

TALLINNA TEHNIKAÜLIKOOL  
Infotehnoloogia teaduskond

Aleksandr Gildi 201362

**VEEBIPÕHINE EHTUSFÜÜSIKA 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

13.03.2024

## **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

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

U-arv	To-Do( <i>To-Do</i> )
CRUD	To-Do( <i>To-Do</i> )
Unit Of Work	To-Do( <i>To-Do</i> )
Domain	To-Do( <i>To-Do</i> )
DAL	To-Do( <i>To-Do</i> )
BLL	To-Do( <i>To-Do</i> )
Migration	To-Do( <i>To-Do</i> )

# Sisukord

<b>1</b>	<b>Sissejuhatus</b>	<b>9</b>
<b>2</b>	<b>Metoodika</b>	<b>11</b>
<b>3</b>	<b>Probleemi olemus</b>	<b>12</b>
3.1	Probleemi uurimine	12
3.2	Olemasolevad lahendused	14
<b>4</b>	<b>Kavandatava veebirakenduse analüüs</b>	<b>18</b>
4.1	Nõuete defineerimine	18
4.1.1	Funktsionaalsed nõuded	18
4.1.2	Mittefunktsionaalsed nõuded	20
4.2	Tehnoloogiate valik	21
4.2.1	Kasutajaliides	22
4.2.2	Serveriosa	23
4.2.3	Andmebaasi juhtprogramm	26
4.3	Veebirakenduse arhitektuur	26
4.4	Andmebaasi projekteerimine	27
4.5	Kasutajaliidese disain	28
<b>5</b>	<b>Veebirakenduse arendus</b>	<b>34</b>
5.1	Andmebaas	34
5.2	Serveriosa	34
5.3	Kasutajaliides	37
<b>6</b>	<b>Testimine</b>	<b>42</b>
<b>7</b>	<b>Hinnang infosüsteemile</b>	<b>43</b>
<b>8</b>	<b>Kokkuvõte</b>	<b>44</b>
	<b>Kasutatud kirjandus</b>	<b>45</b>
	<b>Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks</b>	<b>46</b>
	<b>Lisa 2 – Kasutajaliidese disain</b>	<b>47</b>

## Jooniste loetelu

1	Mitmest kihist koosneva ehituskonstruksiooni näide . . . . .	12
2	Näide niiskustehnilise analüüsi tulemuste esitamisest tabelis . . . . .	13
3	Näide niiskustehnilise analüüsi tulemuste esitamisest graafikul . . . . .	14
4	Ubakus tarkvara katutajaliides, ekraanitõmmis . . . . .	15
5	Ubakus tarkvara materjalide valik, ekraanitõmmis . . . . .	16
6	Physibel Glasta tarkvara kasutajaliides, ekraanitõmmis . . . . .	16
7	Delphin tarkvara kasutajaliides, ekraanitõmmis . . . . .	17
8	Funktsionaalsed nõuded, kliendi kasutajalood . . . . .	20
9	Funktsionaalsed nõuded, administraatori kasutajalood . . . . .	21
10	Infosüsteemi arhitektuur . . . . .	27
11	Konstruksiooni kihtide andmemudel . . . . .	28
12	Materjali omaduste mudel . . . . .	28
13	Pealehe kaartide disaini näide . . . . .	29
14	Arvutuse kasutaja töövoog . . . . .	30
15	Parameetrite ploki disain . . . . .	31
16	Ühest materjalist koosneva kihi lisamine . . . . .	31
17	Mitmest materjalist koosneva kihi lisamine . . . . .	32
18	Konstruksiooni skemaatiline joonis . . . . .	32
19	Tulemuste esitamine diagrammil . . . . .	33
20	Serveriosa kihtide skemaatiline joonis . . . . .	35
21	<i>PropertyCard</i> materjali omaduste komponendid materjali vormis . . . . .	38
22	Kasutajaliidese komponendi <i>Therm.tsx</i> struktuur . . . . .	40
23	ResultBlock.tsx – tulemuste esitamise plokk . . . . .	41
24	Kasutajaliidese disaain: sisselogimise vorm . . . . .	47
25	Kasutajaliidese disaain: pealeht . . . . .	47
26	Kasutajaliidese disaain: materjalide nimekiri . . . . .	48
27	Kasutajaliidese disaain: materjali lisamise vorm . . . . .	48
28	Kasutajaliidese disaain: kalkulaatori vaade . . . . .	49



## **Tabelite loetelu**

1	<i>Backend raamistikute võrdlus . . . . .</i>	25
---	---	----

# 1. Sissejuhatus

Ehitusfüüsika on ehitusvaldkonna haru, mis käsitleb hoonet füüsikaliste nähtuste seisukohalt: soojus, niiskus, õhk, heli ja valgus. Võib väita, et ehitusfüüsikaga puutub oma elus kokku igaüks, kuna hoone sisekliima mugavus, küttarved ja müra, mis kostub tänavalt tuppa, on samuti lahutamatult seotud ehitusfüüsikaga.

Ehitusfüüsika valdkonna projekteerimise peamised eesmärgid on:

- optimeerida hoone kütte ning jahutuskulud
- tagada hoones soojuslikku mugavust, niiskustingimusi ja sisekliima kvaliteeti terveks
- 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 avaldada negatiivset mõju inimeste tervisele ning 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 – jahutusele.

Ehitusfüüsikaga peab arvestama hoone elutsükli igal etapil – kavandamine, projekteerimine, ehitamine ja haldamine. Hoone kavandamisel arvutatakse välja planeeritavad energiakulud ja määratakse hoone energiaklassi. Hoone projekteerimise faasis peavad ehitusfüüsikaga arvestama arhitektid, konstruktorid ja ka tehnosüsteemide projekteerijad, kes valivad õigete omadustega materjalid ning hindavad nende materjalide koosmõju konstruktsiooni toimivusele. 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 professionaalseid 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 platvormi, mis sisaldaks erinevad tööriistad, millega oleks võimaliks lahendada ehitusfüüsika valdkonna erinevaid ülesandeid mugavalt ja operatiivselt sellisel tasemel, mis oleks ehitusinseneridele piisav. See võiks parandada olukorda, kus ehitusfüüsika probleemidega tegelemine jääb üldse erinevatel etapidel tegemata tarkvara puudumise või tarkvara kasutamiseks ebapiisavate oskuste tõttu. See oleks 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 – detailsem soojusjuhtivuse arvutus erinevatele hoone sõlmedele, energiamärgise või ventilatsiooni- ja õhuvahetusega seotud arvutused, ja ka palju muud. Käesoleva töö raames on mahu piiramise mõttes mõistlik keskenduda ühe konkreetse probleemi lahendamisele ja töötada välja selleks tööriist. Lahendatavaks probleemiks võiks olla konstruktsiooni niiskustehnilise toimivuse kalkulaator, millega saaks lihtsamal viisil hinnata kondensaadi tekkimise riski konstruktsiooni kihtides. Tegemist on ehitusinseneridele vajaliku tööriistaga, mille arendamine võiks samuti olla infotehnoloogia valdkonna seisukohalt huvitavaks väljakutseks.

Püstitatud probleemi lahenduse lähteülesanne suures osas toetub autori teadmistele ja kogemustele ehituse valdkonnas. Töö autor omab magistrikraadi tööstus- ja tsiviilehituse erialal ning on rohkem kui 10 aastat töötanud ehitusvaldkonnas erinevates ametites, seetõttu aimab sihtgrupi vajadustest.

## 2. 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 tehnoloogiad – 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 protsess. 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.

### 3. Probleemi olemus

#### 3.1 Probleemi uurimine

Ehitusfüüsika mõistes ehituskonstruksioon kujutab endast erinevate füüsikaliste omadustega kihtidest koosnevat struktuuri, mis eraldab kaks erinevat keskkonda. Näitena võib tuua kolmekihilist välisseina raudbetoonpaneeli, mis koosneb kolmest kihist: Okandev osa, soojustuse kiht ja betoonfassaad. Paneeli ühel pool on hoone sees olev soe õhk ning teisel pool on väljas olev külm õhk – Pilt 1.

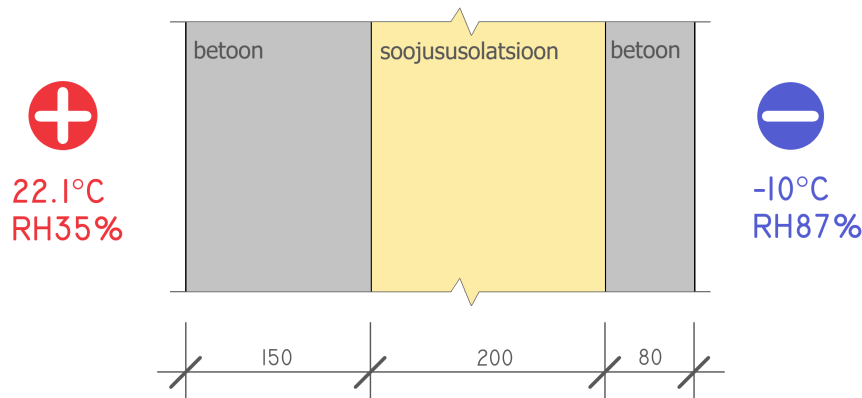


Figure 1. Kihilise konstruktsiooni näide

Sellises olukorras kõige olulisemad materjalide omadused on soojuserijuhtivus  $\lambda$  [W/mK] ja veeaurutakistus, mis võib olla väljendatud mitmel viisil (viise on palju, aga käesolevas töös keskendutakse ainult ühele, kuna see on kõige enam kasutatud nii teadusallikates, kui ka ehitusmaterjalide dokumentides):  $\mu$  - difusioonitakistustegur [1]. Infosüsteemi andmemudeli projekteerimisel tuleb kohe arvesse võtma võimalust lisada tulevikus materjalile ka muud omadused, mis on tarvis teiste tööriistade arendamiseks.

