

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

19.03.2024

## **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

Demo-versioon	Tarkvara demonstratsiooniversioon
2D	Kahe-dimensionaalne graafika
Bootstrap	teek veebilehtede kujunduseks
BLL	Rakenduse äriloo­gi­ka­kiht ( <i>Business Logic Layer</i> )
CRUD	Tarkvara komponentide pattern "loo-loe-muuda-kustuta" ( <i>Create-Read-Update-Delete</i> )
CSS	Veebilehtede kujunduse keel ( <i>Cascade Style Sheet</i> )
DAL	Rakenduse andmete ligipääsemise kiht ( <i>Data Access Layer</i> )
Dashboard	Töölaud
Domain	TODO!!!
EFCore	.NET raamistiku osa ( <i>Entity Framework Core</i> )
GUID	Globaalne unikaalne identifikaator
HTML	Hüperteksti märgistuskeel ( <i>Hyper Text Markup Language</i> )
HTTP	Hüperteksti ülekandeprotokoll ( <i>Hypertext Transfer Protocol</i> )
JWT	JavaScripti veebitoken ( <i>JavaScript Web Token</i> )
JSX	React komponentides kasutatav JavaScripti laiendus ( <i>JSX</i> )
JavaScript	programmeerimiskeel
JSON	JavaScripti objekti andmevahetuse vorming ( <i>JavaScript Object Notation</i> )
MVC	Mudel-vaade-kontroller ( <i>Model View Controller</i> )
MVVM	TODO ( <i>Model-View-ViewModel</i> )
Miro	Interaktiivne keskkond diagrammide ja jooniste tegemiseks
Migration	Andmebaasi migratsioon
MVP	Minimaalne elujõuline toode ( <i>Minimum Viable Product</i> )
ORM	Objektsuhete kaardistamise tööriist ( <i>Object Relation Mapper</i> )
PDF	Digitaalne dokumendivorming ( <i>Portable Document Format</i> )
PHP	programmeerimiskeel
REST	veebiteenuste arhitektuurne stiil ( <i>Representational State Transfer</i> )
SFC	Ühefaililine komponent ( <i>Single File Component</i> )
SPA	Üheleheküljeline rakendus ( <i>Single Page Application</i> )

TypeScript	programmeerimiskeel, JavaScripti laiendus, mis teeb seda staatiliselt tüübitud keeleks ( <i>TypeScript</i> )
U-arv	meterjali või materjalidest koosneva konstruktsiooni soojusjuhtivus
UX/UI	Kasutajaliides/kasutajakogemus ( <i>User Experience/User Interface</i> )
Unit Of Work	Tööühik ( <i>Üksus Töös</i> )
front-end	veebirakenduse kasutajaliides
popup	Hüpikaken
reverse proxy	pööratud puhverserver
stateless	olekuta

# 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>45</b>
	<b>Kasutatud kirjandus</b>	<b>46</b>
	<b>Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks</b>	<b>47</b>
	<b>Lisa 2 – Kasutajaliidese disain</b>	<b>48</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	<i>LayersBlock.tsx</i> – konstruktsiooni kihtide modeleerimine . . . . .	40
24	<i>ResultBlock.tsx</i> – tulemuste esitamise plokk . . . . .	41
25	Kasutajaliidese disaain: sisselogimise vorm . . . . .	48
26	Kasutajaliidese disaain: pealeht . . . . .	48
27	Kasutajaliidese disaain: materjalide nimekiri . . . . .	49
28	Kasutajaliidese disaain: materjali lisamise vorm . . . . .	49
29	Kasutajaliidese disaain: kalkulaatori vaade . . . . .	50



## **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. 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 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 avaldada negatiivset mõju inimeste tervisele [1] ning seevastu õigesti projekteeritud hoone tagab kasutajale mugavusetunnet ja ka hoiab raha kokku minimeerides hoone kasutuskulusid [1]. 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 [2]. 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 ehitusfüüsikad meeles hoidma ka hoone haldamisega tegelevad inimesed.

Probleemi teine külg on ehitusvaldkonna madal digitaliseerimise tase [3] (ja konservatiivsus üldiselt). Viimastel aastatel on arendatud palju professionaalseid tarkvarasid projekteerimise ja ehitusjuhtimise tarbeks, kuid käesoleva töö peatükis 3 läbiviidud analüüsist järeldub, et ehitusfüüsika valdkonnas digitaalsete lahenduste arendamine oli tagasihoidlik. Euroopa turul on olemas mõned üksikud tooted (toodud peatükis 3), kuid need on üsna keerulised ja võrdlemisi ebamugava kasutajaliidesega – sellise tarkvara sihtgrupp on teadusvaldkond. Ehitusinseneride töö hõlmab palju erinevaid asju ning on tavaliselt ajaliselt piiratud, mistõttu keerulise kasutajaliidesega ja tööpõhimõttega tarkvara kasutamine ei ole alati sobiv variant.

Käesoleva töö eesmärk on välja töötada platvormi, mis sisaldaks erinevaid tööriistu, millega oleks võimalik lahendada ehitusfüüsika valdkonna ülesandeid mugavalt ja operatiivselt ehitusinseneridele piisaval tasemel ning tuginedes aktuaalsetele andmetele. See võiks parandada olukorda, kus ehitusfüüsika probleemidega tegelemine jääb erinevatel etappidel tegemata tarkvara puudumise või tarkvara kasutamiseks ebapiisavate oskuste tõttu. See oleks abivahendiks ehitusinseneridele, mis ei vaja sügavat valdkonna tundmist, et teostada piisavas mahu arvutusi tagamaks ehitusprojekti või ehituse kvaliteeti ehitusfüüsika seisukohalt.

Ehitusfüüsika valdkond on lai ning lahendusi on tarvis leida paljudele probleemidele – detailsem soojusjuhtivuse arvutus erinevatele hoone sõlmedele, energiamärgise või ventilatsiooni- ja õhuvahtusega 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, kui hinnakiri on avalikult kättesaadav. Lähtuvalt lahenduste analüüsi tulemustest defineeritakse konkreetsed nõuded kavandatavale infosüsteemile, millest lähtutakse infosüsteemi tehniliste lahenduste projekteerimisel. Lähtuvalt nõuetest valitakse infosüsteemi ehitamise tehnoloogiaid – kasutajaliides, serveriosa ja andmebaas. Tehnoloogiate all mõeldakse konkreetsed programmeerimiskeeled ja raamistikud. Iga infosüsteemiosa puhul tuuakse võrdlust ja põhjendatakse valikult. 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 detailsemad lahendused iga infosüsteemi osale eraldi (näiteks: serveriosa ja kasutajaliidese struktuur).

Arenduse protsessi planeerimisel kavandatav funktsionaalsus jagatakse kasutaja-lugudeks, mida grupeeritakse featuurideks (*feature*) ja eeposteks (*epic*). Kasutajalugudest moodustatakse tehnilised ülesanded, mida võetakse aluseks koodi kirjutamisel.

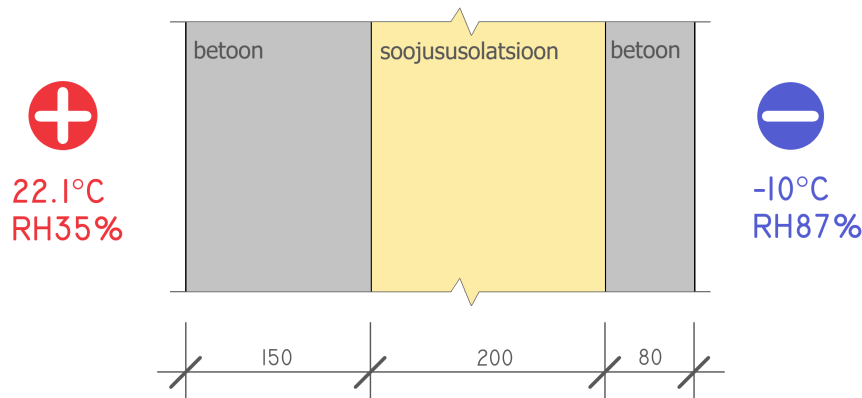
Töömahu piiramiseks valitakse planeeritud funktsionaalsusest minimaalse elujõulise toote funktsionaalsust (MVP) [4], mida viiakse ellu käesoleva lõputöö raames.

