvaldkond on palju laiem kui ainult veebirakendused. Java on tugevalt ja staatiliselt tüübitud keel, mis on suureks eeliseks, kuna alandab vigade tekkimise tõenäosust, lisaks on see piisavalt kiire. Samas eeldab see spetsifilisi teadmisi programmeerialt ja ka rakenduse paigaldamine serverile on erinevalt PHP-st ka keerulisem, kuna projekti ehitamine eeldab palju lisategevusi. Java on kasutusel väga suure kasutajate hulgaga infoüsteemides (sh. ka pangasüsteemid).
- *C#* – objektorienteeritud programmeerimiskeel (avaldatud 2000. aastal), mille arendamisega tegeleb Microsoft. Keele süntaks ja programmi struktuuri põhimõtted on väga sarnased Java-le. Keel on tugevalt tüübitud ja ka programmi struktuur on Java-keelega analoogne. C# samuti sobib suurte infosüsteemide ehitamiseks.
- *Python* – üldotstarbeline programmeerimiskeel, mille kasutusvaldkond on lai – programmeerimise õpetamist koolilastele kuni suurte infosüsteemide ehitamiseni – tänu kõigepealt sellele, et keele süntaks on võrdlemisi lihtne ning vastavalt keel on kergemini õpitav. Keel on dünaamiliselt tüübitud, mida erinevates s ituatsioonides saab pidada nii eeliseks kui ka puuduseks.

Rakenduse ehitamiseks on otstarbekas kasutada analoogselt kasutajaliidesega raamistikku. Kõikidel ülaltoodud programmeerimiskeeltel eksisteerivad raamistiku lahendused, mis sobivad veebirakenduse serveriosa ehitamiseks.

- **Laravel** - PHP keeles kirjutatud raamistik, väga populaarne ja laialt levinud, funktsionaalsus katab kõiki veebirakenduse ehitamise vajadusi.
- **Spring** - Java keeles kirjutatud raamistik veebirakenduse ehitamiseks. Sisaldab palju erinevaid mooduleid (nt Spring Security - turvalisust tagav raamistiku osa, Spring MVC - MVC raamistik jm), mis tervikuna moodustavad tugevat infrastruktuuri suurte infosüsteemi ehitamiseks.
- **.NET** - C# keeles raamistik, mis samuti sobib REST veebirakenduste ehitamiseks. Vajalik funktsionaalsus on tagatav vastavate paketide paigaldamisega (nt Enti-

tyFrameworkCore - ORM raamistik, AspNetCore.Authentification.JwtBearer - JWT tokeni kaudu autentimise võimaldamine). Viimastel aastatel on raamistiku populaarsus oluliselt vähenenud.

- **Django** - Python keeles kirjutatud raamistik. On lihtne ja laia funktsionaalsusega, mis on kohe raamistikus saadaval ilma lisamoodulite paigaldamise vajaduseta.

Arendatava infosüsteemi seisukohalt peab tehnoloogia sobivuse hindama järgmiste aspektide seisukohalt:

- kiirus – teenus peab piisavalt kiiresti teostama kõikvõimalikud arvutused, sh kasutades samal ajal andmeid andmebaasist.
- turvalisus – raamistik peab (sisseehitud funktsionaalsus või laiendus) tegelema rakenduse turvalisusega sh kasutajate autentimisega. Raamistik peab toetama ka JWT tokeniga autentimist.
- ORM – raamistikul peab olema *Object Relational Mapping*-uga tegelev moodul, selleks et lihtsustada andmebaasi andmetega tegutsemist koodis.
- arendaja oskused – infosüsteemi arendamisega tegeleval ressurssil peavad olema piisavalt teadmisi ja kogemusi raamistikuga

Raamistikute vastavus eeltoodud kriteeriumitele on toodud tabelis 1.

Table 1. *Backend raamistikute võrdlus*

Raamistik	**Kiirus**	**Turvalisus**	**ORM**	**Oskused**
Laravel	+	+	+	+/-
.NET	+	+	+	+
Spring	+	+	+	+/-
Django	+	+	+	-

Kõik raamistikud omavad vajalikku funktsionaalsust arendatava infosüsteemi ehitamiseks, seetõttu valiku tegemist lähtutakse saadaval oleva programmeerimisressurssi oskuste tasemest erinevate raamistikutega. Sellest lähtuvalt oli tehnoloogiaks valitud C# keeles kirjutatud .NET raamistik.

4.2.3 Andmebaasi juhtsüsteem

Kuna inosüsteemi äriloogika ei eelda suurte andmete mahtude säilimist, seetõttu ka andmebaasi juhtsüsteemi valiku osas on nõuded tagasihoidlikud: *Open Source* tüüpi litsents, et välistada lisakulusid ning võimalus ühendada andmebaasimootor valitud serveriosa

raamistikuga (.NET). Kõige populaarsemad *Open Source* litsentsiga andmebaasimootorid on:

- MySQL
- PostgreSQL
- MariaDB
- MongoDB
- SQLite

Kõikidele ülaltoodud süsteemidele eksisteerivad juhtprogrammid .NET EFCore ühendamiseks, seetõttu võib kõik toodud lahendused pidada sobilikuks. Valikul lähtutakse sellest, mis tehnoloogiaga on programmerijal rohkem teadmisi ja kogemusi. Antud juhul see on PostgreSQL.

