TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andrei Vassiljev 185849IADB

Töömahtude töötlemise rakendus ehitusprojektidele

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

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: Andrei Vassiljev

02.01.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus, mis võimaldab ehitusprojekti mudeli põhjal töödelda ehitusmahtusid kasutades ainult vabavaralist tarkvara.

Arendusprotsessi analüüsi etapis uuritakse erinevaid olemasolevaid raamistikke ehitusprojektist info kätte saamiseks ning rakenduse arhitektuuri planeerimiseks. Lisaks valitakse viis ehitusmahtude töötlemiseks.

Arendusprotsessi käigus ehitatakse valmis veebirakendus ning kirjeldatakse kasutatavaid tehnoloogiaid. Lõpptulemusena valmib rakendus, kus on võimalik kuvada ehitusprojekti mudelit ja infot ning luua ehitusprojekti mudeli elementidest tööpaketid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34. leheküljel, 5 peatükki, 24 joonist, 3 tabelit.

Abstract

Creating an Application to Process Work Estimates in Construction Projects

The objective of the current thesis is to create a web application that allows a user to process work estimates based on a construction project design model.

In the analysis part of the development different frameworks will be investigated for extracting information from the design model and for planning the architecture of the application. In addition, a method for processing work estimates will be chosen.

During the development process a working web application will be built and technologically described. The end product will be an application that can visualize the design model and information and allow the creation of work packages out of the model elements.

The thesis is in Estonian and contains 34 pages of text, 5 chapters, 24 figures, 3 tables.

Lühendite ja mõistete sõnastik

.NET	Microsofti arendatud tarkvararaamistik.
API	*Application programm interface*. Rakendusliides.
BIM	*Building Information Model, Building Information Modelling, Building Information Management.* Ehitusinformatsiooni mudel, modelleerimine, juhtimine [1].
CCI	*Construction Classification International.* Rahvusvaheline ehituse standard [2].
CCI-EE	*Construction Classification International with local additions.* Rahvusvaheline ehituse standard ühes rahvuslike lisadega [2].
ERD	*Entity Relationship Diagram*. Olemi-suhte diagramm [3].
HTML	*Hypertext Markup Languge.* Hüperteksti markeerimiskeel.
HTTP	*Hypertext Transfer Protocol.* Hüperteksti edastusprotokoll.
HTTPS	*Hypertext Transfer Protocol Secure.* Turvaline hüperteksti edastusprotokoll.
IDE	*Integrated development environment.* Integreeritud Programmeerimiskeskkond.
IFC	*Industry Foundation Classes*, töös ka IFC formaat. Ehitatud keskkonna standardiseeritud digitaalne kirjeldus [4].
JSON	JavaScript Object Notation. Programmeerimiskeelest sõltumatu kergekaaluline andmevahetusvorming [5].
JWT	*Json Web Token*. JSON-põhine standard ligipääsu sõnede koostamiseks [6].
REST	*Representational State Transfer*. Veebiteenusega suhtlemise liidese arhitektuur [7].
RKAS	Riigi Kinnisvara AS.
SSL	*Secure Sockets Layer.* Võrgu turvasertifikaat [8].

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Üks populaarsematest ja efektiivsematest äri optimeerimise võimalustest on protsesside automatiseerimine. Ajalooliselt vanimate valdkondade hulka kuuluv ehitus on võrreldes tööstuse sektoriga maha jäänud infotehnoloogiliste abivahendite rakendamisega. Näiteks ehitavad autosid juba aastakümneid robotid konveierliinidel. Robotite rakendamine objektil on küll välja arendamisel, aga alles katsetamise etapis. Eestis toimub siiani ehitushankeid arvutustabelite baasil nii-öelda käsitsi, kuigi teoreetiliselt on võimalik seda optimeerida sobiva tarkvara välja töötamisega.

Ehitusturul ei ole hinnapäringud rangelt standardiseeritud. See tähendab, et tööde ning materjalide liigitamine ja struktureerimine võib erineda objektilt objektile. Lisaks nõuab selle protsessi õige hindamine haritud ja kogemustega spetsialiste. Eestis enim levinud ehitusprotsesside mudel põhineb peatöövõtul. Enamik suuremaid ehitushankeid põhineb RKAS (Riigi Kinnisvara AS) poolt välja töötatud tehnilistel nõuetel [9]. Juba aastaid on nendes nõuetes sees BIM (*Building Information Model*) projekteerimine. Keerukate 3D mudelite asjakohane hindamine võib olla võimete kohane peatöövõtjale, aga on väga palju alltöövõtjaid, kellel puuduvad selleks vastavad spetsialistid. Selle tõttu on hanke protsessis oht saada hinnapakkumised, mis ei ole omavahel võrreldavad. Need võivad sisaldada mingil määral erinevalt hinnatud materjalide ja tööde mahtu või halvimal juhul jäetakse mingi osa töödest välja. Lisaks on ehitushanked üles ehitatud enamasti alampakkumistele, mis samuti soosib vea tekkimise ohtu, kuna ajaline ressurss on piiratud. Üheks ehituse suurimaks probleemiks on eksimused tööde planeerimisel, millel on otseselt ajalised ja rahalised tagajärjed. Kogemustega peatöövõtjad vormistavad küll üldjuhul sellised lepingud, et vastutus on alltöövõtja kanda ning arvestavad ajagraafikutel ka venimistega, aga lõpuks kaotavad mõlemad pooled. Alltöövõtja võib kaotada kasumit leppetrahvide tõttu ning peatöövõtjal oleks parema planeerimisega võimalik objekt kiiremini valmis ehitada.

Üks võimalik lahendus olukorra parandamiseks on luua platvorm, mis katab kogu ehitise elutsükli. Alustades planeerimisest ja projekteerimise hankest ning lõpetades valmis