Teatud tingimustel võib tekkida olukord, kui konstruktsiooni ristlõikes on mingis punktis suhteline niiskus piisavalt kõrge, et soodustada kondensaadi tekkimist [1]. Nimetatud olukord on ohtlik nii ehituskonstruksioonile, mis võib pikaajalise niiskuse mõjul laguneda, kui ka inimese tervisele, sest konstruktsioonide sees tekkiv hallitus on siseruumide õhku sattuvate bakterite allikaks [1]. Seda, kuidas konstruktsioonis toimuvad niiskuse leviku protsessid sõltuvalt materjalide omadustest ja paiknemisest ning ümbritsevatest tingimustest, nimetatakse konstruktsiooni niiskustehniliseks toimivuseks. Arvutust, mille abil võib neid protsesse modelleerida ja kondenseerumise riski hinnata, nimetatakse niiskustehnilise toimivuse analüüsiks.

Tegemist on klassikalise ehitusfüüsika ülesandega, mille lehandemiseks peab ette võtma järgmiseid samme [2]:

- konstruktsiooni kihtide soojustakistuse ja konstruktsiooni summaarse soojustakistuse arvutus
- 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

Lahendades ülesannet käsitsi, kasutatakse tabeli meetodit: koostatakse tabel, mille ridadele pannakse kirja konstruktsiooni kihid ja veergudesse arvutatakse samm sammult väärtused. Kuigi arvutused ei ole keerulised (valemid nimetatud arvutuste teostamiseks on leitavad viidetud allikatest [ToDo]), paraku käsitsi arvutamine võtab palju aega, kuna igasugune muudatus (kihi lisamine või järjekorra muutmine) tähendab kogu tabeli ümber arvutamist. Pildil 2 on toodud arvutustabeli näide seitsmest kihist koosneva konstruktsiooni puhul.

Arvutustabel											
	Kihi paksus	Soojus-erijuhtivus	Kihi soojustakistus	Temp. muutmine	Temp. kihi piiril	Küllastusrõhk	Veeauru-erijuhtivus	Veeaurutakistus	Rõhkude erinevus	Veeauru osarõhk	
	$d$ [mm]	$\lambda_{ef}$ [W/(mK)]	$R$ [m <sup>2</sup> K/W]	$\Delta t$ [°C]	$t$ [°C]	$p_{sat}$ [Pa]	$\delta_p$ [kg/msPa]	$Z_p$ [m <sup>2</sup> sPa/kg]	$\Delta P$ [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
Väisõhk					-5.8	375					322.1
Kokku			7.5					5.22776E+14			

Figure 2. Arvutustabeli näide

Analüüsi tulemused esitatakse graafiliselt diagrammi kujul, mille  $x$  teljel väärtused on punkti asukoht konstruktsiooni ristlõikes ning  $y$  teljel on temperatuuri, veeauru küllastus- ja osarõhu väärtused vastavas punktis – näide diagrammilt on toodud pildil 3.

Graafik annab head visuaalsed ülevaated konstruktsiooni kihtides toimuvale. Täpsemalt öeldes, peab vaatama veeauru küllastus- ja osarõhkude jaotuse graafikuid. Veeauru osarõhk iseloomustab veeauru tegelikku kontsentratsiooni teatud punktis, küllastusrõhk iseloomustab veeauru kontsentratsiooni, mille ületades hakkab veeaur kondenseeruma. 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 nimetatakse kastepunktiks konstruktsioonis. Näide on toodud pildil 3: vasakul on niiskustehniline olukord hea, kuna aurutõke asub õiges kohas ning seetõttu osarõhk on konstruktsiooni ristlõige kogu ulatuses jääb küllastusrõhult turvaliselt allapoole. Parempoolisel pildil paikneb konstruktsiooni külmemal pool veeaurutihe kiht, mille tõttu läheb osarõhk kõrgeks ning ületab küllastusrõhu väärtust. Tsoonis, kus osarõhk on küllastusrõhust kõrgem, esineb kondensaadi tekkimise oht (pildil tsoon on viirutatud helesinisega).

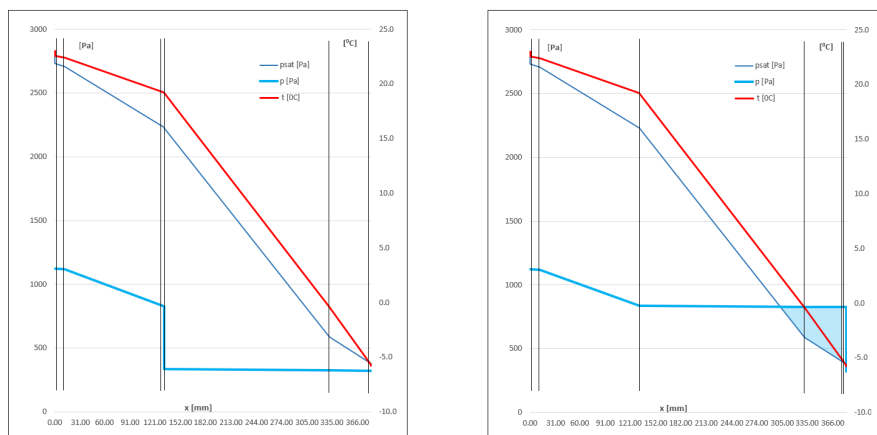


Figure 3. Näide tulemuste esitamisest graafikul

Kuigi probleem on 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. Lisaks sellele, näidatud arvutus kehtib vaid ühe välisõhu parameetrite kombinatsiooni puhul (temperatuur ja suhteline niiskus), ülevaatliku pildi saamiseks olukorda peab hindama aasta lõikes, mis tähendab, et peab arvutust kordama iga kuu kliimaandmetega.

### 3.2 Olemasolevad lahendused

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.

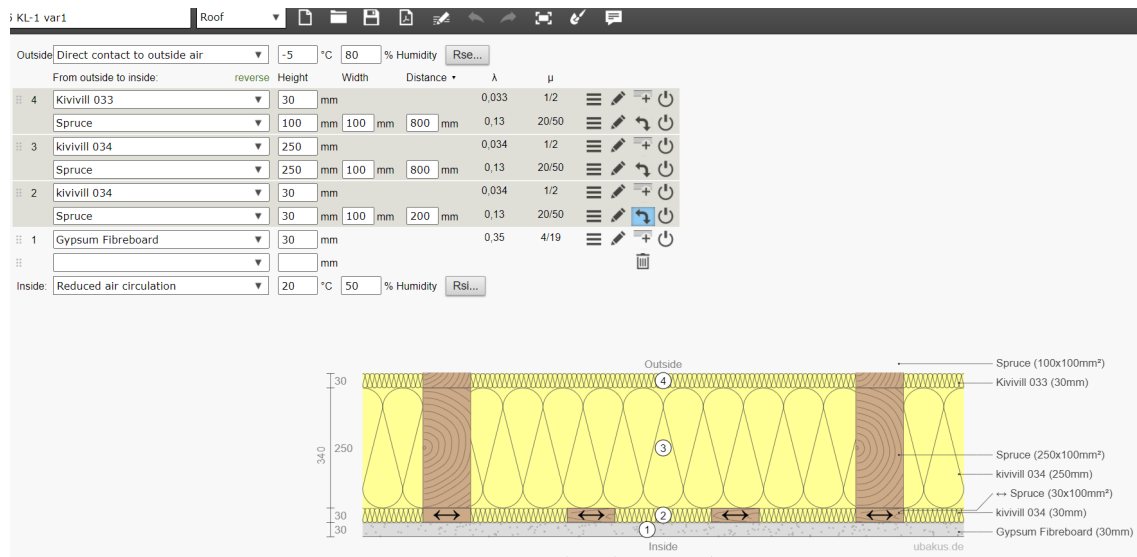


Figure 4. *Ubakus: kasutajaliides, ekraanitõmmis.*

**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 mittehomoogeensete (mitemest erinevast materjalist, nt puitsõrestiksein) kihtidega konstruktsioone – pilt 4 [3].

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, mineraalvill), osa on reaalsed turustatavad tooted (näiteks: Isover soojusisolatsioonide valik) – pilt 5. Võib puuduseks pidada seda, et osa materjale (konkreetsed tooted) 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.

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 paketest. 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.



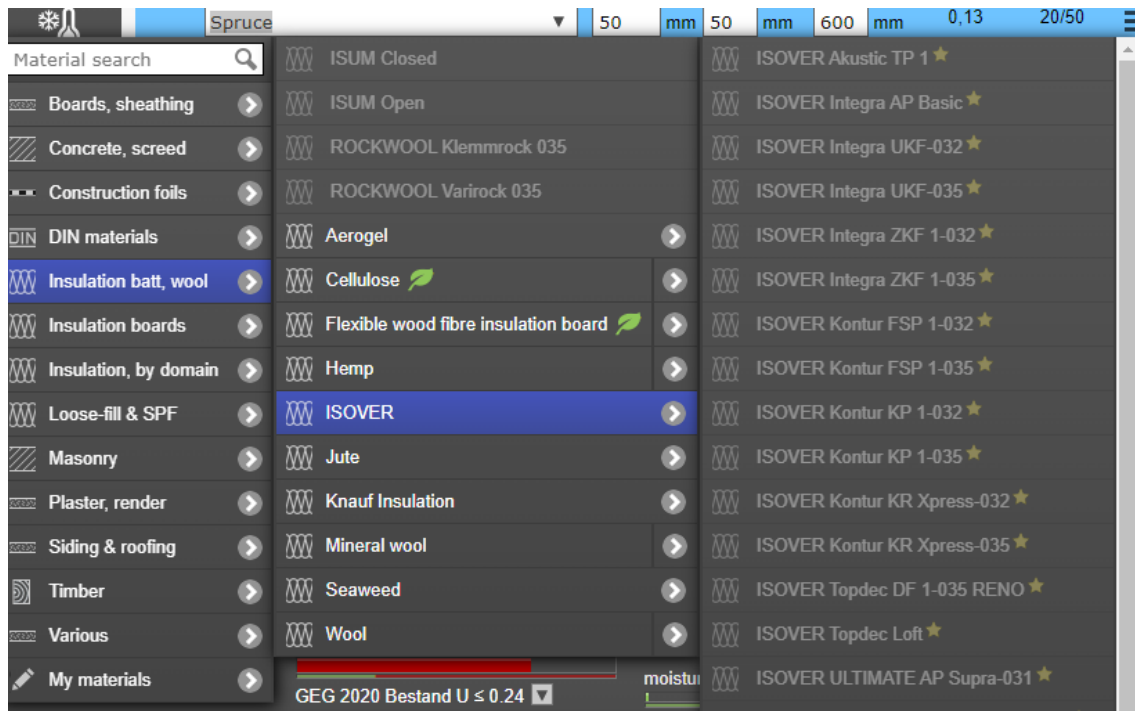


Figure 5. Ubakus: ehitusmaterjalide valik baasis, ekraanitõmmis.