Kui MVP funktsionaalsus on realiseeritud, antakse toodet erinevates ametites töötavatele sihtgrupi esindajatele toote proovimiseks ja tagasiside saamiseks. Tagasisidele tuginedes antakse hinnangut tehtud tööle ja tehakse ettepanekuid toote edasiseks arenguks.

### 3. Probleemi olemus

#### 3.1 Probleemi uurimine

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



Joonis 1. Kihilise konstruktsiooni näide

Sellises olukorras kõige olulisemad materjalide omadused on soojuseri juhtivus  $\lambda$  [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 [5]. Infosüsteemi andmemudeli projekteerimisel tuleb kohe arvesse võtta 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. 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 ehitusfüüsika ülesandega, mille lehandamiseks peab ette võtma järgmiseid samme [6]:

- 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

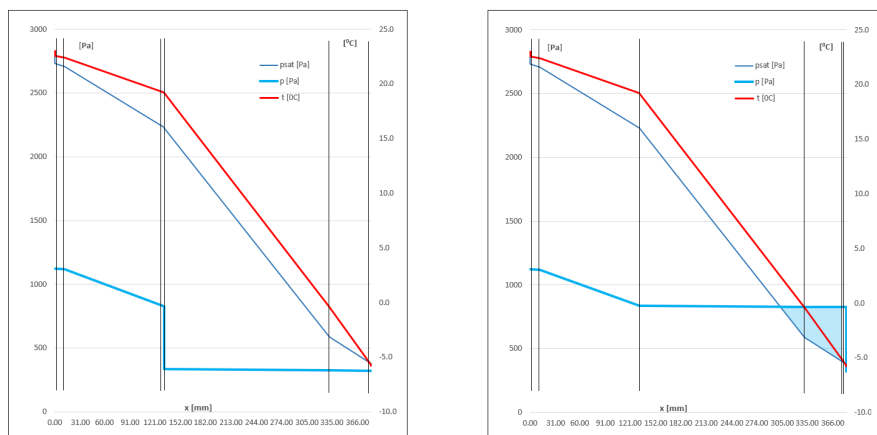
Lahendades ülesannet käsitsi, kasutatakse tabeli meetodit: koostatakse tabel, mille ridadele pannakse kirja konstruktsiooni kihid ja veergudesse arvutatakse samm sammult väärtused. Kuigi arvutused ei ole keerulised (valemid nimetatud arvutuste teostamiseks on leitavad viidetud allikatest [5][6][7], paraku käsitsi arvutamine võtab palju aega, kuna igasugune muudatus (kihi lisamine või järjekorra muutmine) tähendab kogu tabeli ümber arvutamist. Pildil 2 on toodud autori poolt koostatud arvutustabeli näide seitsmest kihist koosneva konstruktsiooni puhul.

Arvutustabel	Kihi paksus	Soojus-erijuhtivus	Kihi soojus-takistus	Temp. muutmine	Temp. kihi piiril	Küllastus-rõhk	Veeauru-erijuhtivus	Veeauru-takistus	Rõhkude erinevus	Veeauru osarõhk
	d [mm]	$\lambda_d$ [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
Välisõhk					-5.8	375				322.1
Kokku			7.5					5.228E+14		

Joonis 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 visuaalset ülevaadet 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).

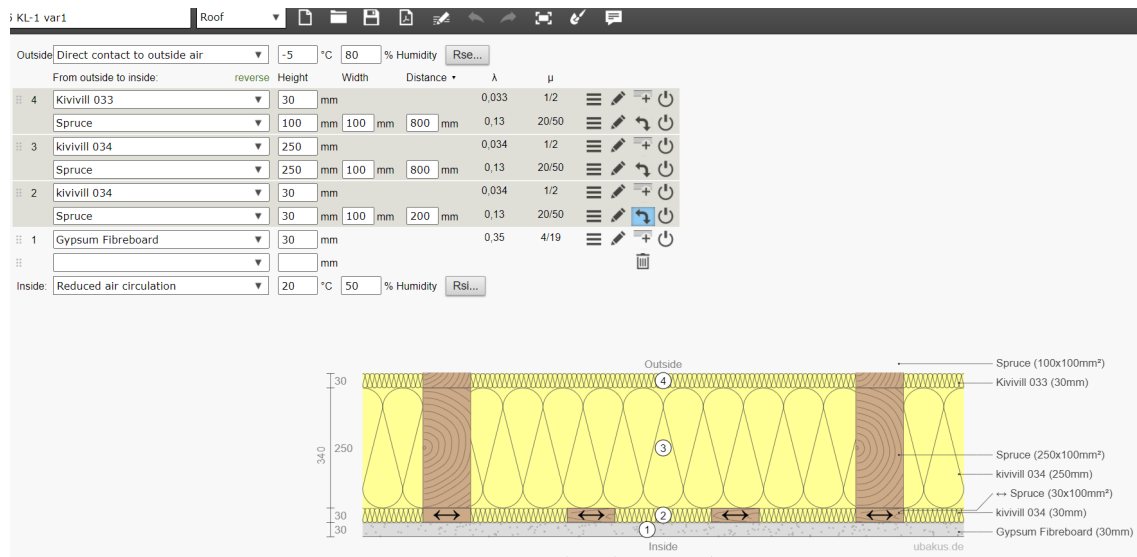


Joonis 3. Näide tulemuste esitamisest graafikul

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 veebirakenduse kujul. Tarvara *demo*-versioon on saadaval tasuta.

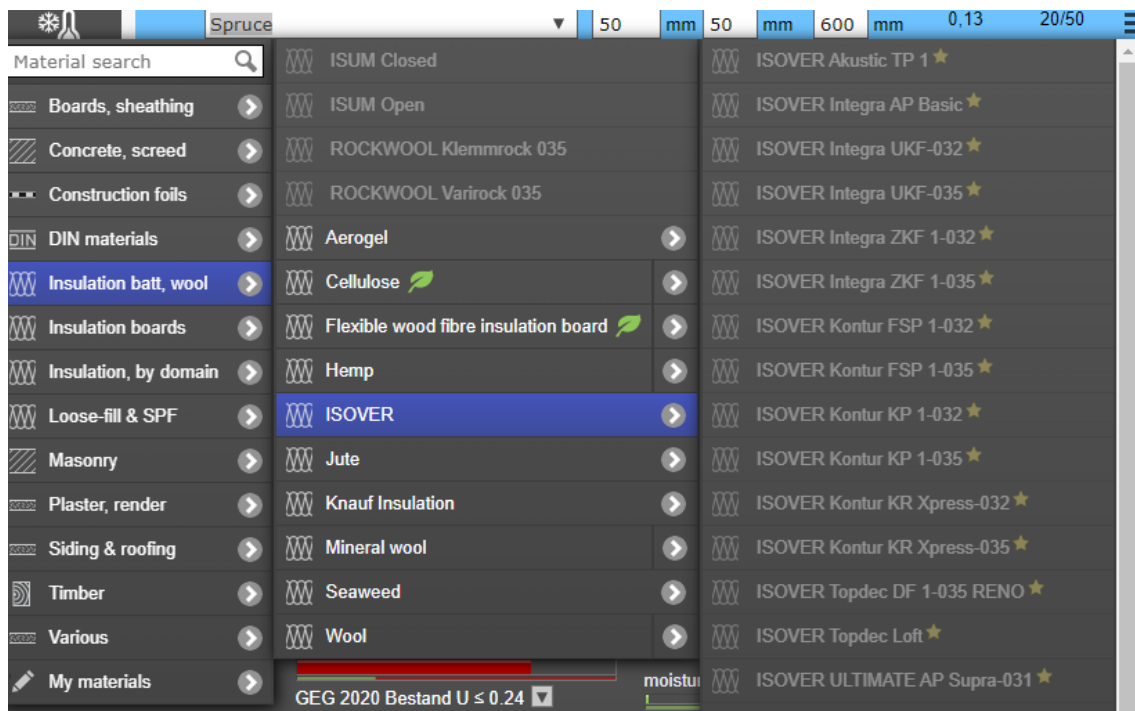


**Ubakus** võimaldab teostada konstruktsiooni niiskustehnilist analüüsi. Kasutajaliides võimaldab mudeldada mitmest kihist koosneva konstruktsiooni, valides igale kihile paksust ja materjali, millest kiht koosneb. Tugev eelis on see, et tarkvaraga saab analüüsida ka mittehomogeensete (mitemest erinevast materjalist, nt puitsõrestiksein) kihtidega konstruktsioone – Joonis 4 [8].