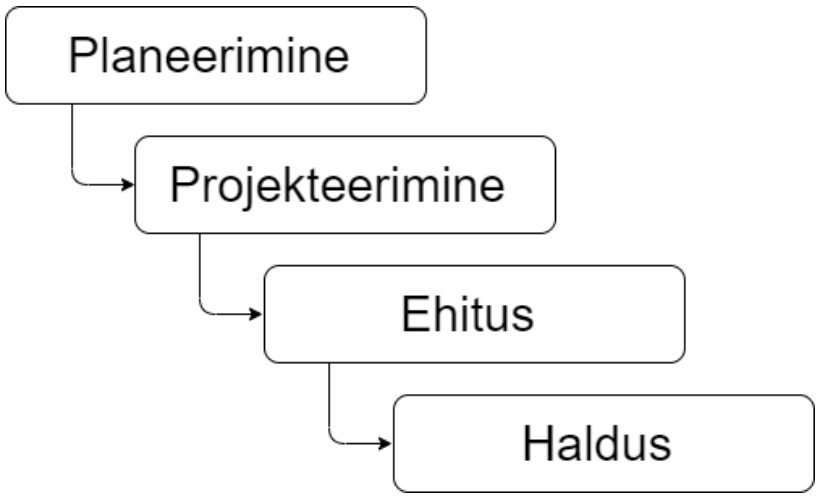
ehitatud ehitise halduse ja hooldusega. Infovahetuse seisukohalt on parim lahendus ehitada platvorm üles põhinedes vabavaralistel standarditel, et ei tekiks ebasoodsaid olukordi ühe või teise projekteerimise tarkvara suhtes. Kogu elutsükli ulatuses lahenduse koostamine on aga mahult liiga suur ühe diplomitöö skoobi jaoks ning selle tõttu tuleb valida sellest üks osa. Käesoleva töö raames üritab autor välja pakkuda lahendust, kuidas projekteerimise etapi lõpus teha ettevalmistused ehitushanke korraldamiseks. Selleks tuleb luua veebirakendus, mis võimaldab töödelda ehitusmahte vabalt kättesaadaval standardil põhineval IFC (*Industry Foundation Classes*) formaadis BIM ehitusprojekti mudelist.

2 Loodava veebirakenduse analüüs

Rakenduse loomise eel on vajalik sõnastada probleem ja püstitada lähteülesanne ning teha ära peamised valikud kasutatava tehnoloogia osas.

2.1 Lahendatav probleem

Ehituse elutsüklit on võimalik erinevate teooriate põhjal jagada osadeks erineval moel. Käesolevas töös käsitletav probleem ei sõltu otseselt, kuidas selline jaotus teha ning selle tõttu valitakse lihtsustamise eesmärgil üldisem neljaosaline jaotus, mida on kujutatud joonisel 1.



Joonis 1 Ehitusprotsesside skeem.

Planeerimise etapis on üldjuhul olemas kavandatava ehitustegevuse idee ja ligikaudne eelarve ning viiakse läbi projekteerimise hange. Projekteerimise käigus koostatakse etappide kaupa ideele ja eelarvele vastav ehitusprojekt ning üldjuhul taotletakse ehitusluba. Ehitusprojekt on aluseks ehitushankele, mille põhjal valitakse ehitaja ning algab ehitustegevus. Pärast ehitustööde lõppemist jätkub haldus ja hooldus.

Ehitusprojektid koostatakse tänapäeval BIM põhimõttel. BIM on infomudel ehitatavast hoonest või ehitusinformatsiooni modelleerimine, tegevus, mille tagajärjel sünnib ehitise infomudel või ehitusinformatsiooni juhtimine ehk mudeli kasutamine äri- ja

ehitusprotsesside juhtimiseks, organiseerimiseks ja kontrollimiseks ehitise eluea kõikides etappides [10].

Ehitusprojekt koosneb erinevatest osadest, mille koostavad erinevad spetsialistid kasutades selleks erinevaid tarkvarasid. Näiteks on üldjuhul projektis eraldi osadena lahendatud arhitektuur, konstruktsioonid, elekter, küte ja ventilatsioon. Projekteerimise tarkvarad kasutavad erinevaid standardeid ja faili vorminguid, mis tihti ei ühildu omavahel ideaalselt. Selle probleemi lahenduseks töötas välja ettevõte buildingSMART standardiseeritud IFC formaadi, mille eesmärk on avatud BIM andmevahetus [4].

Käesoleva töö raames lahendatava probleemi asukoht joonisel 1 on ehituse ja projekteerimise vahel – kuidas töödelda ehitusprojekti mudeli andmeid ehitushanke ja tööde planeerimiseks kastutavale kujule. IFC formaadi valik ehitusprojekti andmete töötlemiseks võimaldab jääda neutraalseks erinevate konkureerivate ehitustarkvarade vahel ning kasutada olemasolevaid vabavaralisi raamistikke ilma litsentsi probleemideta.

2.2 Ülesande püstitus

Eesmärk on kavandada ja arendada veebirakendus, mis võimaldab kasutajal saada ligipääsu IFC formaadis ehitusprojekti mudeli andmetele ning neid andmeid grupeerida ja töödelda ehitushankes kasutatavale kujule.

Mudel tuleb kuvada visuaalselt interaktiivsel ruumilisel kujul, et kasutajal oleks võimalik selles ringi liikuda parema mõistmise eesmärgil. Lisaks on vaja teha mudeli elementide info väljavõte. Selleks, et saadud info muuta kasulikuks, on seda vajalik töödelda ja grupeerida.