**Physibel Glasta** on üks analoogne lahendus veel, mis on samuti komertstarkvara. Tegemist on samuti arvutile paigaldatava tarkvaraga, mille kasutajaliides on veidi keerulisem ja ka disain on oluliselt konservatiivsem (pilt 6) [4].

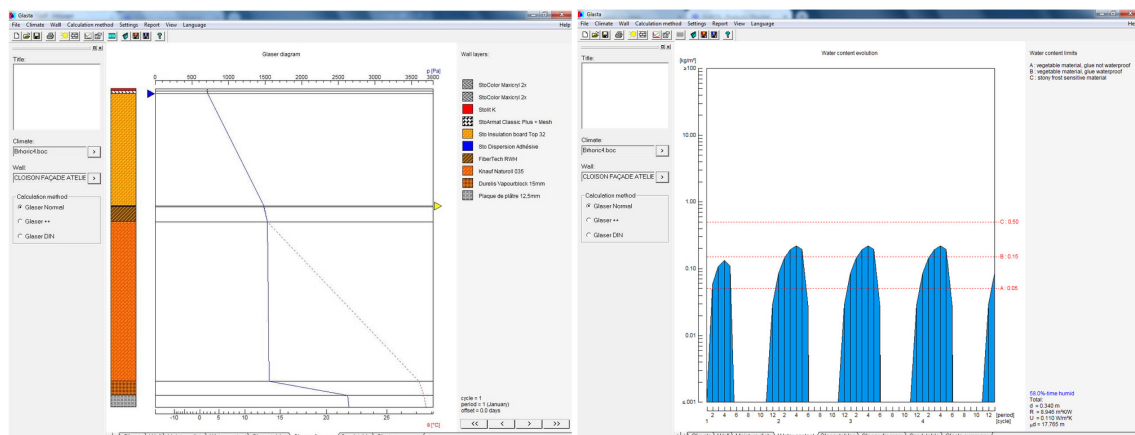


Figure 6. Glasta: kasutajaliides, ekraanitõmmis.

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 [5]. Eelisteks on väga lai funktsionaalsus ning ka täielik vabadus konstruktsiooni mudeldamisel. Erineval eeltoodu analoogidest, tarkvara võimaldab koostada ja analüüsida konstruktsiooni sõlme. Samuti on tarkvaras võimalik mudeldada analüüsi lähteandmeteks olevad ilmaandmed. Tavakasutaja jaoks 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 nendele, kellel ehitusfüüsika arvutused on põhitegevuseks (nt tarkvara on laialt kasutusel teadusvaldkonnas).

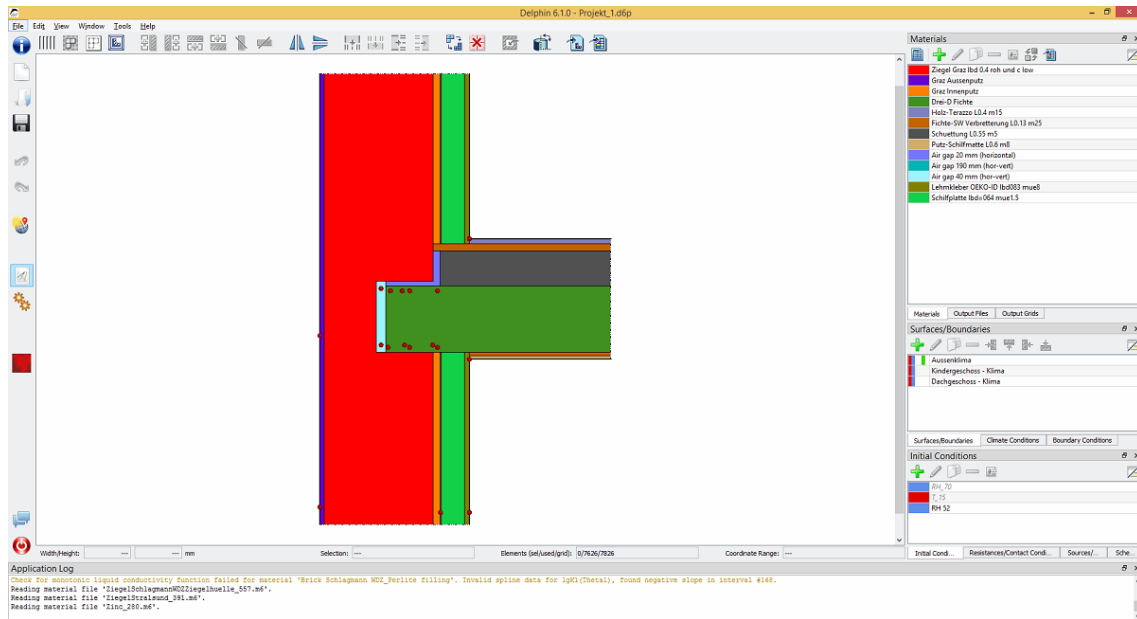


Figure 7. Delphin: kasutajaliides, ekraanitõmmis.

## **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. Nõuete sõnastamisel on aluseks töö autori isiklik kogemus ja teadmised sihtgrupi vajadustest ja nõuetest.

#### **Infosüsteemi kasutajal peab olema võimalus:**

- registreerida endale konto ja 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 administraatoril peab olema võimalus:**

- 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 8):

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

Administraatori kasutajalood on jagatud kolmeks featuuriks (pilt 9):

- 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 mittehomoogeensete 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 8. 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 uuen-

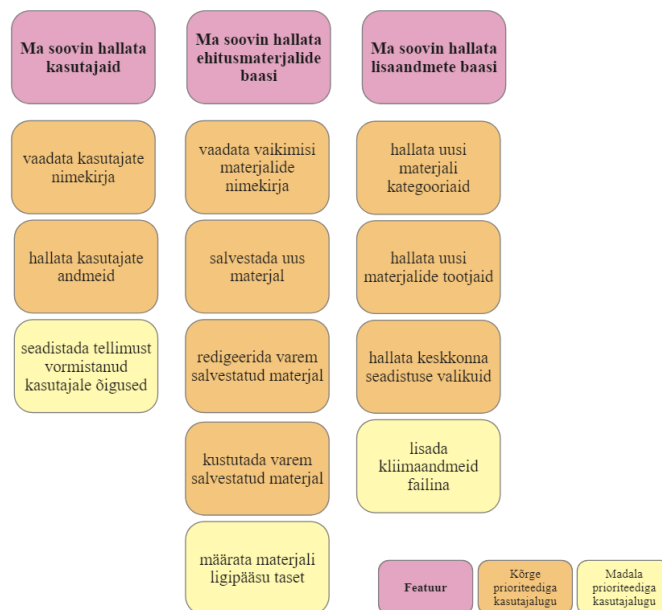


Figure 9. *administraatori kasutajalood*

damist kohe peale lähteandmete (kihid, parameetrid) muutmist, pikk ooteaeg või lehe ümberlaadimine tulemuste näitamiseks ei ole aktsepteeritav;

- 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 kaasaegne, st välimus ja veebilehe struktuur peavad vastama kaasaegsetele UX/UI printsiibidele.

## 4.2 Tehnoloogiate valik

Tehnoloogiate valimisel lähtutakse kaasaegsetest 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: mobiilirakendus). 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[6]. Lisaks sellele on oluline, et kõik päringud on omavahel sõltumatud, see tähendab, et ühe päringu raames serveriosas alustatakse ja lõpetatakse kõik päringuga seotud protses-

sid, päringute vahel serveril puudub kliendiga seotud olek (*stateless* protokoll)[6].

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 [7]. 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 [8]. 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. Spetsiifilise funktsionaalsuse jaoks kasutatakse eraldi pluginaid ja teeke. Näiteks päringute saatmise ja serveri vastuse töötamiseks 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 šabloon 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 šabloon kujundamiseks kasutatakse HTML, mis siseldab Vue-spetsiifilist süntaksi, mille abil juhib raamistik lehe logikat. Vue rakendus koosneb SFC komponentidest, igas komponendis 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 tõenäoliselt 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 töötlustega ja andmete saatmisega. Samuti *backend* osa suhtleb andmebaasiga, küsides ja redigeerides andmeid. Rakenduse serveriosa on võimalik implementeerida kasutades järgmiseid programmeerim-



iskeeli:

- *PHP* – populaarne 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 kasutakse 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 vaikumisi, 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 spetsiifilisi 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 situatsioonides saab pidada nii eeliseks kui ka puuduseks.

Rakenduse ehitamiseks on otstarbekas kasutada analoogselt kasutajaliidesega raamistikku. Kõikidel ülaltoodud programmeerimiskeelidel 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 EntityFrameworkCore - ORM raamistik, ASP.NET Core.Authentication.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 tegelevatel ressurssidel 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 juhtprogramm

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 EF Core ühendamiseks, seetõttu võib kõik toodud lahendused pidada sobilikuks. Valikul lähtutakse sellest, mis tehnoloogiaga on programmeerijal 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 paigaldakse sama pilveserveri operatsioonisüsteemile

Infosüsteemi komponentide omavahelised seosed on toodud pildil 10. 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.