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) – Joonis 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.

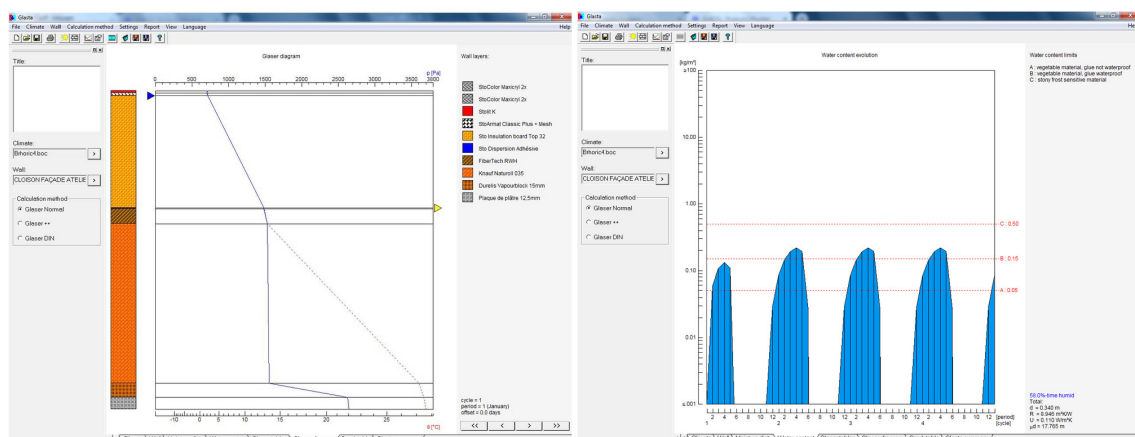
Keskkonnatingimused valitakse manuaalselt sisestades õhutemperatuuri ja -niiskuse väärtused. Rakendus võimaldab arvutuste tulemust vaadata erineval viisil, alates lihtsamast 3D visualiseerimist kuni värvilise temperatuurikaardini. Tarkavara saab osa aastase tellimusega, mille maksumus on alates 50 kuni 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, tarkvara on suunatud eelkõige Kesk-Euroopa ja Kanada turgudele, Balti ja Skandinaavia riikidele lokaliseerimine puudub. Kokkuvõttes antud lahendust võib kindlasti võtta arvesse toote funktsionaalsuse kavandamisel.





Joonis 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 (Joonis 6) [9].

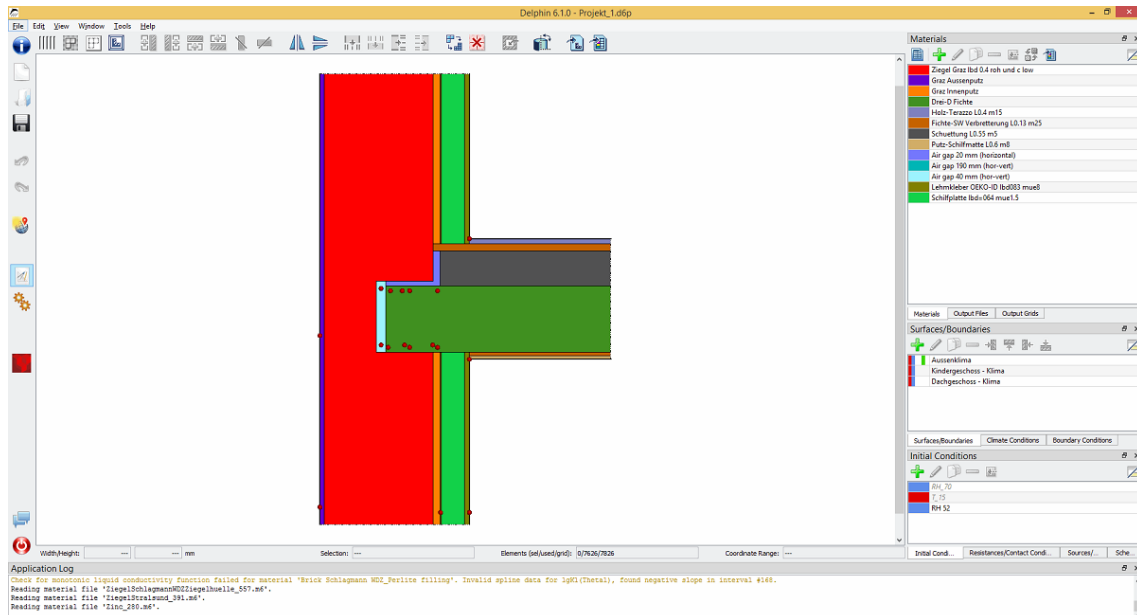


Joonis 6. Glasta: kasutajaliides, ekraanitõmmis.

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 graafikult (Joonis 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 [10]. 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 ilmingimuste (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).



Joonis 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 väliste tingimuste parameetrid
- valida sisemiste tingimuste 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

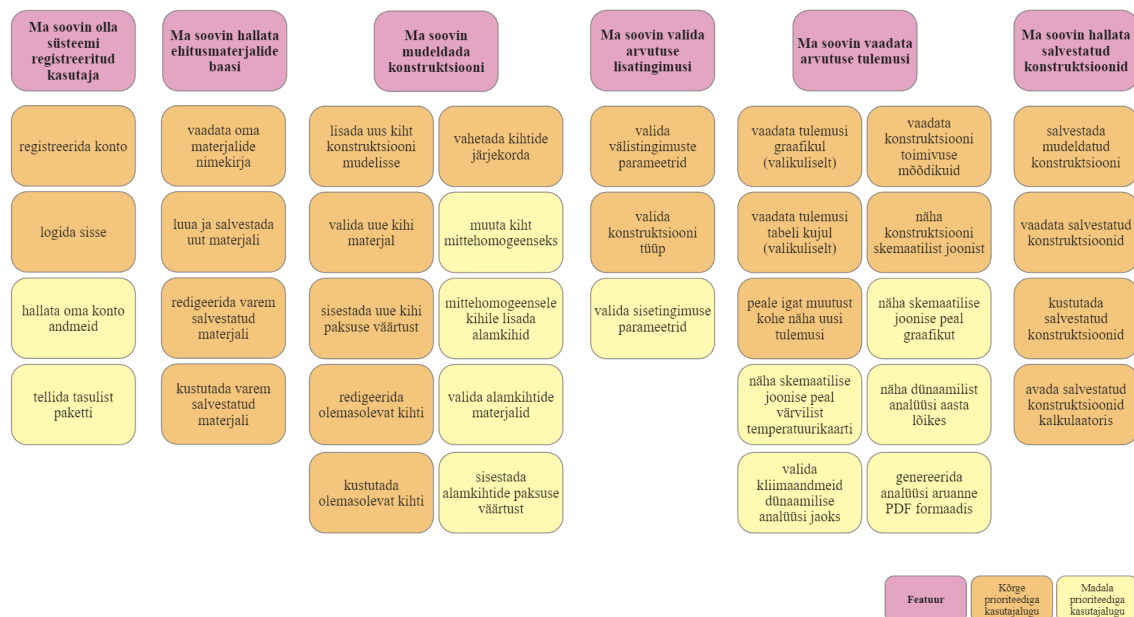
Funktsionaalsest nõuetest on kokku pandud kasutajalood, mis omakorda jagatud featurideks. Protsessi visualiseerimiseks on kasutatud Miro interaktiivne keskkond. Kliendi kasutajalood on jagatud kuueks featuuriks (Joonis 8):