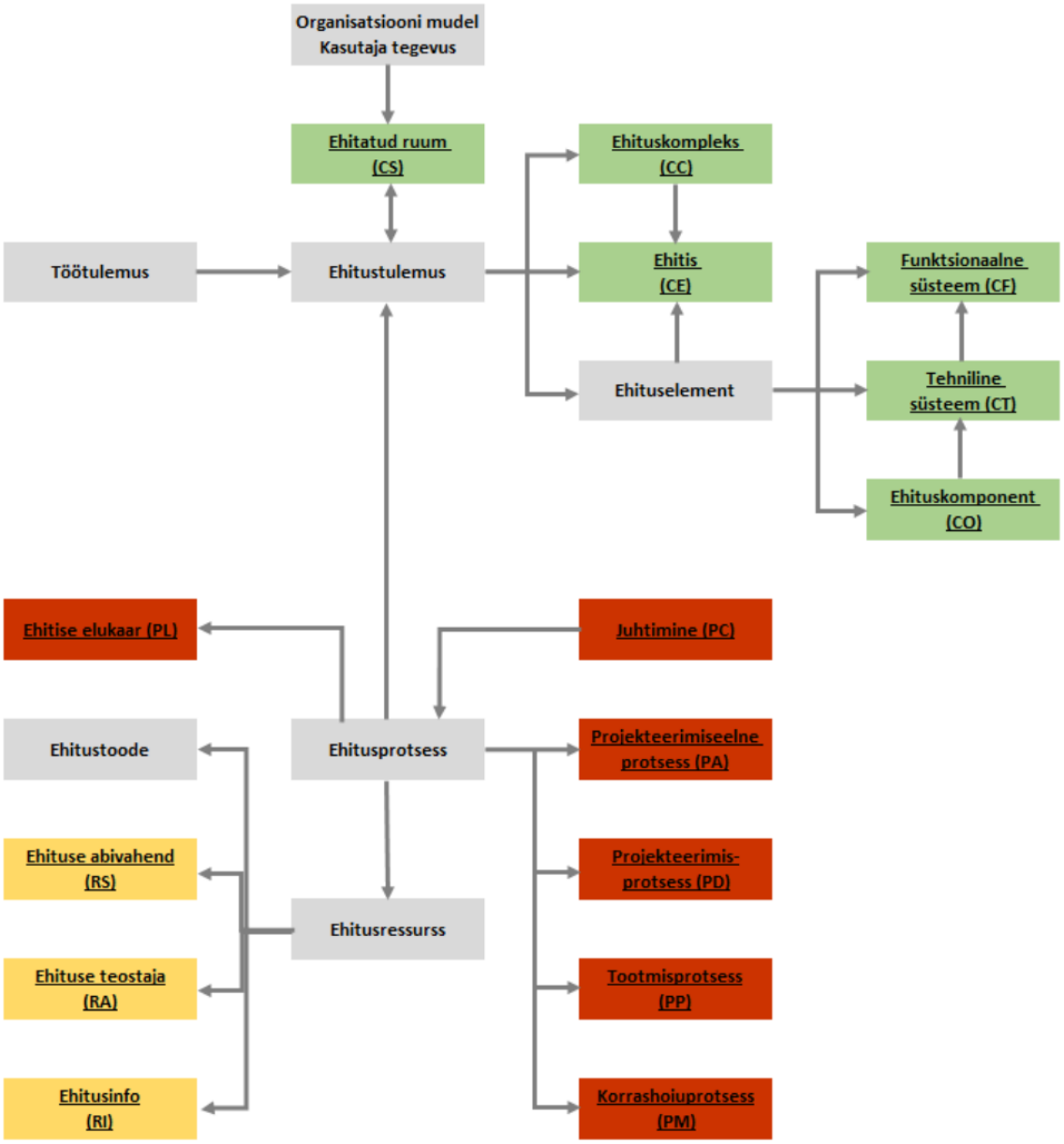
BIM mudelis üks ehituse ühik on element ning see on näiteks välisseina lõik ühest nurgast teiseni. Sellel seinal on mitu kihti ning need on näiteks järgevad alates väljastpoolt:

- fassaadi krohv ja värv,
- soojustuse kiht,
- kergbetoonist ehitusplokk,
- siseviimistlus.

Iga kiht võib sisaldada omakorda erinevaid materjale ning need võivad vajada paigaldust erinevatel ajahetkedel. Ülal toodud näite puhul on sein jaotatud nelja tööetappi.

Projektijuhtimise teoorias on võimalik töö jaotamiseks kasutada tööpaketi mõistet. Tööpakett on projekti alamosa, mida saab rakendada mingi projekti osa ellu viimiseks [11]. Loodav veebirakendus peab võimaldama valida ühe või mitu ehitusmudeli üksikosa ning luua nendest üks kuni mitu tööpaketti. Iga tööpakett on vaja liigitada konkreetseks ehituse tööliigiks. Kasutades tööpakettide teooriat tuleb eelneva näite puhul luua esimeseks paketiks konstruktiivne töö liigiga kergbetoonist müüritise püstitamine. Sama sein vajab valmimiseks veel kolme tööpaketti. Ühes tööpaketis saab olla omakorda mitu elementi, mis võimaldab luua grupeeringuid. Selline grupp moodustuks näiteks, kui näidis seina mudeli elemendile lisaks valime kõik samasuguste kihtidega sama korruse seinad.

Tööpakett vajab lisaks kirjeldust, milleks saab kasutada klassifikaatorit. 2020.a. lõpus valmis Tallinna Tehnikaülikooli juhtimisel Majandus- ja Kommunikatsiooni-ministeeriumi arendusprojekt Ehituse ühtse klassifitseerimissüsteemi loomiseks, mis toetuks rahvusvahelistele standarditele ja võimaldaks ehitusprotsesse klassifitseerida ühtsetel alustel. Projekti lõpptulemusena valmis klassifikatsioonisüsteem CCI (*Construction Classification International)* ühes rahvuslike osadega CCI-EE (*Construction Classification International with local additions*) [12]. Klassifikatsioonisüsteemi põhimõtteline skeem on kujutatud joonisel 2.



Joonis 2. CCI-EE klassifikatsiooni süsteemi põhimõtteline skeem [12].

Kavandatav veebirakendus peab olema võimeline lugema ehitusprojekti mudeli infot IFC formaadis failist, seda kuvama ning võimaldama grupeerida mudeli osad ehituse tööliigi järgi klassifitseeritud tööpakettideks.

2.3 Nõuete määramine

Nõuete määramisel on otstarbekas lähtuda lõppkasutajate vajadustest. Käesoleva töö raames nõuete selgitamiseks vestles töö autor kolme ehitussektoris enam kui viie aastast töökogemust omava isikuga, kelle tööülesannete täitmisel oleks loodavast rakendusest kasu.

Kasutajagruppide üldistamiseks on järgnevates alampeatükkides loodud persoonad ning nende vajadused. Persoona on kasutajagruppi esindav personaliseeritud isik (mitte päris isik), kes on loodud oma kasutajagrupi kõige tüüpilisema universaalse näitena. Nõuded põhinevad persoonadel, kus põhirollideks on tööpakettide looja (edaspidi projektijuht) ning andmete lugeja (edaspidi eelarvestaja) [13].

2.3.1 Funktsionaalsed nõuded

Projektijuhina soovin ma:

- luua uue projekti info salvestamiseks või täiendada olemasolevat projekti,
- kuvada ehitusprojekti andmed BIM mudelist IFC formaadis,
- kuvada BIM mudelit interaktiivses 3D vaates,
- luua BIM mudeli elementidest tööpaketid ning lisada neile ehitusliku tööliigi klassifikaator,
- kuvada loodud tööpakettide loend.

Eelarvestajana soovin ma:

- kuvada ettevalmistatud projekti,
- kuvada BIM mudelit interaktiivses 3D vaates,
- kuvada loodud tööpakettide loend.

2.3.2 Mittefunktsionaalsed nõuded

Projektijuhina ja eelarvestajana soovin ma:

- eesti keelset kasutaja keskkonda,
- avada rakendus veebi brauseris, mis ühildub standardiga HTML (*Hypertext Markup Languge*) *Living Standard* [14].

2.3.3 Tulevikus loodavad funktsionaalsused

Rakenduse skoobile eelnev keskkond aitab hallata ehituse planeerimist ja projekteerimist.