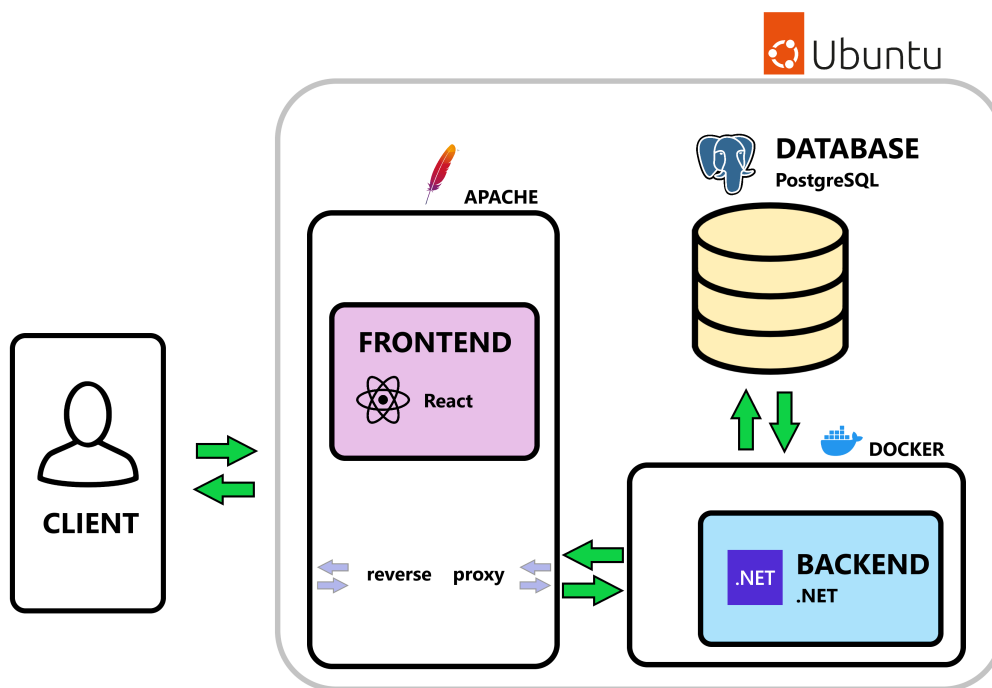


Figure 10. Infosüsteemi arhitektuur

#### 4.4 Andmebaasi projekteerimine

Arendatavas infosüsteemis kasutatakse relatsioonilist andmebaasi, mille puhul andmed 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 [9]. Kuna infosüsteemis on kasutusel ORM Entity Framework, siis andmebaasi skeemi luuakse automaatselt koodis ehitatud mudeli järgi. Vaatamata sellele, enne andmemudeli realiseerumist koodis peab läbi mõtlema selle ülesehitust ja loogilisi seoseid andmemudeli objektide vahel.

Kuigi minimaalse elujõulise toote arendamise raames on otsustatud arvutusprogrammist elimineerida mittehomoogeensete (mitmest materjalist koosnevate) kihtide arvutamist, siiski tuleb sellise võimalusega arvestama tulevikus. Selle jaoks on andmemudelis olemas alamkihi objekt (*SubLayer*). Iga konstruktsiooni kiht võib koosneda mitmest alamkihist. Näiteks paksusega 200 mm puitsõrestik seinas 50 mm paksusega puitpostide vahel paikneb 550 mm mineraalvilla. Sellisel juhul oleks kihi (*MainLayer*) sees oleks kaks alamkihti, üks on paksusega 50mm, mille materjaliks on puit ning teine on paksusega 550 mm, mille materjaliks on vill. Selline struktuur võimaldab mudeldada peaaegu kõikvõimalikud ehituses kasutusel olevad lahendused. Kuna esialgu antud funktsionaalsust ei tehta, luuakse igakord uus kiht ühe alamkihiga. Sellisel juhul ka tulevikus süsteem toetab varem loodud konstruktsioonide arvutamist. Kihtide mudeldamist käsitlev diagrammi osa on toodud

pildil 11.

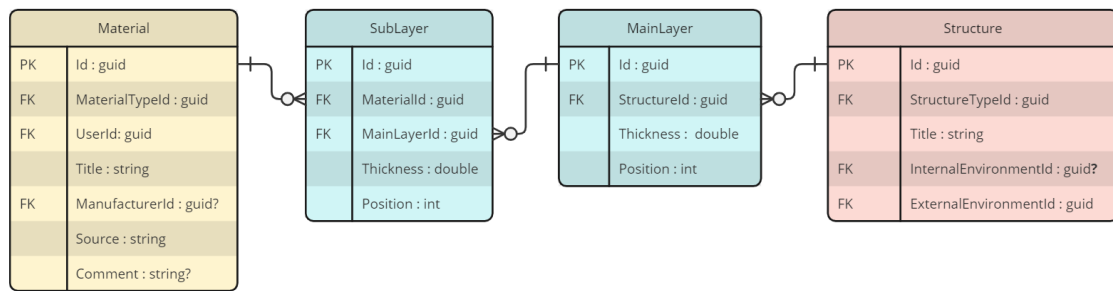


Figure 11. *Konstruksiooni kihtide andmemudel*

Selleks, et konstruktsioonide ehitusfüüsikaline arvutamine oleks võimalik, süsteemis peavad olema ka materjali füüsilised omadused: soojusjuhtivus ja üks veeaurujuhtivust kirjeldavatest omadustest. On aga ka võimalik, et tulevikus tööriistakasti lisatakse teised arvutusprogrammid, mille jaoks on tarvis teada teisi omadusi. Selleks, et võimaldada tulevikus luua vajadusel uusi materjalide omadusi, andmebaasimudelil on olemas järgmised objektid: omadus (*Property*) ja materjali omadus (*MaterialProperty*) - pilt 12. Lisaks, selline mudel võimaldab säilitada omaduste muutuste ajalugu. Selleks, et arvutused oleksid usaldusväärsed, peab igal materjali omadusel olema märge sellest, kas tegemist on verifitseeritud andmetega ja mis on andmete allikas (nt tootja dokumentatsioon, teadusartikkel).

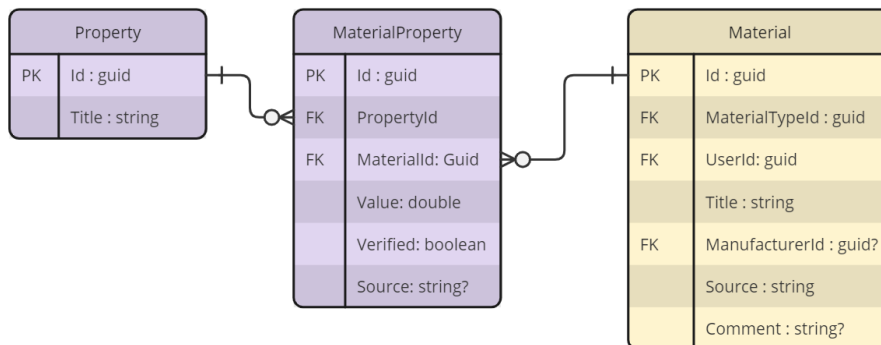


Figure 12. *Materjali omaduste mudel*

Muus osas on andmemudeli ülesehitus on võrdlemisi lihtne. Andmemudeli diagramm tervikuna on esitatud käesoleva töö Lisas X. Diagrammil puudub kasutaja, kasutajarollidega ja autentimisega seotud andmebaasi osa, kuna valitud raamistik (.NET) haldab seda ise.

## 4.5 Kasutajaliidese disain

MVP funktsionaalsusega (osa 4.1) prototüübi realiseerimiseks on vaja luua veebirakenduse kasutajaliidesele järgmised vaated:

- Menüüriba

- Registreerimise ja sise logimise vaated
- Dashboard-vaate
- Kasutaja andmete vaade
- Ehitusmaterjalide nimekirja vaade
- Ehitusmaterjali loomise ja redigeerimise vaade
- Kalkulaatori vaade
- Kasutajate nimekiri (administraator)
- Parameetrite ja lisaandmete vaade (administraator)

Mõned vaated on rollispetsiifilised (nt kasutajate nimekirja administraatori vaade), teised on universaalsed - kasutajarolli alusel kasutajaliideses otsustatakse, kas näidata (või lubada) midagi kasutajale või mitte. Igal tegevusel kontrollitakse kasutajaõigusi täiendavalt ka serveriosas enne tegevuse algust.

Kasutajaliidese disaini projekteerimine on teostatud veebipõhise tarkvaraga **Figma**. Figma võimaldab mugaval viisil luua ja redigeerida lehtede visuaalsed prototüübid ja seostada erinevad vaated omavahel.

Arendatava veebirakenduse kasutajaliidese disain on minimalistlik, aga samas kaasaegne. Kasutatakse värvilisi ikoone ja värve elementide kujunduses, et vältida ülemääraselt ametliku mulje tekkimist. Värvitoonid on valitud selliselt, et välimus oleks lakooniline ja professionaalne.

Pealehel (*Dashboard*) näidatakse kaardid, mille kaudu kasutajat juhitakse tööriistadele ja moodulitele, millele on kasutajal ligipääs. Administraatori ja tavakasutaja *Dashboard*-vaated on vastavalt erinevad. *Dashboard*-kaartide näide on toodud pildil 13.

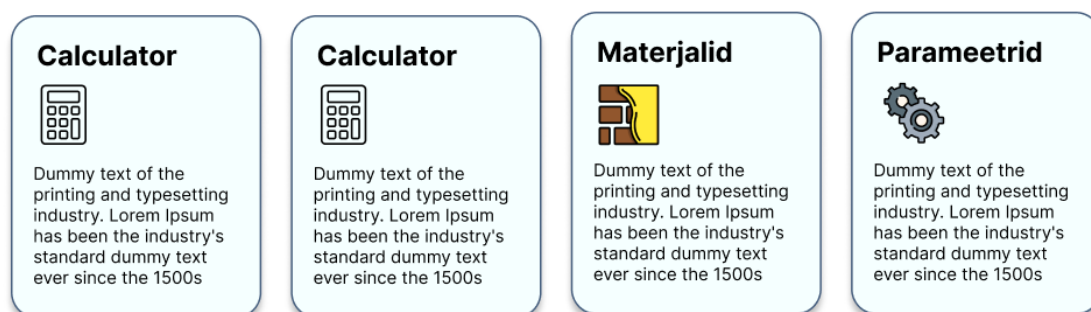


Figure 13. *Pealehe kaartide disaini näide*

Kõige keerulisem kasutajaliidese osa on kalkulaatori vaade. Vaade on jaotatud kolmeks loogiliseks osaks: arvutuse parameetrid, konstruktsiooni kihtide komplekteerimine ja

tulemuste graafiline esitamine. Kalkulaatori töövoog on esitatud Pildil 14.