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

Administraatori kasutajalood on jagatud kolmeks featuuriks (Joonis 9):

- infosüsteemi 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 mittehomogeensete kihtidega konstruktsioonide arvutus. Andmemudelid kohe tuleb projekteerida nii, et tulevikus see oleks võimalik ellu viia ilma suurte muutusteta, kuid arvutuste ja kasutajaliidese lihtsustamise mõttes jääb see osa esimese iteratsiooni skoobist välja.

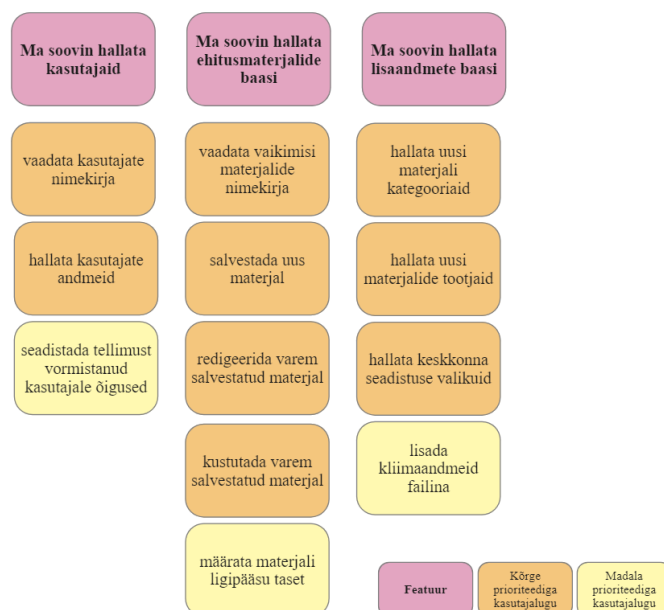


Joonis 8. Kliendi kasutajalood

## 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-



Joonis 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 printsiipidele.

## 4.2 Tehnoloogiate valik

Tehnoloogiate valimisel lähtutakse kaasaegsetest veebirakendamise ehitamise printsiipidest, 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 lisada 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 teksti kujul, mida on võimalik saata HTTP päringutega üle veebi[11]. 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)[11].

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 teostatakse 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 [12]. See on ka oluline, kuna tulemuste esitamist peab uuendama dünaamiliselt kohe peale arvutuse lähteandmete muutmist.

Üheleherakenduste ehitamiseks kasutatakse JavaScript programmeerimiskeelt veebilehe dünaamilise loogika juhtimiseks. Soovitavalt kasutada JavaScript keele laiendust – TypeScript, mis muudab JavaScript-i tugevalt ja staatiliselt tüübitud keeleks [13]. 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 loogikat ellu viia kasutades puhta 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 töötlemiseks kasutatakse *Axios* – teek, mida saab kasutada erinevate raamistikutega.

Üheleherakenduse ehitamiseks 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 sisaldab Vue-spetsiifilist süntaksi, mille abil juhib raamistik lehe logikat. Vue rakendus koosneb SFC komponentidest, igas komponendis on eraldi defineeritud 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 realiseerida 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 laeindusi, 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 realiseerida 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 kasutatakse 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 vaikinisi, 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 programmeerijalt ja ka rakenduse paigaldamine serverile on erinevalt PHP-st ka keerulisem, kuna projekti ehitamine eeldab palju lisategevusi. Java on kasutusel väga suure kasutajate hulgaga infosüsteemides (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 pakettide paigaldamisega (nt EntityFrameworkCore - ORM raamistik,.AspNetCore.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 (sisse ehitatud 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 ressursil 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 infosüsteemi äriloomika 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

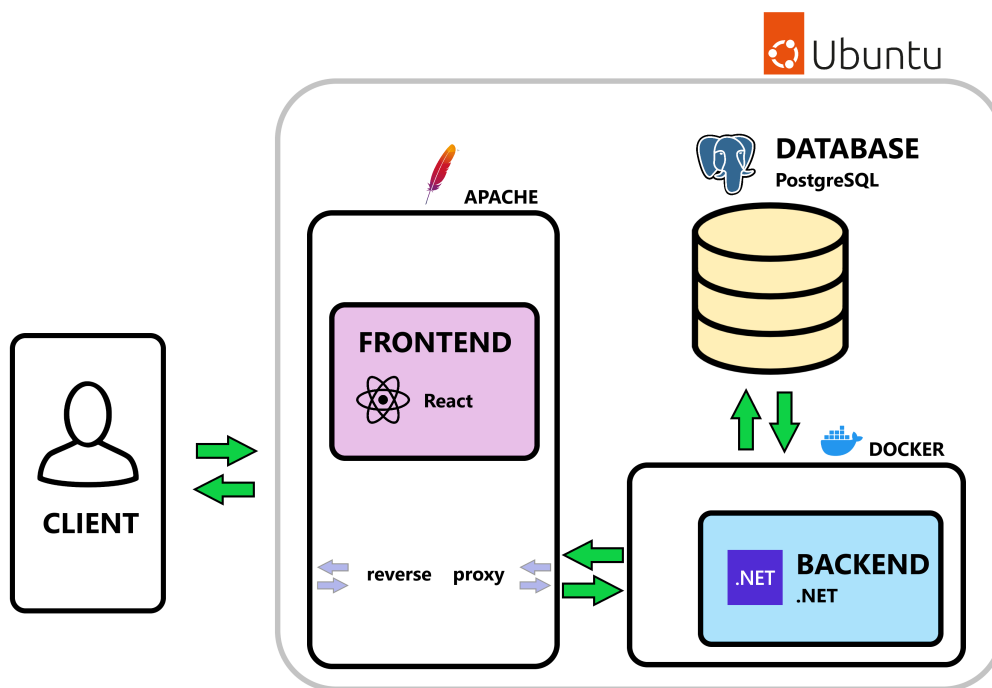
Kõikidele ülaltoodud süsteemidele eksisteerivad juhtprogrammid .NET EF Core ühendamiseks, seetõttu võib kõik toodud lahendused pidada sobilikuks. Valikul lähtutakse sellest, mis tehnoloogiaga on programmeerijal rohkem teadmisi ja kogemusi. Antud juhul see on PostgreSQL.

## 4.3 Veebirakenduse arhitektuur

Kavandatava infosüsteemi komponendid on järgmised:

- **server** – renditud pilveserver 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 Joonisel 10. Pilveserveri teenuse rentimise valikult 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.