Skoobile järgnev tegevus hõlmab tööpakettide edasist töötlemist ja omakorda grupeerimist ehitushanke korraldamiseks sobivale kujule. Terviklik halduskeskkond võimaldab läbi viia kogu ehitushanke ning valida sobiv pakkuja. Seejärel on võimalik tööpakettide põhjal koostada tööde graafik ning läbi viia ehitustegevus.

2.4 Tehnoloogia valik

Arendatava veebirakenduse skoopi arvestades ei ole otstarbekas luua täiesti uut lahedust IFC failide töötlemiseks, kuna ajakulu sellele oleks liiga suur. Järgnevate peatükkide käigus uuritakse olemasolevate raamistike sobivust ning teostatakse selle põhjal tehnoloogia valik.

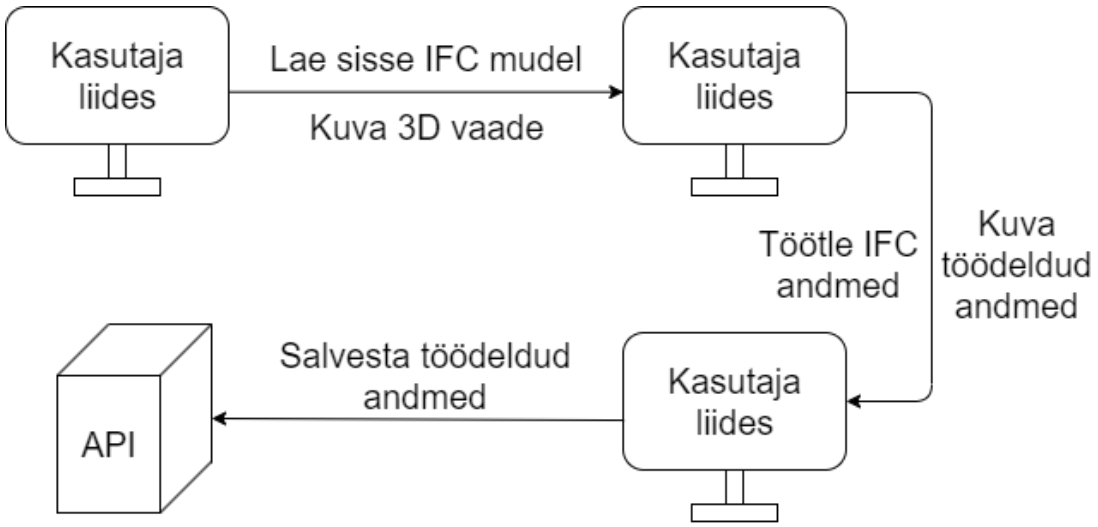
2.4.1 IFC raamistiku valik

Autori poolt püsitatud põhikriteeriumid raamistike uurimisele on järgnevad:

- raamistik peab olema vabavaralise litsentsiga,
- raamistiku dokumentatsioon peab olema internetist vabalt kättesaadav,
- raamistik peab võimaldama täita peatükis 2.2. ja 2.3. esitatud nõudeid.

IFC.js BIM toolkit for JavaScript.

Käesoleva töö koostamise hetkel on tegemist umbes kahe aasta vanuse projektiga, mis on selle lühikese aja jooksul autori hinnangul rakendanud üsna palju funktsionaalsust ning on aktiivses kasvu perioodis ja kogub populaarsust. IFC töötlemise loogika tuum on kirjutatud C++ keeles ning kompileeritud WebAssembly keelde. See loogika on peidetud JavaScript keeles kirjutatud raamistiku sisse. 3D mudeli kuvamiseks on kasutatud populaarset raamistikku Three.js. Joonisel 3 on kujutatud andmevahetuse protsesside skeemi selle raamistiku kontekstis [15].



Joonis 3. IFC.js raamistiku andmevahetuse protsesside skeem.

Ülesehituse põhimõte on, et kogu andmete töötlemise arvutustöö teeb ära kasutajaliidest jooksutav seade lokaalselt. Kasutades seda raamistikku oleks loodava rakenduse skaleerimine tulevikus lihtne, kuna serveri poolne ülesanne lihtsustatud kujul on töödeldud andmete salvestamine ja laadimine. Samuti on see hea toote arenduse kiiruse jaoks, kuna peamine loogika on kokku koondatud ühte kohta – kliendirakendusse.

xBIM toolkit

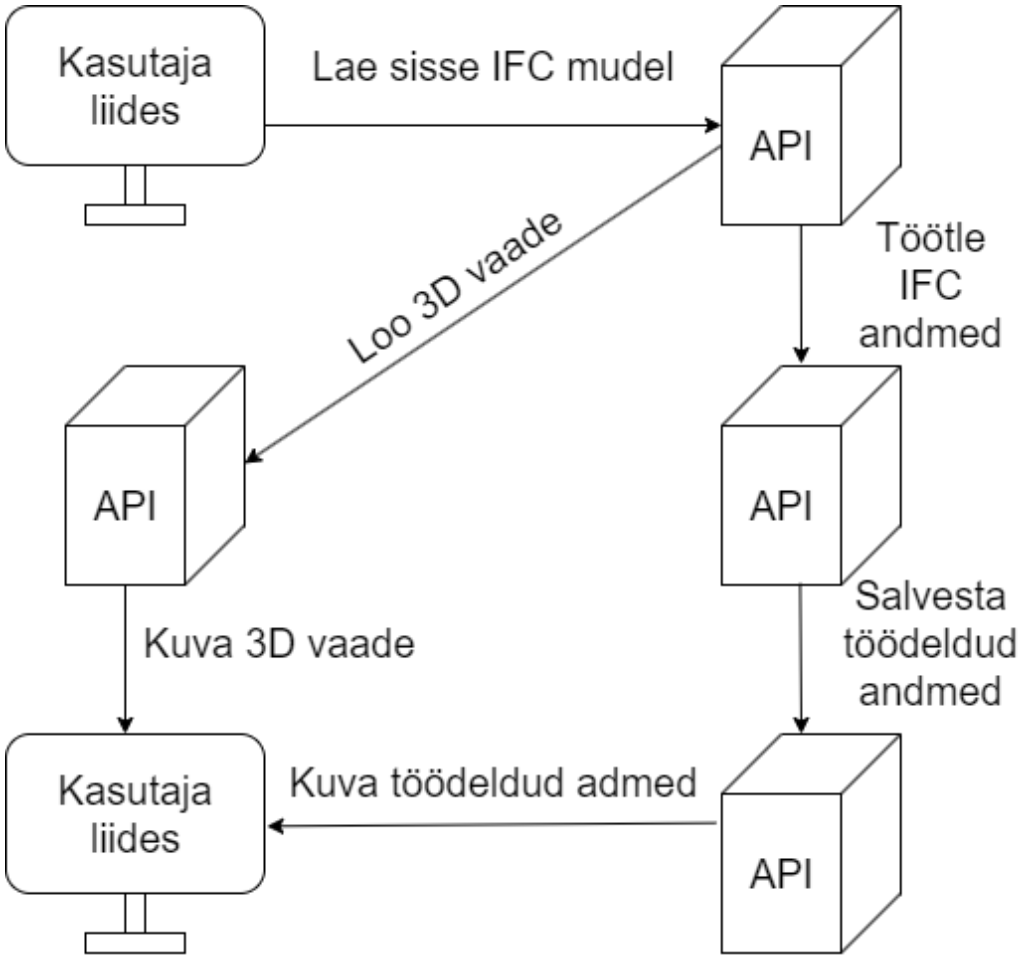
Võrreldes eelneva raamistikuga on xBIM oluliselt küpsem lahendus. Vaadates nende koodihoidla ajalugu on viimasel ajal tehtud uuendusi vähe ja sellest võib järeldada, et peamised probleemid ja puudujäägid on juba likvideeritud. Arendatava veebirakenduse jaoks on olulised selle kolm järgnevat komponenti:

1. xBIM Essentials – C# keeles .NET (Microsofti arendatud tarkvararaamistik) raamistikule kirjutatud IFC failide lugemise ja töötlemise raamistik.

2. xBIM Geometry – C++ keeles kirjutatud geomeetria töötlemise rakendus. See on vajalik IFC mudeli töötlemiseks(ja optimeerimiseks) WexBIM formaati, kuna xBIM 3d mudeli kuvamise tööriist ei toeta IFC mudeli kuvamist otse.

3. xBIM Web UI – JavaScript keeles kirjutatud raamistik 3D mudeli kuvamiseks, mis kasutab WebGL funktsionaalsust, aga on täiesti eraldiseisev kolmandate osapoolte WebGL raamistikest [16].

Koodihoidlas olevate näidistega tutvumise käigus hindas autor, et võrreldes IFC.js raamistikuga on tehtud rohkem optimeerimist ning IFC failide muutmiseks on võimalused paremad. Küll aga nõuab see keerukamat arhitektuuri ning rohkem serveri võimekust. Joonisel 4 on kujutatud andmevahetuse protsesside skeemi selle raamistiku kontekstis [17].



Joonis 4. xBIM raamistiku andmevahetuse protsesside skeem.

Kasutajaliides peab saatma esimese sammuna IFC faili serverisse rakendusliidesele ning kuna IFC failid võivad mahu poolest olla üsna suured, tekitab see suure kasutajate arvuga väga suure koguse andmemahu liiklust. Serveris tuleb IFC fail töödelda ümber WexBIM formaati ning see tagasi saata kliendirakendusele mudeli visuaalseks kuvamiseks. Samaaegselt saab rakendada rakendusliideses IFC andmete töötlemise ning tulemused saata tagasi kasutajaliidesele kuvamiseks.

Xeokit SDK

Xeokit on peamiselt keskendunud parima kasutajakogemusega mudeli visualiseerimisele. Mudelist info kätte saamine on võrreldes eelnevate raamistikega algeline ning selle viimane uuendus on käesoleva töö koostamise ajal välja antud kolm aastat tagasi. IFC mudeli muutmise kohta ei leidnud autor infot dokumentatsioonist. Arendatava veebirakenduse jaoks on olulised selle kolm järgnevat komponenti:

1. xeokit-metadata– C# keeles kirjutatud IFC failide lugemise konsooli rakendus.

2. xeokit-convert – JavaScript keeles kirjutatud geomeetria töötlemise rakendus. See on vajalik IFC mudeli töötlemiseks(ja optimeerimiseks) XKT formaati, kuna xeokit-bim-viewer 3D mudeli kuvamise tööriist on optimeeritud selle kuvamiseks.

3. xeokit-bim-viewer – JavaScript keeles kirjutatud veebirakendus 2D ja 3D mudeli kuvamiseks. Toetab lisaks IFC formaadile veel näiteks glTF, OBJ, STL, 3DXML, LAZ/LAS, CityJSON ja XKT formaate [18].

Esmase analüüsi käigus kogutud info võrdlemisel eelnevate raamistikega otsustas autor Xeokit raamistiku valikust kõrvaldada ning seda mitte täpsemalt edasi uurida.

Raamistiku valiku tulemus

Uuringu käigus välja valitud raamistikud koos põhiandmetega on kujutatud tabelis 1. Lisaks peatükis 2.4.1 kirjeldatud nõuetele otsustas autor lisada IFC faili modifitseerimise kriteeriumi. See ei ole vajalik käesolevas töös arendatava rakenduse jaoks, aga on väga kasulik funktsionaalsus potentsiaalsete edasi arendamise ideede jaoks.

Tabel 1. Raamistike üldandmed.

Raamis-tiku nimi	**3D mudeli kuva-mine**	**BIM andmete lugemine**	**IFC faili modifitseeri-mine**	**Keel**	**Arenda-mise alguse aasta**	**Viimase uuenduse aasta**
IFC.js	Jah	Jah	Jah	Javascript	2021 [19]	2022 [19]
xBIM toolkit	Jah	Jah	Jah	C#, Javascript	2014 [17]	2022 [17]
xeokit	Jah	Jah	Ei	Javascript	2014 [20]	2021 [20]

Kolmest uuritud potentsiaalsest kandidaadist on kõrvale jäetud Xeokit, kuna selle võimekus jääb autori poolt kogutud info põhjal alla IFC.js ja xBIM raamistikele. Alles jäänud kahest oleks mõlemad sobilikud arendatava veebirakenduse loomiseks. Autori otsus on valida IFC.js, kuna selle põhjal veebirakenduse loomise keerukus on väiksem ning see vajab vähem serveripoolseid ressursse.

2.4.2 Kasutajaliidese raamistiku valik

Autori soov on rakenduse loomise keerukust hoida võimalikult madalal. Samuti peab olema toetatud edasi arendamise võimalus luua projektijuhtimise ja haldamise terviklik veebikeskkond. Autor soovib kasutada mõnda olemasolevat ja laialt levinud veebiraamistikku. Keerukuse vähendamise kaalutlustel on parimaks valikuks JavaScript keele põhine raamistik, kuna samal keelel põhineb ka varasemalt valitud IFC.js raamistik.