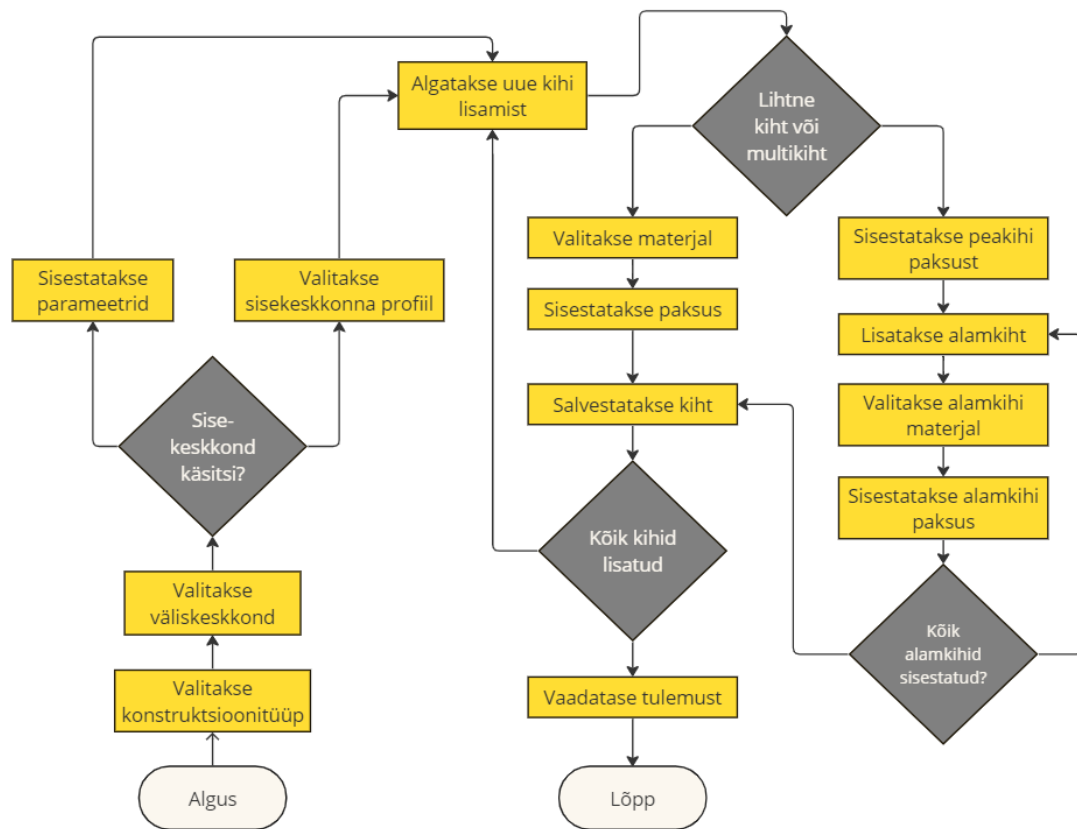


Figure 14. Arvutuse kasutaja töövoog

Parameetrite plokk on esimene, millega peab kasutaja tegelema - valitakse arvutuse parameetrid: konstruktsiooni tüüp, väliskeskkond. Samuti valitakse, kas sisekeskkonna parameetrid sisestatakse käsitsi või arvutatakse automaatselt välja kasutades Eesti elamute sisekliima mudelit. Esimesel juhul kasutajale näidatakse väljad temperatuuri ja niiskuse sisestamiseks, teisel juhul näidatakse valiklist sisekeskkonna profiilidega (EN ISO 13788 kohase klassifikatsiooni järgi). Samuti selles ploki on esitatud ka arvutuse üldised arvulised tulemused - konstruktsiooni summaarne soojusjuhtivus (U-arv) ja veeaurutakistus. Parameetrite ploki disain on näidatud Pildil 15.

Järgmine tegevus, mis peab olema tehtud tulemuste näitamiseks on konstruktsiooni mudeldamine, mis kujutab ennast erinevatest materjalidest ja erineva paksusega kihtidest nimekirja koostamist. Kasutaja lisab, muudab või kustutab nimekirjas olevad kihid. Uue kihi lisamiseks vajutab kasutaja vastavat nuppu, mille järel näidatakse uue kihi parameetrite vormi. Vormist peab kasutaja valima uue kihi materjal, paksust ning salvestada kiht. Kui uue kihi lisamine (või olemasoleva kihi redigeerimine) on algatatud, siis uue kihi lisamise nupp on deaktiveeritud, et suunata kasutajad tegelema vaid ühe kihiga korraga. Lihtsa

### Parameetrid

Konstruksiooni tüüp

Välissein

Väliskeskkond

Tallinn, Jaanuar (-5.3C, 87%)

☒ arvuta siseõhu parameetrid automaatselt sisekliima mudelist

Elamud asustustihedusega kuni 30 m2 inimese kohta

Soojusjuhtivus U

0 W/mK

Veeaurutakistus Z<sub>p</sub>

0×10<sup>12</sup> m²sPa/kg

Figure 15. Parameetrite plokki disain

(ühest materjalist koosneva) kihi lisamise näide on toodud Pildil 16.

### Konstruksiooni kihid

Materjal	Paksus [mm]	λd [W/mK]	μ [kg/msPa]	
Armeeritud betoon	200 mm	2	130	
<div>Vali materjal</div>	<div>0</div>			<div>Salvesta</div>

Lisa kiht

Kombineeritud kiht

Figure 16. Ühest materjalist koosneva kihi lisamine

Mittehomogeense (mitmest erinevast materjalist koosneva) kihi lisamine eeldab keerulise-mat töövoogu. Kasutaja kõigepealt peab määrama peakihi paksust (konstruktsiooniga põiki suunas), seejärel lisada vajalikku arvu alamkihte ja määrama alamkihtide materjalid ja paksused. Alamkihi paksuseks loetakse materjali paksust konstruktsiooniga piki suu-nas. Kui kõik andmed on valitud, siis uus kiht salvestatakse vastava nuppu vajutamisega. Mitmest materjalist koosneva kihi lisamise näide on toodud Pildil 16.

Tulemuste esitamiseks tehakse kalkulaatori lehele kaks plokki. Üks neist on mudeldatud konstruktsiooni skemaatiline joonis, mille peal on näidatud konstruktsiooni kihid paksuste väärtustega ja kasutatud materjaliga. Kihtide tausta värv määratakse sõltuvalt materjali tüübist (nt betoonid - hall, soojusisolatsioonid - kollane). Tulevikus joonise peale peab lisama temperatuuri ja veeauruosarõhkude jaotuse graafikud, kuid prototüübi puhul antud koht on lihtsustatud ja graafikud on toodud eraldi plokis. Graafikute ehitamiseks kasu-tatakse valmislahendust, nt Javascript-i põhine teek Graph.js. Tulemuste plokide disain on näidatud piltidel 18 ja 19..

Ülejäänud lehed – ehitusmaterjalid, parameetrid, kasutajad – kujutavad ennast võrdlemisi



Konstruksiooni kihid



Materjal	Paksus [mm]	$\lambda_d$ [W/mK]	$\mu$ [kg/msPa]	
Armeeritud betoon	200 mm	2	130	 
<div> <div>Kombineeritud kiht</div> <div>+ alamkiht</div> <div>250 mm</div> <div>Salvesta</div> </div>				
<div>Puit</div> <div>Mineraalvill</div>	<div>50</div> <div>550</div>			
<div>Lisa kiht</div> <div>Kombineeritud kiht</div>				

Figure 17. Mitmest materjalist koosneva kihi lisamine

Konstruksiooni skemaatiline joonis

Soojusjuhtivus U

0 W/mK

Veeaurutakistus  $Z_p$

$0 \times 10^{12}$  m<sup>2</sup>sPa/kg

beton 80 mm

soojustus 200 mm

beton 200 mm

Figure 18. Konstruksiooni skemaatiline joonis

lihtsaid CRUD-tüüpi kasutajaliideseid, millistel keeruline loogika puudub. Kõikide lehtede disaini projektid on esitatud käesoleva töö **Lisas X**.

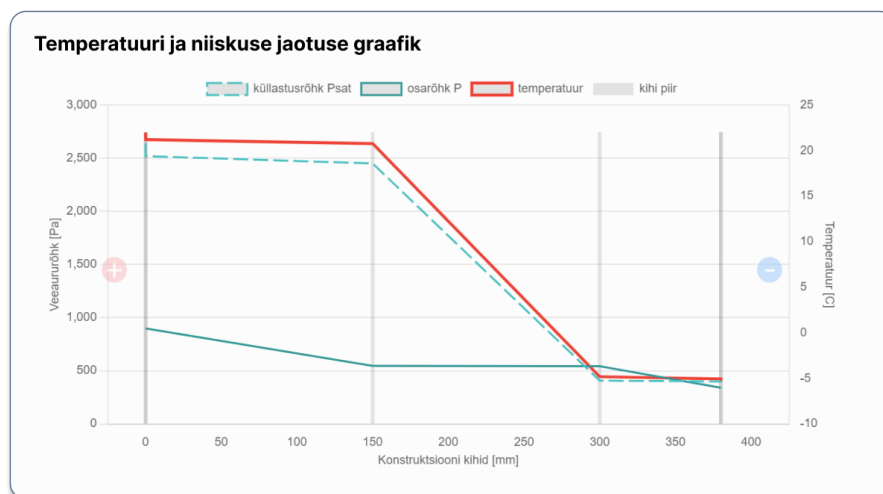


Figure 19. Tulemuste esitamine diagrammil

## 5. Veebirakenduse arendus

Töö käesolevas osas käsitletakse infosüsteemi arendusega seotud aspekte. Alampeatükides on toodud üldine info ja põhimõtted vastava infosüsteemi osa realiseerimisest. Samuti mõnede kriitilisemate elementide puhul on toodud ka detailne lahendus ja lahenduse valiku põhjendus.