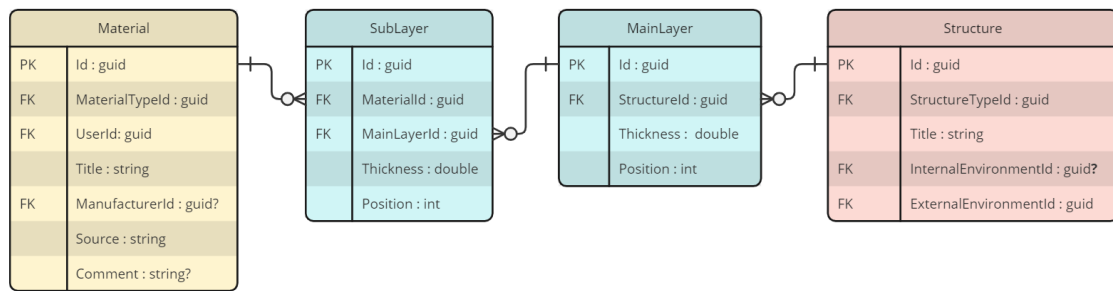
Joonis 10. Infosüsteemi arhitektuur

#### 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 [14]. 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.

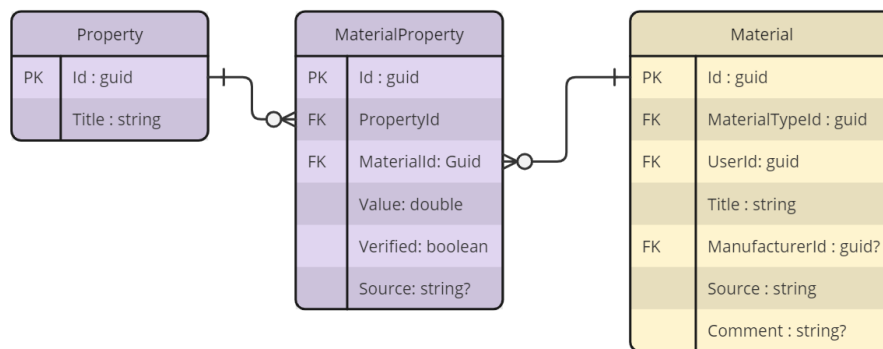
Kuigi minimaalse elujõulise toote arendamise raames on otsustatud arvutusprogrammist elimineerida mittehomoogeensete (mitmest materjalist koosnevate) kihtide arvutamist, siiski tuleb sellise võimalusega arvestama tulevikus. Selle jaoks on andmemudelis olemas alamkihi objekt (*SubLayer*). Iga konstruktsiooni 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

Joonisel 11.



Joonis 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, andmebaasimodelis on olemas järgmised objektid: omadus (*Property*) ja materjali omadus (*MaterialProperty*) - Joonis 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).



Joonis 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:

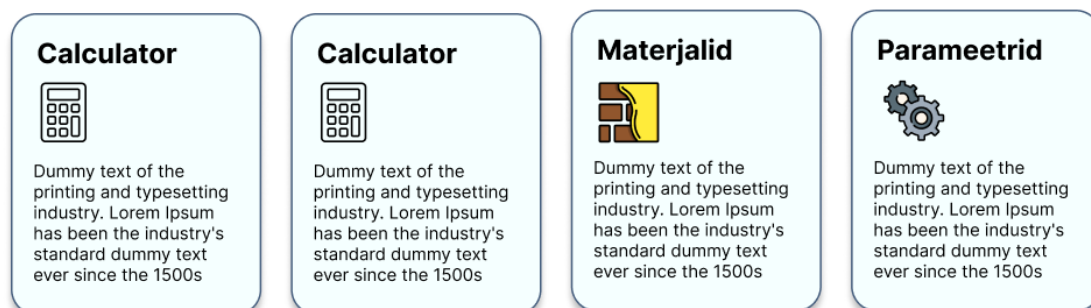
- Menüüriba
- Registreerimise ja sisselogimise vaated
- *Dashboard*-vaade
- Kasutaja andmete vaade
- Ehitusmaterjalide nimekirja vaade
- Ehitusmaterjali loomise ja redigeerimise vaade
- Kalkulaatori vaade
- Kasutajate nimekiri (administraator)
- Parameetrite ja lisaandmete vaade (administraator)

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.

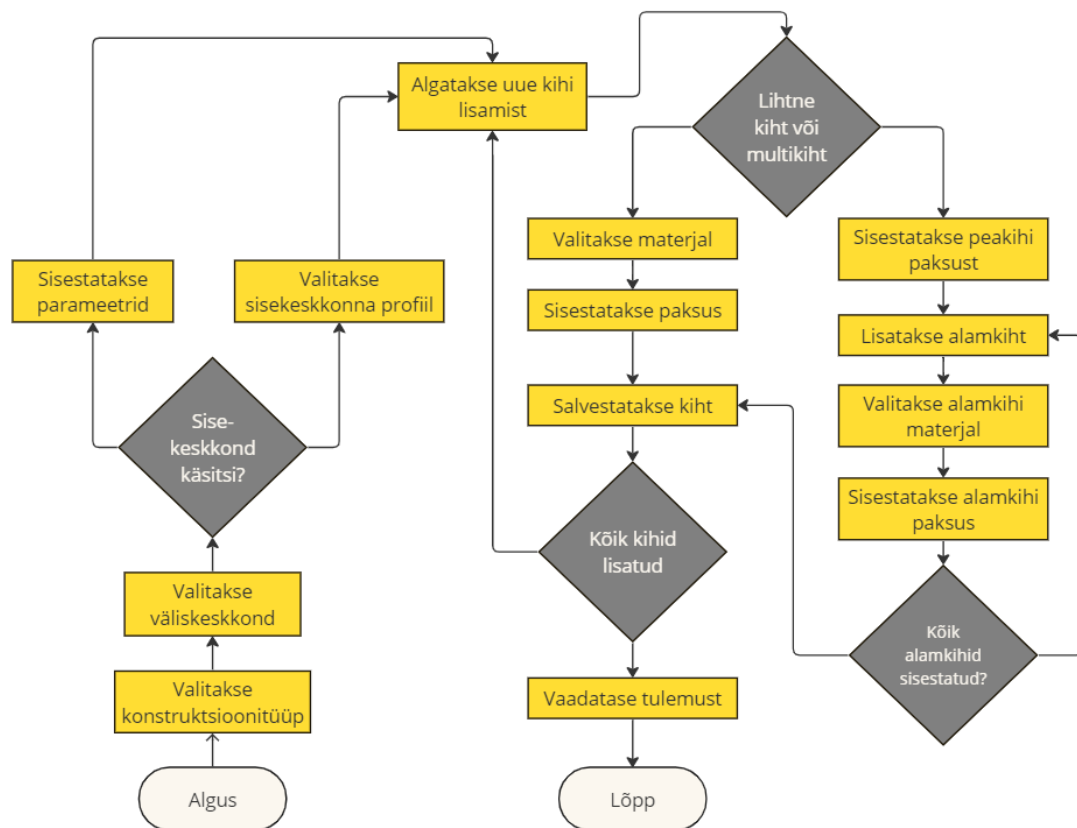
Pealehel (*Dashboard*) näidatakse kaardid, mille kaudu kasutajat juhitakse tööriistadele ja moodulitele, millele on kasutajal ligipääs. Administraatori ja tavakasutaja *Dashboard*-vaated on vastavalt erinevad. *Dashboard*-kaartide näide on toodud Joonisel 13.



Joonis 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 Joonisel 14.



Joonis 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 plokis on esitatud ka arvutuse üldised arvulised tulemused - konstruktsiooni summaarne soojusjuhtivus (U-arv) ja veeaurutakistus. Parameetrite plokki disain on näidatud Joonisel 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

### Parameetrid

Konstruksiooni tüüp

Välissein

Väliskeskond

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

Joonis 15. Parameetrite plokki disain

nupp on deaktiveeritud, et suunata kasutajad tegelema vaid ühe kihiga korraga. Lihtsa (ühest materjalist koosneva) kihi lisamise näide on toodud Joonisel 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

Joonis 16. Ühest materjalist koosneva kihi lisamine

Mittehomogeense (mitmest erinevast materjalist koosneva) kihi lisamine eeldab keerulist 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 suunas. Kui kõik andmed on valitud, siis uus kiht salvestatakse vastava nuppu vajutamisega. Mitmest materjalist koosneva kihi lisamise näide on toodud Joonisel 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 veeauru osarõhkude jaotuse graafikud, kuid prototüübi puhul antud koht on lihtsustatud ja graafikud on toodud eraldi plokis. Graafikute ehitamiseks kasutatakse valmislahendust, nt Javascript-i põhine teek Graph.js. Tulemuste plokkide disain on näidatud Joonisel 18 ja Joonisel 19..