Stack Overflow arendajate küsitlus 2021 on üks suurimaid uuringuid tarkvara arenduse maailmas, milles 2021. aastal osales üle 80 000. inimese. Joonisel 5 on toodud kümme populaarsemat veebiraamistikku. Nendest viis populaarsemat on kõik JavaScript põhised. Arvestades loodava rakenduse madalat keerukust ei ole otstarbekas neid põhjalikult uurida ja võrrelda. Autor omab vähest isiklukku kogemust Vue.js tehnoloogiaga ning otsustab selle kasuks, et hoida kokku arendamise aega vähem tuttavate raamistike õppimise arvelt.

Joonis 5. Kuvatõmmis Stack Overflow uuringust veebi raamistike kategoorias [21].

2.4.3 Rakendusliidese raamistiku valik

Valitud IFC raamistiku loogika töötab täielikult kasutajaliidese poolel. Käesoleva töö raames arendatav veebirakendus võiks tulevikus olla üks osa suuremast veebikeskkonnast ning selle arendamine käesoleva töö raames toimub lihtsustatud kujul. Suurema keskkonna puhul võivad tehnilised vajadused ja nõuded olla erinevad. Kuna serveri poolt on vaja peamiselt ainult andmebaasiga suhtlemiseks, soovib autor valida võimalikult lihtsa lahenduse arendusprotsessi ajakulu säästmiseks. Autori suurimad kogemused arendamises on C# keeles ning .NET raamistikus. Varasema kogemuse põhjal valib autor .NET 6 raamistiku uuenduses välja tulnud kergekaalulise Minimal API (*Application programm interface*) lahenduse [22].

Kasutades Visual Studio tarkvara malli saab mõne liigutusega luua töötava API näidislahenduse, kus peamist loogikat on ainult 43 rida koodi. Selle sisse mahuvad ka näidisandmed ning see on kuvatud joonisel 6.

```
1. var builder = WebApplication.CreateBuilder(args);
2.
3. // Add services to the container.
4. // Learn more about configuring Swagger/OpenAPI at
   https://aka.ms/aspnetcore/swashbuckle
5. builder.Services.AddEndpointsApiExplorer();
6. builder.Services.AddSwaggerGen();
7.
8. var app = builder.Build();
9.
10. // Configure the HTTP request pipeline.
11. if (app.Environment.IsDevelopment())
12. {
13.     app.UseSwagger();
14.     app.UseSwaggerUI();
15. }
16.
17. app.UseHttpsRedirection();
18.
19. var summaries = new[]
20. {
21.     "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy",
   "Hot", "Sweltering", "Scorching"
22. };
23.
24. app.MapGet("/weatherforecast", () =>
25. {
26.     var forecast = Enumerable.Range(1, 5).Select(index =>
27.        new WeatherForecast
28.        (
29.            DateTime.Now.AddDays(index),
30.            Random.Shared.Next(-20, 55),
31.            summaries[Random.Shared.Next(summaries.Length)]
32.        ))
33.         .ToArray();
34.     return forecast;
35. })
36. .WithName("GetWeatherForecast");
37.
38. app.Run();
39.
40. internal record WeatherForecast(DateTime Date, int TemperatureC,
   string? Summary)
41. {
42.     public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
43. }
```

Joonis 6. Näidisprojekti Program.cs faili sisu.

2.5 Andmebaasi valik

Andmebaasid jaotuvad tüübi järgi relatsioonilisteks ja mitterelatsioonilisteks andmebaasideks [23]. Vastavale varasemalt püstitatud nõuetele sobib veebirakendusele väga hästi relatsiooniline andmemudel. Kuna käesolevas skoobis on tegemist üsna lihtsa struktuuriga, siis toetavad seda kõik levinumad relatsioonilised andmebaasid. Autor

otsustab valida MariaDB server lahenduse, kuna sellel on lihtsasti loetav dokumentatsioon ning veebis on palju kergesti leitavaid juhendeid selle seadistamiseks, näiteks MariaDB Tutorial keskond [24].

2.6 Arenduskeskkonna valik

Käesoleva töö skoopi arvestades ei ole otstarbekas investeerida palju aega arenduskeskkonna analüüsiks, kuna enamik levinumaid lahendusi on piisava võimekusega. Arenduskeskkonna puhul tugineb autor varasematele kogemustele vabavaraliste võimalustega. Integreeritud arenduskeskkonnaks klientrakendusele sobib Visual Studio Code, mis toetab hästi Vue.js valitud raamistikku ning serveripoolseks arenduseks Microsoft Visual Studio Community 2022 (64-bit), mis toetab hästi C# ja .NET. Koodihalduseks osutus valituks üks populaarsemaid keskkondi GitHub.

2.7 Veebirakenduse haldus

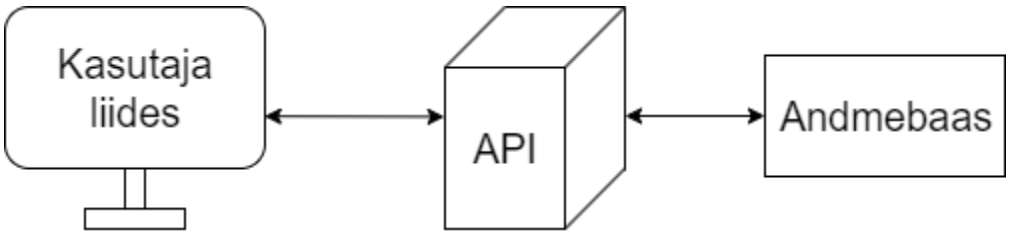
Veebirakenduse avalikkusele kättesaadavaks tegemiseks on lihtsaim võimalus kasutada pilveteenuseid. GitHub pakub tudengitele arendajate paketti, mis sisaldab erinevaid soodustusi ja tasuta litsentse rakendustele ning teenustele [25]. Nende hulgast sobib Microsoft Azure hästi rakenduse majutamiseks. Autori hinnangul piisab praeguses etapis olevale rakendusele standardne ja suhteliselt odava hinnaga B1s seeria virtuaalmasin ühe protsessori tuuma ja 1GiB mäluga, kuna peamine arvutuskoormus on kasutaja veebibrauseri kanda. Virtuaalmasina tarkvaraks sobib Linux Ubuntu versioon 20.04. Veebiserveri tarkvaraks sobib laialt levinud ja hästi dokumenteeritud Apache HTTP Server. Käesoleva töö esitamise ja kaitsmise ajal on rakendus ligipääsetav aadressilt http://dreifc.live.

2.8 Arhitektuur

Loodava veebirakenduse arhitektuur koosneb kolmest järgnevast osast:

- kasutaja veebibrauseris jooksev klientrakendus,
- virtuaalmasinas jooksev API serverirakendus,
- virtuaalmasinas jooksev andmebaas.