### 5.1 Andmebaas

Vastavalt käesoleva töö osale 4.2.3 projekti realiseerimiseks on valitud PostgreSQL versioon 16. Andmebaasi juhtprogramm paigaldatakse samale serverile, milles käivitatakse ka ülejäänud infosüsteemi osad. PostgreSQL on Ubuntu paigaldamiseks saadaval *apt* repositooriumil. Peale paigaldamist redigeeritakse PostgreSQL konfiguratsiooni - *postgresql.conf* ja *pg hba.conf*, seadistades porti, mida kuulatakse (5432) ning IP aadressid, millistelt võetakse päringuid vastu.

Turvalisuse tagamiseks luuakse arendatavale rakendusele eraldi PostgreSQL kasutajat parooliga, piirates kasutaja õigusi ühe konkreetse andmebaasiga, mis on rakenduse funktsioneerimiseks tarvis. Andmebaasile tehakse iganädalaseid varukoopiaid, mida salvestatakse serveril. Selle jaoks luuakse skripti, mida määratakse Ubuntu Cron regulaarse tööna.

### 5.2 Serveriosa

Serveriosa arhitektuur on kihiline – andmete töötlus alates andmebaasist küsimisest kuni kasutajale saatmiseni toimub neljas kihis:

- andmebaasi ja domeeni kiht,
- andmete ligipääsu kiht,
- äriloogika kiht,
- andmete representeerimise kiht.

Kihtide eesmärk on selgelt eristada andmete töötlust loogika ja otstarve järgi. Andmete representeerimise kiht tegeleb andmete kasutajaliidesele saatmise ja kasutajaliideselt vastuvõtmisega ning ei pea muretsema sellest, mis toimub andmetega äriloogika kihis. Analoogselt ka andmete ligipääsu DAL kihis teostatakse ainult andmete küsimist ja salvestamist and-

mebaasile (antud töö raames – raamistiku kaudu), võimalikult välistades igasugused infosüsteemi äriloogikaga seotud tegevused. Kihid on näidatu pildil 20.

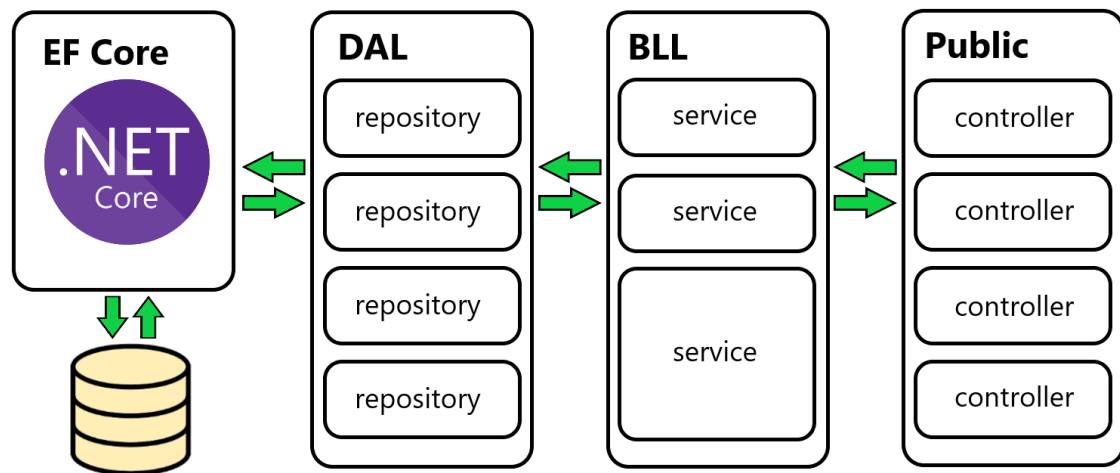


Figure 20. Serveriosa kihtide skemaatiline joonis

**EF Core** ja **Domain** kihis realiseeritakse projekteeritud andmebaasi mudel vastavate klassidena (näiteks *Domain.App.Material*, *Domain.App.Material-Category*, *Domain.App.Material-Property* jne). Igas klassis määratakse olemitele vajalikud omadused ja olemite vahelised seosed. Domeeni kiht moodustab andmebaasi konteksti (*DbContext*), mis on *Entity Framework Core* raamistiku klass, mille kaudu raamistik suhtleb andmebaasiga. Raamistiku kaudu luuakse andmebaasi genereerimise ja uuendamise skripte - migratsioonid (*migration*). Kui üks olemite klassidest oli muudetud, siis uue migratsiooni tegemisel võrdleb raamistik uut olekut eelmisega ning genereerib uue skripti.

**DAL** andmete ligipääsu kihis teostatakse tööd andmebaasi konteksti objektiga (*DbContext*): andmeid küsitakse *Entity Framework Core*-ist ja vastavalt ka salvestatakse *DbContext*-isse. Kiht on jagatud üksusteks – reposiooriumideks (*Repository*), mis on baasimplementatsioonis kujutavad ennast klassikalist CRUD-tüüpi repositooriumi. Samuti repositooriumis tehakse andmetele ligipääsu kontrolli kasutaja ID alusel - nt meetod *GetAllAsync(Guid uid, bool includePublic = false)* vaikimisi küsib andmebaasist kõik olemid, mis kuuluvad kasutajale ID-ga *uid* ning valikuliselt lisatakse ka olemid, mis on ette nähtud ühiskasutuseks. **TODO: Repositooriumi näidiskood, töö Lisa X**

Repositooriumiteks jagamist teostatakse andmemudeli ülesehituse alusel – iga olemit jaoks on eraldi repositoorium. Selleks, et äriloogika kihis erinevates teenustes oleks repositooriumite kasutamine paindlik, moodustatakse kõikidest repositooriumitest üks objekt (*UOW - Unit Of Work* muster), mille kaudu on võimalik juurde pääseda igale repositooriumile.

Kui rakendus vajab mõne olemi puhul keerulisemat repositooriumi loogikat, siis baas-funktsionaalsus saab olla üle kirjutatud või laiendatud. Näiteks, materjalide (*Domain.App.Material*) puhul on otstarbekam kohe agregeerida andmeid, mis puudutavad materjalide omadusi (*Domain.App.MaterialProperty*) ja (*Domain.App.Property*)), siis äriloogika kihis ei pea tegema lisapäringuid, et teostada vajalikke arvutusi ja tegevusi.

**BLL** äriloogika kihis toimub andmete põhiline töötlus, mis on seotud vahetult rakenduse funktsionaalsust puudutava loogikaga. Äriloogika kihi tööüksuseks on teenus (*Service*). Teenuste moodustamise loogika suuresti vastab **DAL** kihi jagamise loogikale, et võimalikult isoleerida erinevad teenused omavahelt. On ka teenused, mille eesmärk on erinevatest repositooriumitest andmete agregeerimine – näiteks, arvutuste teenus (*CalculationService*), mille otstarve on arvutusteks vajalike andmete komplekteerimine kasutajaliidesele saatmiseks, kasutajaliidest tulnud arvutuse päringu töötlemine, arvutuste teostamine ja arvutuse tulemuste tagastamine. *CalculationService* on äriloogika kihi kõike mahukam teenus. Kuna kalkulaatori tööks vajalike andmete maht on suur (materjalide andmed, konstruktsioonide tüüpidega seotud andmed, sise- ja välistingimuste andmed), ei ole otstarbekas neid kasutajaliidese poolelt küsida eraldi päringutega. Teenuses koostatakse andmete komplekt (*dataset*), mida saadetakse kasutajaliidesele ühe päringuga.

**Public** andmete representeerimise kiht tegeleb päringute vastuvõtmisega ja vastuste saatmisega. Kasutajaliidest tulnud andmed valideeritakse, kontrollides etteantud mudelile vastavust, teisaldatakse äriloogika kihi andmeedastus objektiks ning antakse üle vastavale äriloogika kihi teenusele. Kuna rakenduse kõigis kihis teatud vea tekkimisel võib programm visata erindeid, siis tegeleb kiht ka veahaldusega. Vea tekkimisel püüakse seda try-catch plokiga kinni ja saadetakse kasutajaliidesele informatiivset vastus.

Andmebaasiga ühendust haldab raamistik ise. Programmi käivitamise konfiguratsiooni faili (*appsetting.json*) kaudu antakse raamistikule andmebaasiga ühenduse sõne (*connection string*), mis sisaldab andmebaasi, kasutajat ja parooli.

Kuna veebirakendus peab kasutama HTTPS protokoll, peab tegema ka vastavad seadistused Asp.NET raamistikus. Asp.NET kasutab päringute saatmiseks-vastuvõtmiseks Kestrel programmi, mis HTTPS protokollil töötamisel vajab serveri SSL sertifikaati. Sertifikaadi asukoht ja parool antakse raamistikule käivituse konfiguratsiooni failis (*appsetting.json*), millest neid loetakse ja antakse raamistikule programmi käivitamisel failis *Program.cs*: **koodinäide**.

### 5.3 Kasutajaliides

Vastavalt peatükile 4.2.1 kasutajaliidese realiseerimise tehnoloogiaks on valitud React.Js raamistik.

Kasutajaliidese rakendus koosneb *React JSX.Element* komponentidest ja teenustest. Komponentid tagavad rakendusele nõutud funktsionaalsust ja välimust ning teenused vahendavad andmeid komponentide ja serveriosa vahel.

Rakenduse koosseisus on palju erinevaid komponente, suureb osa nendest on lihtsamad komponendid või tavalised CRUD-loogikaga komponendid, mistõttu ei ole nende realiseerimise käesolevas peatükis eraldi kirjeldatud. Koodibaas täismahus on vaadatav Lisas X oleva viitega koodi repositooriumile. Käesolevas peatükis kirjeldatakse vaid mõned komponendid, mis infosüsteemi äriloogika seisukohalt vajavad kompleksset ülesehitust ja funktsionaalsust – ehitusmaterjali lisamise vorm ja kalkulaator.