4.3 Veebirakenduse arhitektuur

Kavandatava infosüsteemi komponendid on järgmised:

- **server** – renditud pilverserver Ubuntu 20.04 operatsioonisüsteemiga
- **back-end** – infosüsteemi serveriosa .NET 8.0.1, käivitatud Docker-konteinerina serveri operatsioonisüsteemis
- **front-end** – serveri kasutajaliidese osa React v18.2
- **andmebaas** – andmete salvestamiseks, andmebaasimootor PostgreSQL 16.2 paigaldatakse sama pilveserveri operatsioonisüsteemile

Infosüsteemi komponentide omavahelised seosed on toodud pildil 9. Pilveserveri rentimisel tuleb kindlasti arvestada, et teenus peab võimaldama operatsioonisüsteemi haldamist *root*-kasutajana, et oleks võimalik Docker-i kaudu käivitada rakenduse serveriosa ning paigaldada PostgreSQL andmebaasimootor. Samuti tuleb paigaldada Apache server, mis hakkab serveerima kasutajaliidese rakendust ning edastada teatud aadressile tulnud päringud (*reverse proxy*) Docker-konteineris töötavale *backend*-rakendusele.

4.4 Andmebaasi projekteerimine

Arendatavas infosüsteemis kasutatakse relatsioonilist andmebaasi, mille puhul andmeb on koondatud tabelitesse. Andmebaasi primaarvõtmeteks kasutatakse GUID võtmed, mis on 128-biti pikkusega sõne. GUID on piisavalt pikk, et võtme unikaalsus oleks piisava tõenäosusega tagatud. Samuti GUID on genereeritud raamistikuga, mis on kiirem [2].

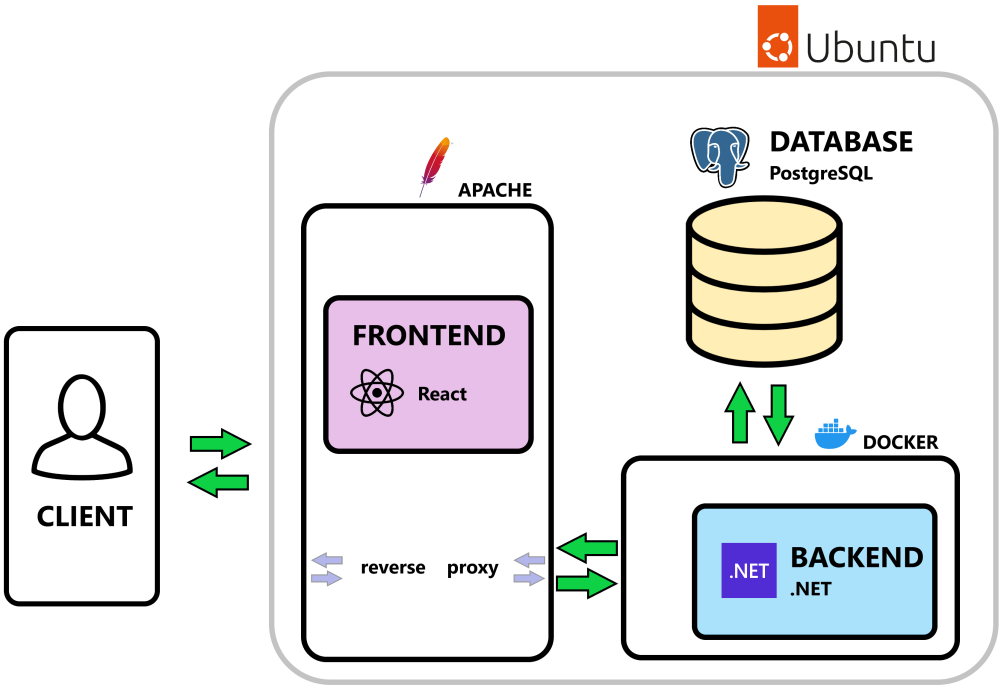

Figure 9. *Infosüsteemi arhitektuur*

Kuna infosüsteemis on kasutusel ORM Entity Framework, siis andmebaasi skeemi luuakse automaatselt koodis ehitatud mudeli järgi. Vaatamata sellele, enne andmemudeli implementeerimist koodis peab läbi mõtlema selle ülesehitust ja loogilisi seoseid andmemudeli objektide vahel. Selle jaoks on koostatud andmebaasi ERD diagrammil, millel esitatakse andmemudelis olevad seosed graafilisel kujul. Andmemudeli diagramm on esitatud pildil X.

4.5 Kasutajaliidese disain

Osa, kus käsitletakse kasutajaliidest ja selle kavandamist

5. Veebirakenduse arendus

Veebirakenduse arendamise osa

5.1 Andmebaas

Andmebaasi tehnoloogia valik, projekt, püstipanek ja seadistamine

5.2 Serveriosa

Rakenduse serveriosa tehnoloogia valik, arhitektuur, koodi näited, deployment protsess, SSL

5.3 Kasutajaliides

Kasutajaliidese tehnoloogia valik, arhitektuur, koodi näited, deployment (Apache, proxy requests to backend)

6. Kokkuvõte

Kasutatud kirjandus

[1] Codecademy Team. *What is REST*. [Online; loetud 10. veebruar 2024]. URL: https://www.codecademy.com/article/what-is-rest.

[2] Alexander S. Gillis. *GUID (global unique identifier)*. [Online; loetud 10. veebruar 2024]. 2021. URL: https://www.techtarget.com/searchwindowsserver/definition/GUID-global-unique-identifier.

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

Mina, Aleksandr Gildi

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

12.02.2024

[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 - Something

```
<!DOCTYPE html>
<html>
<body>

<h1>Example Title</h1>

<p>Some text here</p>

</body>
</html>
```

Lisa 3 – Something Else

Pythagorean theorem

$$x^n + y^n = z^n \tag{1}$$

Normal distribution

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{2}$$