Joonisel 7 on kujutatud osade omavahelise suhtluse põhimõtteline skeem. Kasutajaliides ei suhtle otse andmebaasiga.



Joonis 7. Rakenduse arhitektuuri skeem.

2.9 Analüüsi kokkuvõte

Tabelis 2 on kokkuvõte valitud tehnoloogiatest ja arendusvahenditest.

Tabel 2. Valitud tehnoloogiad ja arendusvahendid.

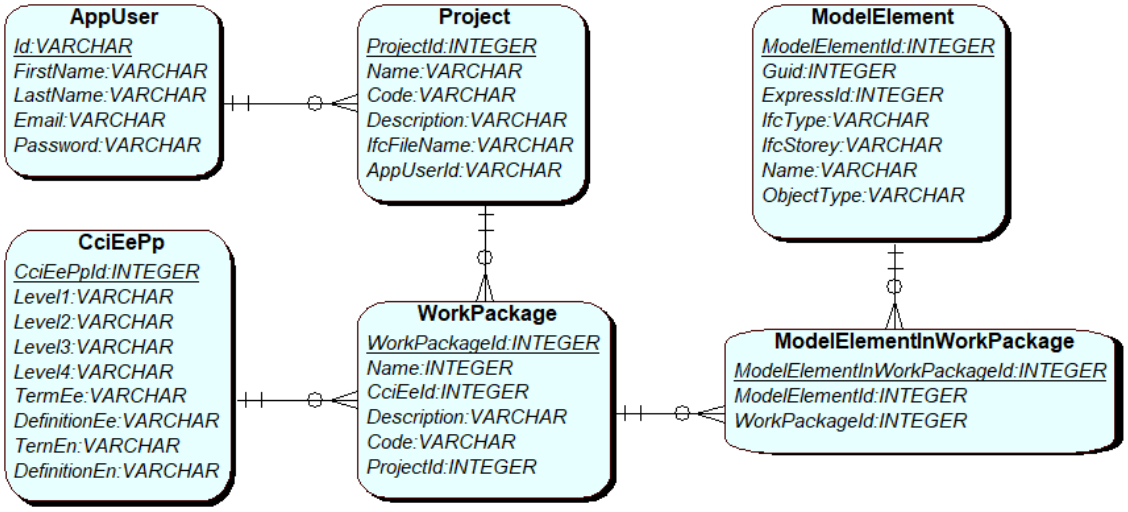
Kliendi poolne rakendus	IFC.js, Vue.js
Serveri poolne rakendus	Minimal API
Andmebaas	MariaDB
IDE (*Integrated development environment*)	Visual Studio Code, Visual Studio Community
Koodihaldus	GitHub
Veebirakenduse haldus	Microsoft Azure

3 Veebirakenduse arendus

Arendusprotsess on jaotatud arhitektuursete osade kaupa ning seda on käsitletud järgnevates peatükkides.

3.1 Andmemudel

Andmemudeli visualiseerimiseks sobiv võimalus on ERD (*Entity Relationship Diagram*) diagrammi, mis on kõige laiemalt levinud metoodika andmemudelite koostamiseks ja kirjelduse esitamiseks [3].



Joonis 8. Andmemudeli olemi-suhte diagramm.

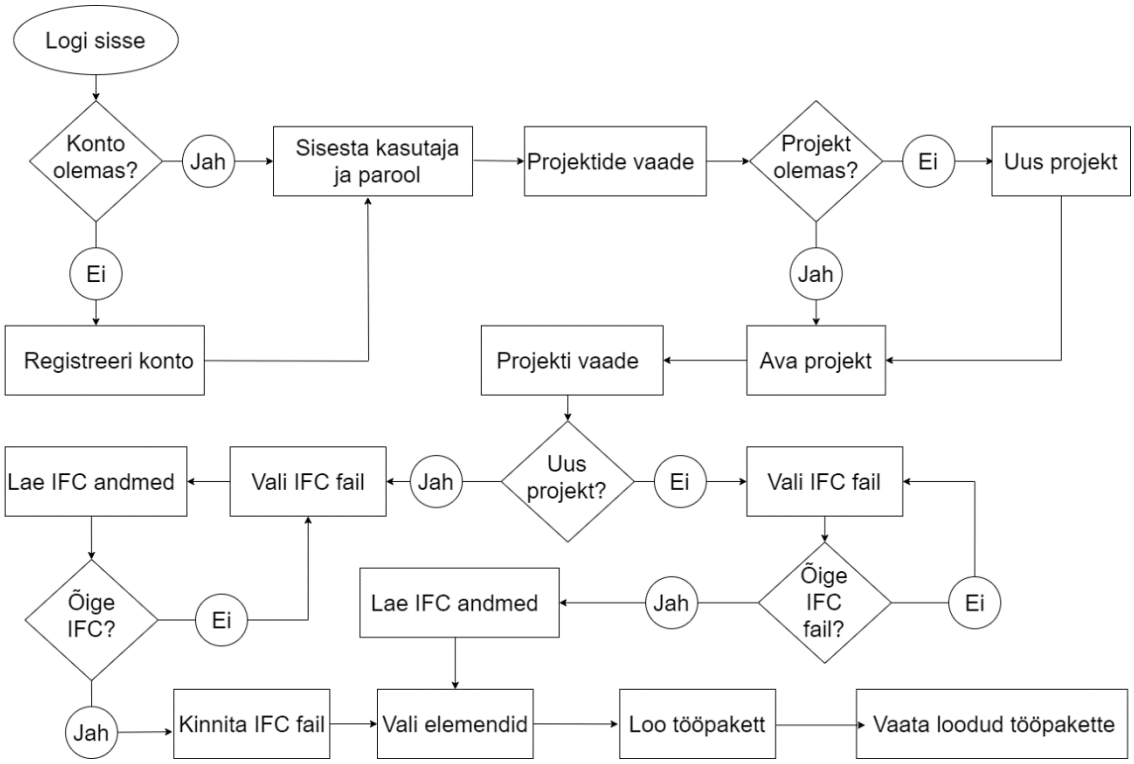
Joonisel 8 on kujutatud olemi-suhte diagramm ning tabelis 3 on kirjeldatud olemite semantikat.

Tabel 3. Olemite semantikad.