Materjali loomiseks ja redigeerimiseks on luuakse vorm, mis täidab samaaegselt mõlemat rolli. Kui vormi avamisel oli antud ehitusmaterjali ID, siis laetakse vastav materjal andmebaasist redigeerimiseks, vastasel juhul vormi avamisel luuakse serveriosale päringuga uus materjal mustandi staatusega, mida hakatakse vormis redigeerima. Materjali vorm koosneb kahest React komponendist - vormi põhi ja materjali omaduse plokk. Vormi põhi on suurem komponent *MaterialForm.tsx*, mille kaudu redigeeritakse materjali staatilised väljad:

- nimetus (*title*) – tekstiväli,
- materjali tüüp (*materialCategoryId*) – valiklist materjalide kategooriatest,
- tootja (*manufacturerId*) – valiklist materjalide tootjatest,
- allikas (*source*) – tekstiväli,
- viide allikale (*link*) – tekstiväli,
- kommentaar (*comment*) – tekstiväli.

Materjalide omaduste (nt tihedus, soojusjuhtivus) arv on muutuv – mõne materjali puhul võivad olla salvestatud kõik omadused, teise materjali puhul vaid mõned neist. Lisaks sellele, tulevikus võib infosüsteemile olla lisatud võimalus salvestada muud omadused, mis on tarvis teiste arvutuste jaoks. Sellest tingituna peab olema võimalus kuvada ja redigeerida need omadused, mis on materjali puhul määratud. Lisaks peavad olema näidatud ka omadused, mida on võimalik lisaks määrata. Materjali omadusi redigeerimiseks tehakse väiksem taaskasutatav komponent *MaterialPropertyCard.tsx*. Komponentil on järgmised väljad:

- väärtus (*value*) – materjali omaduse arvuline väärtus,
- tõendatud (*verified*) – boolean väärtus,
- allikas (*source*) – tekstiväli.

Põhikomponendis on redigeeritava materjali puhul eksisteerivad omadused on salvestatud vormi oleku objektis *materjal.materialProperties* listi kujul. Iga omaduse jaoks luuakse iteratiivselt vastav komponent *MaterialPropertyCard*, millele antakse omaduse objekt ja funktsioon, mis töötleb sisendi andmete muutmist ja tagastab põhikomponendisse sisestatud väärtused. Põhikomponendis uuendatakse olekut ja vastavalt saadetakse ka päringuid materjali uuendamiseks andmebaasis. Komponendil *MaterialPropertyCard* välimuse seisukohalt on kaks olekut sõltuvalt sellest, kas

- vastav omadus on materjalile lisatud
- vastav omadus ei ole materjalile lisatud, kuid seda on võimalik lisada.

Materiali omaduste komponentide välimus mõlemas olekus on näidatud pildil 21.

The image shows a form for material properties. At the top, there are three dropdown menus: 'Nimetus' (Name) with 'Isover OL Top', 'Materjali tüüp' (Material type) with 'Soojusisolatsioonid', and 'Tootja' (Manufacturer) with 'Bauroc'. Below these, there are three main sections. The first section is for 'Tihedus  $\rho$  kg/m<sup>3</sup>' (Density) with a text input containing '35', a checked checkbox for 'tõendatud?' (verified), and a text input for 'Ehitusfüüsika ABC, T.Massc'. The second section is for 'Soojusjuhtivus' (Thermal conductivity) with a '+ Lisa' button. The third section is for 'Diffusioonitakistustegur' (Diffusion resistance) with a '+ Lisa' button.

Figure 21. *PropertyCard.tsx* komponendid materjali vormis

Infosüsteemi teine oluline ning keerulisema loogikaga osa on kalkulaator. Kalkulaator koosneb järgmistest plokkidest:

- arvutuse seadistuste osa, milles valitakse arvutuse parameetrid (sise- ja väliskeskkond, konstruktsiooni tüüp);
- konstruktsiooni kihtide plokk, milles lisatakse konstruktsioonile kihte ja valitakse kihtide omadused (materjal, paksus);
- tulemuste esitamise plokk, milles graafiliselt esitatakse konstruktsiooni kihtide skeematilist joonist, graafikuid ning arvuline tulemus (soojusjuhtivus  $U$  ja veeaurutakistus  $Z_p$ ).

Kalkulaatori põhiline *React* komponent on *Therm.tsx*. Komponent juhib ja korraldab alluvate komponentide tööd. Komponendis on mitu oleku (*state*) objekti:

- *optionsLibrary* – objekt, mis hoiab arvutuse parameetrite valikud (keskkonnad, konstruktsioonitüübid),
- *materials* – objekt, mis hoiab materjale, mida kasutatakse konstruktsiooni kihtide mudeldamisel,
- *calculationData* – objekt, mis hoiab arvutuse lähteandmeid, mida edastatakse serverile
- *calculationResult* – objekt, mis hoiab serverilt tulnud arvutuse tulemused.

Ülaltoodud objektide loomiseks kasutatakse sisseehitatud lahendust *useState*, mille abil loob React juhitavat oleku objekti. *useState* objektide eelis on oleku muutuste jälgimine ning kõikide olekust sõltuvate komponentide automaatne uuendamine oleku muutmisel.

Kalkulaatori põhikomponent koosneb väiksematest komponentidest, mis oma funktsionaalsusest lähtuvalt vastavad varem nimetatud kalkulaatori plokkidele. Kalkulaatori alamkomponendid on:

- *ThermOptions.tsx* – komponendile antakse arvutuse parameetrite andmeobjekt, arvutuse lähteandete objekt ning funktsioon kasutajasisendi töötlemiseks; sisendi muutmisel tagastab komponent põhikomponendisse sisendi uut väärtust, millega uuendatakse arvutuse lähteandmete objekti.
- *LayersBlock.tsx* – komponendile antakse edasi ehitusmaterialide andmeobjekt, mida kasutatakse konstruktsiooni kihtide lisamise vormis, ning funktsiooni konstruktsiooni kihtide salvestamiseks; komponent on kompleksse ülesehitusega ja omakorda omab struktuuris alamkomponente; komponent tagastab põhikomponendisse konstruktsiooni kihtide listi;
- *ResultBlock.tsx* – komponendile antakse kihtide plokkis koostatud kihtide objekt ning serverilt tulnud arvutuse tulemused graafikute joonestamiseks.

*Therm.tsx* komponendi struktuur on toodud Pildil 22.

Alamkomponentide loogika on piiratud oma vastutusalaga, kõik alamkomponendid on üksteisest sõltumatud ning on juhitud põhikomponendist. Serveriosaga suhtlemine toimub põhikomponendis oleva teenuse kaudu (*CalculationService*). Põhikomponent küsib teenuse kaudu serveriosalt andmeid, mis on tarvis kalkulaatoris olevate valiklistide täitmiseks ja komplekteerib kasutajas isendist andmeobjekti, mille struktuur vastab vastab serveriosa vastavale liidesele. Samuti komponent saadab serverile päringut arvutuse lähteandmetega ning töötleb ja edastab vastust tulemuste esitamise komponendile.

Tulemuste esitamise komponent *ResultBlock.tsx* tegeleb tulemuste representeerimisega



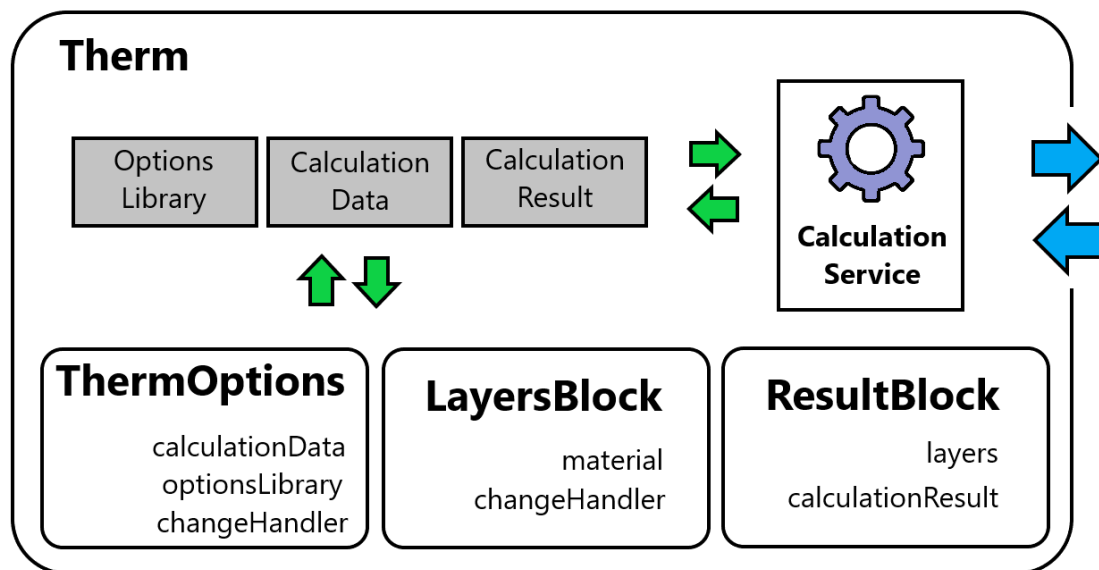


Figure 22. *Therm.tsx* komponendi struktuur

graafilisel viisil: joonestatakse lihtsustatud kujul mudeldatud konstruktsiooni skeemi, mille peale kantakse serverilt tulnud andmete põhjal temperatuuri ja veeaururõhkude graafiku jooned. Nõutud funktsionaalsus on otstarbekas lahendada HTML5 *canvas* elemendiga, mis on ette nähtud kahemõõtmeliste piltide genereerimiseks; element on manipuleeritav *JavaScript* keelega.

Konstruktsiooni skemaatilise joonise koostamiseks on lähteandmeteks konstruktsiooni kihtide objekt (list). Igas kihis sisaldub teave, mis on tarvis joonise koostamiseks: kihi paksus (väljendatud millimeetrites), kihi materjal koos materjali kategooriaga. Konstruktsiooni kihtide paksustest arvutatakse proportsionaalsed väärtused *canvas* elemendil kuvamiseks. Iga kihi lisamisega arvutatakse proportsioonid uuesti - kihtide arvu ja vastavalt konstruktsiooni kogupaksuse suurendamisega muutub mõõtkava väiksemaks. Kihtide värvilise täide tegemiseks võetakse värvi kood materjali kategoorias salvestatud koodist. Materjali kategooriate värvikode saab kasutaja vastaval rakenduse lehel redigeerida. ...