### 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> <div> <div>250 mm</div> <div> <div>Puit</div> <div>Mineraalvill</div> </div> <div> <div>50</div> <div>550</div> </div> </div> <div> <div>Salvesta</div> </div>				
<div> <div>Lisa kiht</div> <div>Kombineeritud kiht</div> </div>				

Joonis 17. Mitmest materjalist koosneva kihi lisamine

### Konstruksiooni skemaatiline joonis

Soojusjuhtivus U

0 W/mK

Veeaurakistus  $Z_p$

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

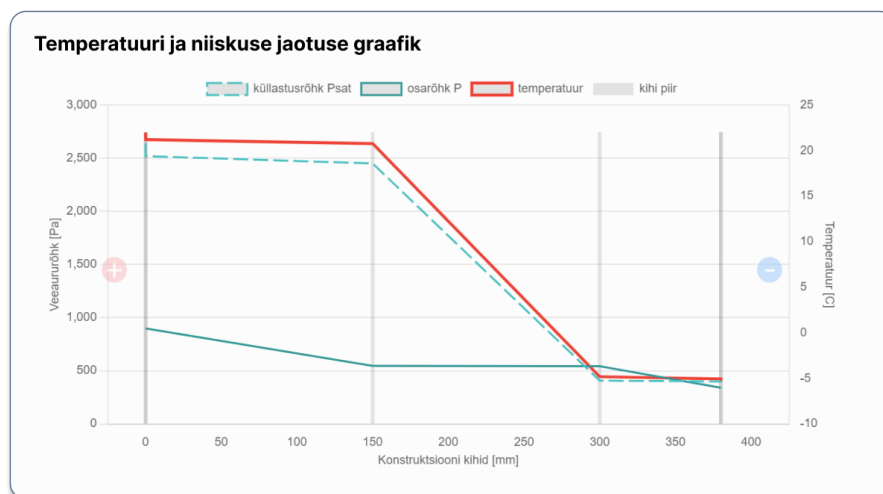
betoon 80 mm

soojustus 200 mm

betoon 200 mm

Joonis 18. Konstruksiooni skemaatiline joonis

Ülejäänud lehed – ehitusmaterjalid, parameetrid, kasutajad – kujutavad ennast võrdlemisi lihtsaid CRUD-tüüpi kasutajaliideseid, millistel keeruline loogika puudub. Kõikide lehtede disaini projektid on esitatud käesoleva töö **Lisas X**.



Joonis 19. Tulemuste esitamine diagrammil

## 5. Veebirakenduse arendus

Töö käesolevas osas käsitletakse infosüsteemi arendusega seotud aspekte. Alampeatükides on toodud üldine info ja põhimõtted infosüsteemi vastava osa realiseerimisest. Samuti mõnede kriitilisemate elementide puhul on toodud ka lahenduse detailne kirjeldus ja lahenduse valiku põhjendus. Koodinäited ei ole suure mahu tõttu käesolevas töös toodud. Koodibaasiga saab tutvuda projekti GitHub repositooriumist <https://github.com/gildix/ehitusandmebaas>

### 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 iganädalased varukoopiad, mida salvestatakse serveril. Selle jaoks luuakse skripti ning määratakse Ubuntu Cron regulaarse tööna.

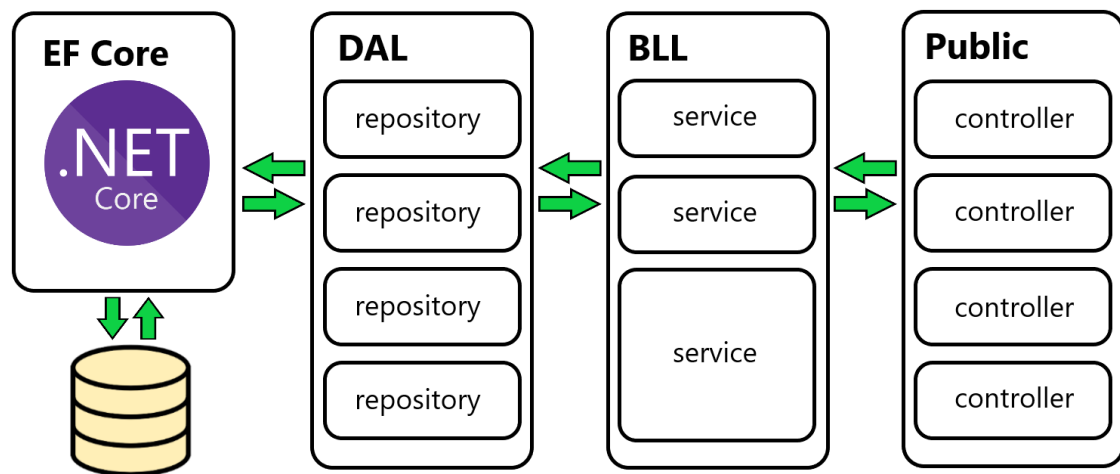
### 5.2 Serveriosa

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

- andmebaasi ja domeeni kiht,
- andmete ligipääsu kiht,
- ärilooika 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 ärilooika kihis. Analoogselt

ka andmete ligipääsu DAL kihis teostatakse ainult andmete küsimist ja salvestamist andmebaasile (antud töö raames – raamistiku kaudu), võimalikult välistades igasugused infosüsteemi äriloogikaga seotud tegevused. Kihid on näidatu Joonisel 20.



Joonis 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 – repositooriumideks (*Repository*), mis on baasimplementatsioonid kujutavad ennast klassikalist CRUD-tüüpi repositooriumit. Samuti repositooriumis tehakse andmete 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, sisemiste ja välimiste tingimuste 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 püütakse seda *try-catch* ploki 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, suurem 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.

Materjali omaduste komponentide välimus mõlemas olekus on näidatud Joonisel 21.

The image shows a form for entering material properties. It includes three dropdown menus at the top: 'Nimetus' (Name) with 'Isover OL Top', 'Materjali tuup' (Material type) with 'Soojusisolatsioonid', and 'Tootja' (Manufacturer) with 'Bauroc'. Below these are three main sections: 'Tihedus  $\rho$  kg/m<sup>3</sup>' with a value of 35 and a 'tõendatud?' checkbox, 'Soojusjuhtivus' (Thermal conductivity) with a '+ Lisa' button, and 'Diffusioonitakistustegur' (Diffusion resistance) with a '+ Lisa' button. A text field at the bottom left contains 'Ehitusfüsika ABC, T.Massc'.

Joonis 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 (sisemine ja välimine keskkonnad, 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:

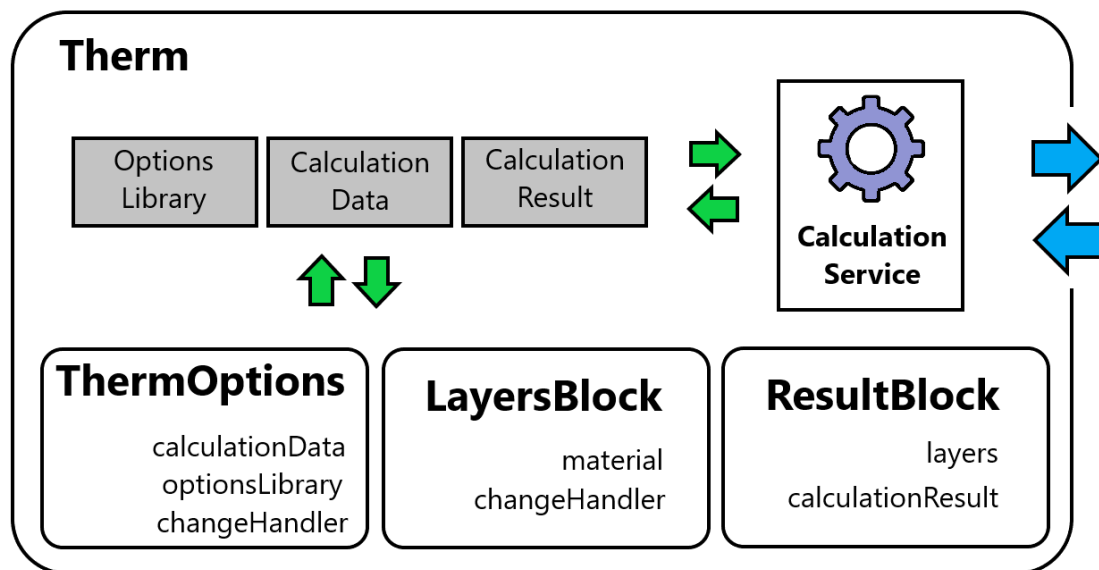
- *ThermOptions.tsx* – komponendile antakse arvutuse parameetrite andmeobjekt, arvutuse lähteandete objekt ning funktsioon kasutajasisendi töötlemiseks; sisendi muutmisel tagastab komponent põhikomponendisse sisendi uut väärtust, millega uuendatakse arvutuse lähteandmete objekti.
- *LayersBlock.tsx* – 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 plokis koostatud kihtide objekt ning serverilt tulnud arvutuse tulemused graafikute joonestamiseks.

*Therm.tsx* komponendi struktuur on toodud Joonisel 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 serveriosa vastavale liidesele. Samuti komponent saadab serverile päringut arvutuse lähteandmetega ning töötleb ja edastab vastust tulemuste esitamise komponendile.

Konstruktsiooni kihtide modelleerimisega tegeleb kihtide plokk *LayersBlock.tsx* (Joonis





Joonis 22. *Therm.tsx* komponendi struktuur

23). Komponent kujutab ennast konstruktsiooni kihtide nimekirja koos uue kihi lisamise ja olemasolevate kihtide redigeerimise vormidega. Rakenduse äriloojika eeldab, et kasutajal peab olema võimalus kihtide järjekorda muuta, seetõttu on nimekiri tehtud dünaamilise nimekirjana, mille puhul saab kasutaja elementide asukohta muuta. Igal nimekirja real on ees ikoon, mille haarates saab kihti tõmmata õigesse kohta.

**Konstruktsiooni kihid**

Materjal	Paksus [mm]	$\lambda_d$ [W/mK]	$\mu$ [kg/msPa]	
↕ Beton	150 mm	2	130	
↕ Mineraalvill	150 mm	0.035	1	
↕ Beton	80 mm	2	130	

- Vali materjal
0

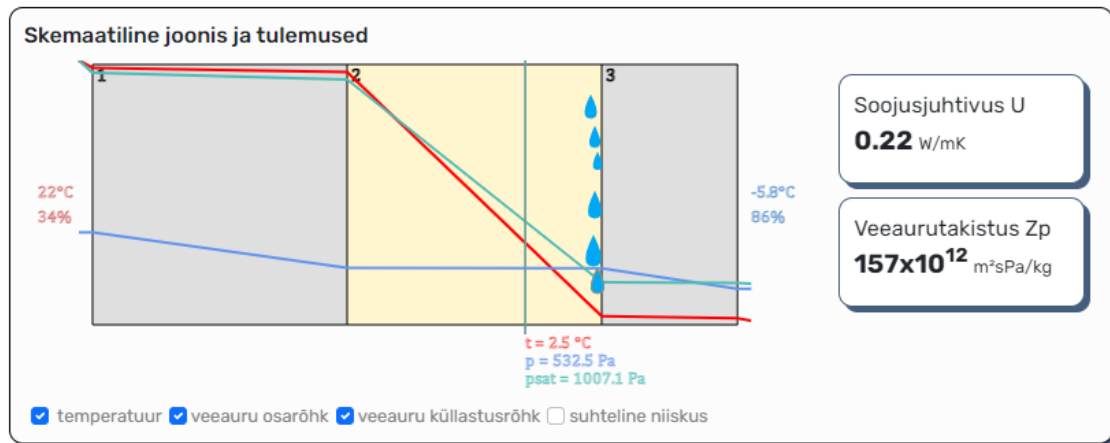
Salvesta

Lisa kiht

Joonis 23. *LayersBlock.tsx* – konstruktsiooni kihtide modelleerimine

Tulemuste esitamise komponent *ResultBlock.tsx* tegeleb tulemuste representeerimisega 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.

Konstruktiooni skemaatilise joonise koostamiseks on lähteandmeteks konstruktiooni kihtide objekt (list). Igas kihis sisaldub teave, mis on tarvis joonise koostamiseks: kihi paksus (väljendatud millimeetrites), kihi materjal koos materjali kategooriaga. Konstruktiooni 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 värvikoode saab kasutaja vastaval rakenduse lehel redigeerida. ...



Joonis 24. *ResultBlock.tsx* – tulemuste esitamise plokk

## 6. Testimine

Infosüsteemi serveriosa kõige kriitilsema funktsionaalsuse testitakse automaattestimisega. Kõigepealt peab automaatteste luua materjalide lisamise testimiseks. Testimisega kontrollitakse järgmised materjalide (ja nende omaduste) lisamisega ja turvalisusega seotud väited:

- kasutaja A näeb avalikke materjale
- kasutaja A saab luua uus materjal
- kasutaja B ei näi kasutaja A poolt loodud materjale
- sisse logimata kasutaja näev avalikud materjalid
- sisse logimata kasutaja ei näe kasutaja A ja kasutaja B poolt lisatud materjalid
- kasutaja saab lisada materjalile omadust
- kasutaja saab redigeerida materjali omadust
- kasutaja näeb omadusi, mis on materjali puhul olemas

Samamoodi testitakse ka arvutusteenust. Arvutusteenuse testimisel peab kontrollima järgmiseid väiteid:

- arvutuse lähteandmetes näeb kasutaja tema poolt sisestatud materjalid
- arvutuse lähteandmetes näeb kasutaja vaid need materjalid, millistele on lisatud arvutuse teostamiseks nõutud omadused
- teenus tagastab õiged arvutuse tulemused

Samuti kontrollitakse, et ebaõnnestunud päringutele tagastab alati teenus adekvaatset ja informatiivset vastust (nt ligipääsu puudumine, vigased päringuga saadetud andmed).

Kasutajaliidese testimist viiakse läbi manuaalse testimisena. Testimise aluseks on peatükis 4 toodud kasutajalugud.

## 7. Hinnang infosüsteemile

Hinnang arendatud infosüsteemile oli küsitud sihtgrupi esindajatelt. Sihtgrupi erinevatest valdkondadest (arhitektuurne projekteerimine, konstruktiivne projekteerimine, ehitusfirma kvaliteedijuhtimine, ehitusprotsessi juhtimine, omaniku järelevalve teostaja) olid leitud esindajad, kes oli valmis infosüsteemi läbiproovida ja tagasisidet anda.