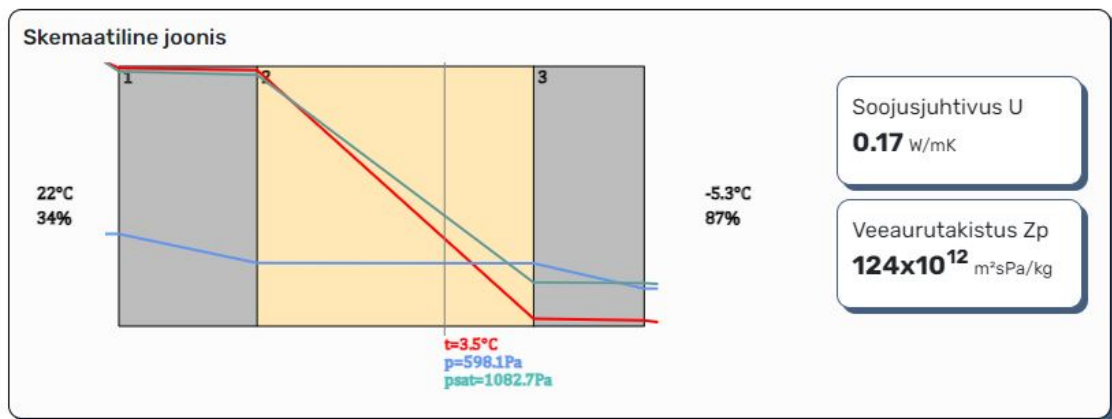


Figure 23. *ResultBlock.tsx* – tulemuste esitamise plokk

## **6. Testimine**

Testimisest..

## **7. Hinnang infosüsteemile**

Tagasiside arendatud süsteemile.

## **8. Kokkuvõte**

## Kasutatud kirjandus

- [1] T. Masso. *Ehitusfüüsika ABC*. Ehitame Kirjastus, 2012.
- [2] *Hygrothermal performance of building components and building elements Internal surface temperature to avoid critical surface humidity and interstitial condensation Calculation methods*. Standard. Estonian Centre for Standardisation, Jan. 2012.
- [3] *U-Bakus: U-Wert, Feuchteschutz, Hitzeschutz*. [Online; loetud 14. veebruar 2024]. URL: <https://www.ubakus.de/u-wert-rechner/>.
- [4] *GLASTA: condensation control | Physibel*. [Online; loetud 14. veebruar 2024]. URL: <https://www.physibel.be/en/products/glasta>.
- [5] *Delphin - Simulation program for the calculation of coupled heat, moisture, air, pollutant, and salt transport*. [Online; loetud 14. veebruar 2024]. URL: <https://bauklimatik-dresden.de/delphin/index.php?aLa=en>.
- [6] Codecademy Team. *What is REST*. [Online; loetud 10. veebruar 2024]. URL: <https://www.codecademy.com/article/what-is-rest>.
- [7] Codecademy Team. *What is a SPA?* [Online; loetud 10. veebruar 2024]. URL: <https://www.codecademy.com/article/fecp-what-is-a-spa>.
- [8] Krzysztof Kęsy. *What Is TypeScript? Pros and Cons of TypeScript vs. JavaScript*. [Online; loetud 11. veebruar 2024]. URL: <https://www.stxnext.com/blog/typescript-pros-cons-javascript/>.
- [9] 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<sup>1</sup>**

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.


13.03.2024

---

<sup>1</sup>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 loomingu tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtjaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

## Lisa 2 - Kasutajaliidese disain

Login

 **U-ARV** / ehitusfüüsika

Kalkulaatorid Konstruktsioonid Materjalid Parameetrid Kasutajad ➔


**Kasutaja**

**Parool**

Logi sisse


Figure 24. Sisselogimise vorm

Dashboard

 **U-ARV** / ehitusfüüsika


Kalkulaatorid Konstruktsioonid Materjalid Parameetrid Kasutajad ➔

**Calculator**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Calculator**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Calculator**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Calculator**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Calculator**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Calculator**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Tarindid**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Kasutajad**




Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Materjalid**



Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

**Parameetrid**



Dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s

Figure 25. Pealeht



Materjalid

U-ARV / ehitusfüüsika

Kalkulaatorid Konstruktsioonid Materjalid Parameetrid Kasutajad

Nimetus	Tüüp	Tootja	$\rho$ [kg/m <sup>3</sup> ]	$\lambda$ [W/mK]	$\mu$ [-]	Info
Betoon	Betoonid	-	2400	2	130	□
Aurutõkkekle	Membraanid	-	1390	0.17	100000	□
Bauroc Classic	Plokid	Bauroc	2400	2	130	□
Mineraalvill	Soojusisolatsioonid	-	25	0.035	1	□
OSB plaat	Puitplaadid	-	650	0.13	50	□
Kingspan Therma TW58	Soojusisolatsioonid	Kingspan	30	0.022	150	□
Keramiitbetoon	Betoonid	-	800	0.26	10	□
EPS 60 Fassaad	Soojusisolatsioonid	Reideni plaat	30	0.039	30	□
EPS 60 Silver Fassaad	Soojusisolatsioonid	Reideni plaat	30	0.032	30	□
Uus materjal	Muud materjalid	-	100	1	10	□
Uus materjal	Muud materjalid	-	100	1	10	□
Uus materjal	Muud materjalid	-	100	1	10	□
Uus materjal	Muud materjalid	-	100	1	10	□
Uus materjal	Muud materjalid	-	100	1	10	□
Uus materjal	Muud materjalid	-	100	1	10	□
Uus materjal	Muud materjalid	-	100	1	10	□

Figure 26. Materjalide nimekiri

Uus materjal

U-ARV / ehitusfüüsika

Kalkulaatorid Konstruktsioonid Materjalid Parameetrid Kasutajad

**Loo uus materjal**

**Nimetus**

**Kategooria**

**Tihedus  $\rho$  [kg/m<sup>3</sup>]**

**Soojusjuhtivus  $\lambda$  [W/mK]**

**Diffusiooni takistus  $\mu$  [-]**

☒ tõendatud

☐ tõendatud

☐ tõendatud

**allikas**

**allikas**

**allikas**

**Andmete allikas**

**Viide andmete allikale**

**Kommentaar**

Figure 27. Materjali lisamise vorm

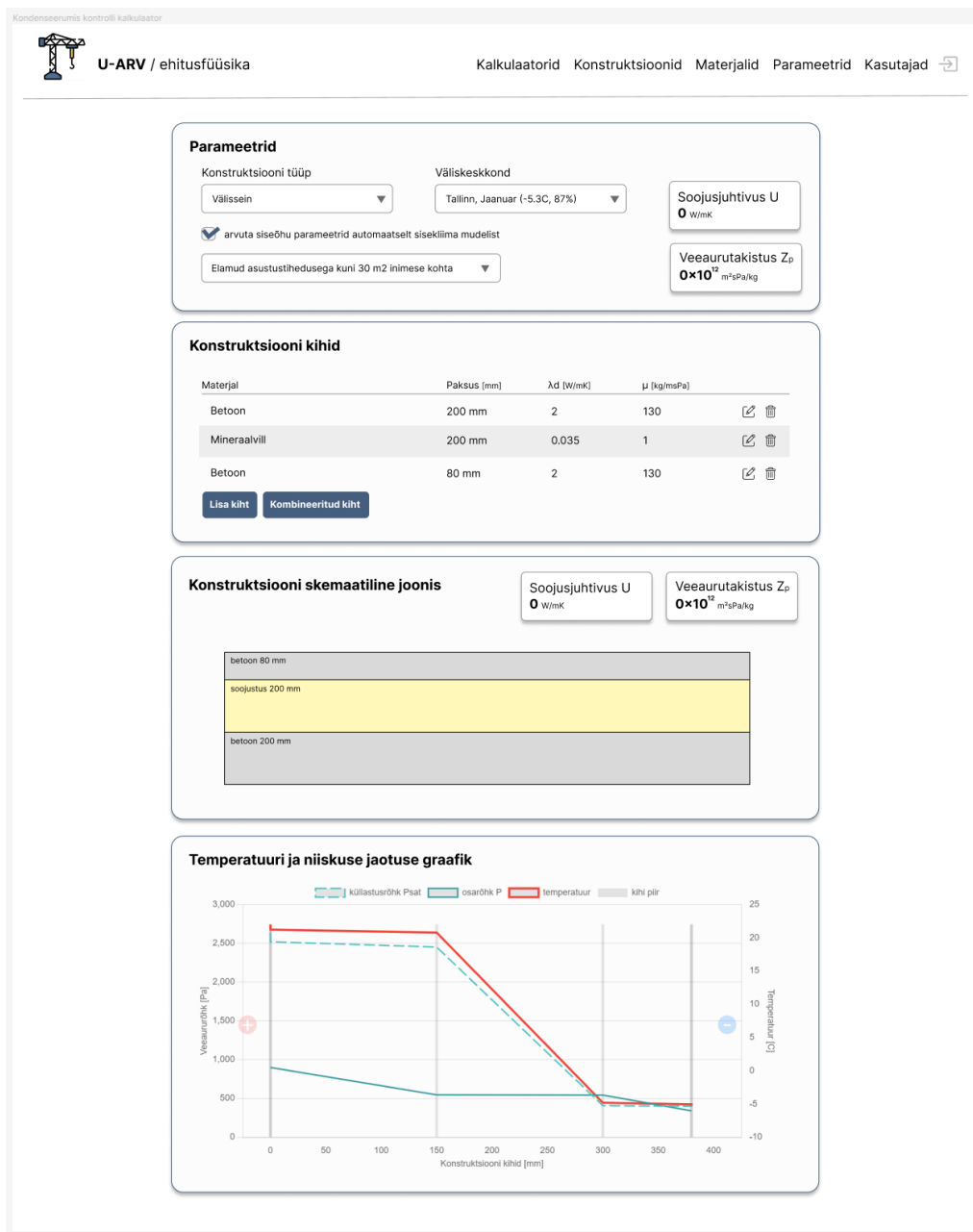


Figure 28. Kalkulaatori vaade