Hinnangu moodustamiseks oli korraldatud minimaalse väärtusliku funktsionaalsuse proovimine ja tagasiside korjamine. Proovijatele oli saadetud link veebirakendusele, registreerimise võti ja lihtne stsenaarium, mida võiks katsuda rakenduses läbi mängida:

- registreerida kasutajaks kasutades saadetud registreerimise võtit
- logida sisse
- luua ehitusmaterjal ja lisada sellele arvutuseks vajalikud omadused (vihje: Puit, tihedus: 800g/m<sup>3</sup>, soojuserijuhtivus: 0.3 W/mK, difusiooni takistustegur 20)
- avada soojusjuhtivuse kalkulaator ja modelleerida konstruktsiooni, mis sisaldab sisestatud materjali
- proovida modelleerida konstruktsiooni, mis koosneb kolmest kihist (betoon 100 mm, mineraalvill 150 mm, betoon 80 mm), uurida kas esineb kondensaadi tekkimise tõenäosus ja millesel aastaajal

Toodud stsenaariumi alusel oli pakutud anda hinnangut järgmistele aspektidele:

1. kasutajaliidese üldine kasutusmugavus
  - (a) stsenaariumi läbimine läks sujuvalt
  - (b) vajalike kasutajaliidese elementide leidmisega (nupud, väljad, lehed) ei olnud raskusi
  - (c) veebirakenduse käitumine oli ootuspärane
2. veebirakenduse äriloogika
  - (a) ehitusmaterjali lisamise vorm võimaldab salvestada kõiki vajalikke (käesoleva süsteemi kontekstis) andmeid materjalist
  - (b) ehitusmaterjalide nimekiri on ülevaatlik, sellest on näha kõiki salvestatud andmeid
  - (c) kalkulaatori lehel arvutuse teostamiseks tegevuste järjekord on intuiitselt arusaadav
  - (d) arvutuse tulemuste esitamine kalkulaatori lehel on piisavalt selge ja arusaadav

### 3. üldine kontseptsioon

- (a) arendatud infosüsteem leiab teie töös kasutust
- (b) näete rakenduse ideel edasise arengu potentsiaali

Kõik aspektid hinnati Likerti neljaarvulise skaalaga, valides järgmistest hinnetest: "ei ole nõus", "pigem ei ole nõus", "pigem nõus" ja "nõus". Samuti küisiti ettepanekuid veebirakenduse edasise arengu vajadusest ja suunast.

## **8. Kokkuvõte**

## Kasutatud kirjandus

- [1] Roland Rokka. “Jäiga plastvahust soojustusega puitsõrestiktartarindite soojus- ja niiskustehniline toimivus. Magistritöö”. [Online; loetud 28. veebruar 2024]. 2020.
- [2] *Hoone energiatõhususe miinimumnõuded. Ettevõtlus- ja infotehnoloogiaministri määrus nr 63, vastu võetud 11.12.2018.* [Online; loetud 27. veebruar 2024]. 2018. URL: <https://www.riigiteataja.ee/akt/113122018014>.
- [3] *Global industry digitisation index, McKinsey Global Institute.* [Online; 17. 2024]. URL: <https://www.ubakus.de/u-wert-rechner/>.
- [4] Eric Nies. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful.* Crown Currency, 2011.
- [5] Tiit Masso. *Ehitusfüüsika ABC.* Ehitame-Kirjastus, 2012.
- [6] *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.
- [7] Jaan Rohusaar et al. *Ehituskonstruktori käsiraamat.* Ehitame-Kirjastus, 2010.
- [8] *U-Bakus: U-Wert, Feuchteschutz, Hitzeschutz.* [Online; loetud 14. veebruar 2024]. URL: <https://www.ubakus.de/u-wert-rechner/>.
- [9] *GLASTA: condensation control | Physibel.* [Online; loetud 14. veebruar 2024]. URL: <https://www.physibel.be/en/products/glasta>.
- [10] *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>.
- [11] Codecademy Team. *What is REST.* [Online; loetud 10. veebruar 2024]. URL: <https://www.codecademy.com/article/what-is-rest>.
- [12] Codecademy Team. *What is a SPA?* [Online; loetud 10. veebruar 2024]. URL: <https://www.codecademy.com/article/fecp-what-is-a-spa>.
- [13] 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/>.
- [14] 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.

19.03.2024


---

<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


Joonis 25. 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

Joonis 26. 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	

Joonis 27. 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**

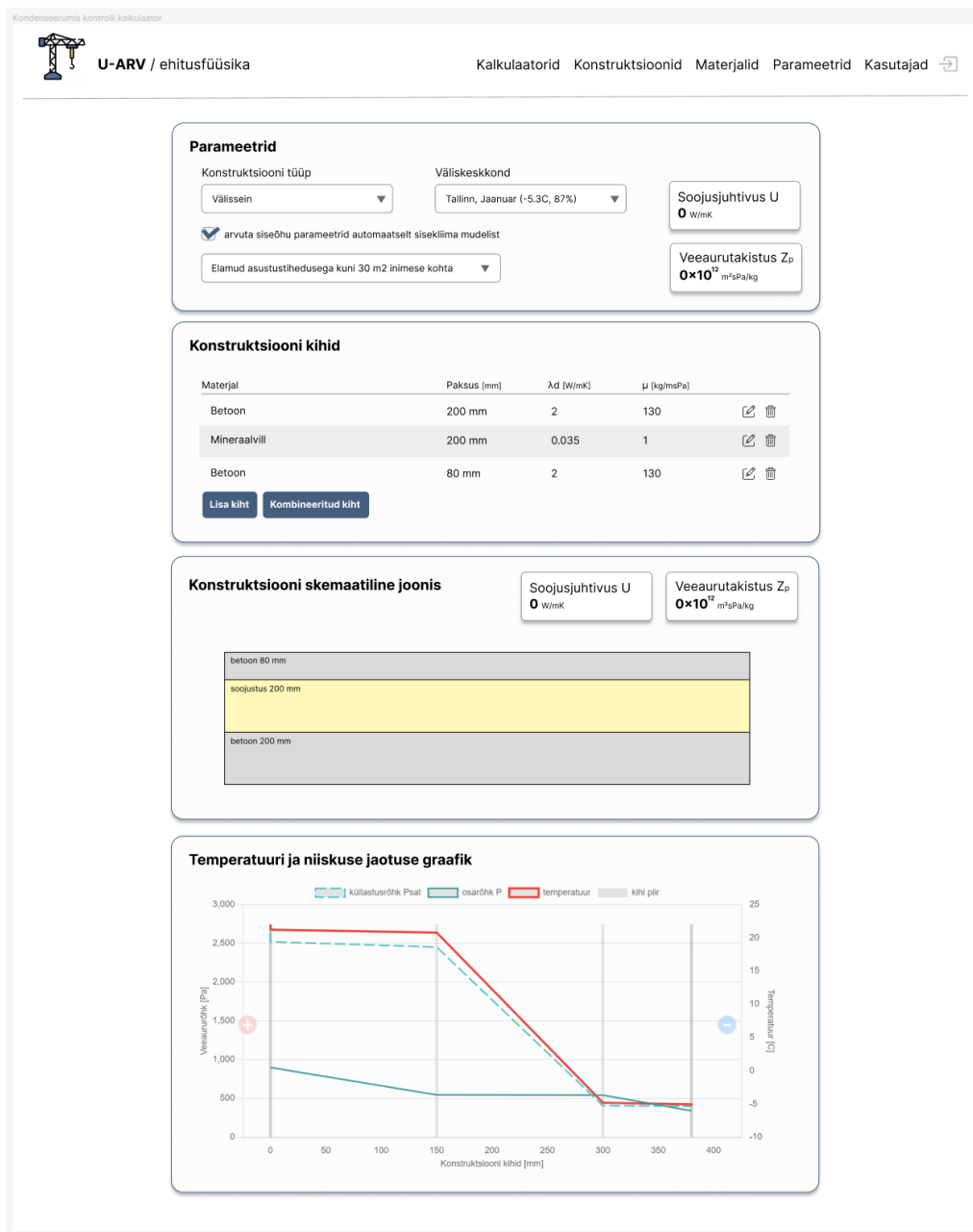
**allikas**

**Andmete allikas**

**Viide andmete allikale**

**Kommentaar**

Joonis 28. Materjali lisamise vorm



Joonis 29. Kalkulaatori vaade