Tabeli nimi	**Semantika**
AppUser	Objekt-tüüpi olem, mis tähistab rakenduse kasutajat.
Project	Objekt-tüüpi olem, mis tähistab veebirakenduses ühte ehitusprojekti.
WorkPackage	Objekt-tüüpi olem, mis tähistab tööpaketti.
ModelElementInWorkPackage	Seos-tüüpi olem, mis tähistab seost tööpaketi ja mudeli üksikosa vahel.
ModelElement	Objekt-tüüpi olem, mis tähistab IFC.js raamistiku kaudu laetud BIM mudeli üksikosa. Salvestatakse andmebaasi tööpaketi loomisel.
CciEePp	Teatmik-tüüpi olem, mis tähistab tööpaketi klassifikatsiooni.

3.2 Kasutajakogemuse disain

Joonisel 9 on välja toodud rakenduse kasutajakogemuse põhimõtteline skeem. Rakenduse kasutamine ei ole võimalik anonüümselt ning tuleb luua kasutaja konto. Sisse logimise järel loob kasutaja uue projekti ning siseneb projekti vaatesse. Esimene samm projekti vaates on õige IFC faili valimine ning selle sidumine projektiga. Õige faili valimisel on abiks 3D mudeli kuvamine. Mudeli valimise järel laetakse sisse info mudeli elementide kohta ning kuvatakse need tabelis 3D mudeli all. Tööpaketi loomiseks on vaja märgistada vähemalt üks element ning valida tööpaketi loomine. Tööpaketi loomise vaates on vaja valida elementide kogule vastav klassifikaator. Tööpaketi loomist tuleb korrata kuni soovitud elemendid on kaetud tööpakettidega. Mudeli elementide tabeli vaade on võimalik asendada tööpakettide tabeliga. Tööpakettide vaatest on võimalik kuvada selles käsitletud elemente 3D joonisel.



Joonis 9. Rakenduse kasutajakogemuse diagramm.

3.3 Klientrakendus

Rakenduse nimeks sai valitud dreIFC, mis on kombinatsioon autori eesnimest ning rakenduses töödeldavast failivormingust.

3.3.1 Klientrakenduse failide struktuur

Rakendus koosneb järgnevatest vaadetest:

1. Pealeht – rakenduse tutvustus.
2. Konto registreerimine.
3. Sisse logimine.
4. Projektide nimekiri.
5. Uue projekti loomine.
6. Olemasoleva projekti muutmine.
7. Projekti vaade – selle osad:

7.1.IFC vaatur – IFC.js.

7.2.Elementide info tabel.

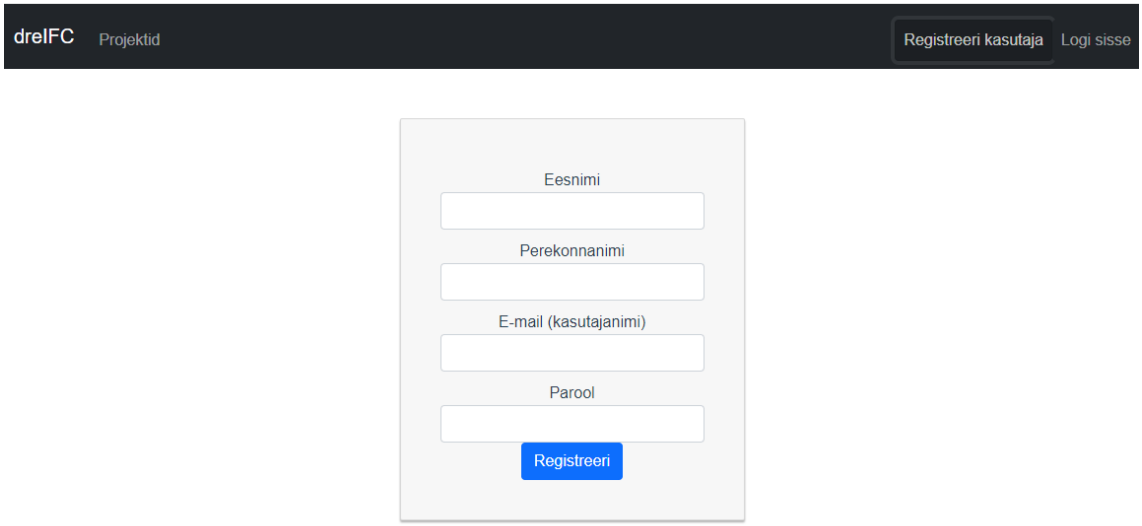
7.3.Tööpakettide tabel.

7.4.Tööpaketi loomise aken.

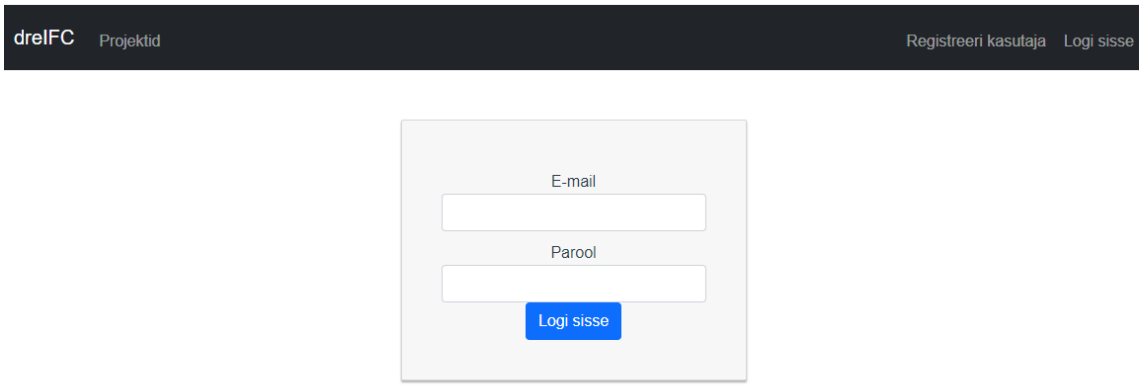
Lisas 2 on esitatud kuvatõmmis kaustade ja failide struktuurist. Vue.js raamistikul põhineva projekti struktuuri loomisel kasutas autor IFC.js raamistiku loojate poolt koostatud näidist Github versioonihalduse hoidlas [26].

3.3.2 Kasutajaliidese turvalisus

Kasutaja andmete turvalisuse tagamiseks on vaja rakenduse kasutamiseks luua konto ning seejärel sisse logida. Registreerimise ja sisselogimise vaated on antud joonistel 10 ja 11. Autentimiseks sai valitud JWT (*Json Web Token*) põhine lahendus. Kasutajaliidesesse sisse logimisel saadetakse serveri poolsele rakendusele päring ning õige kasutajanime ja parooli stsenaariumi puhul tagastab server JWT, mis salvestatakse Vuex Store komponendi abil [27]. Järgnevatel päringutel saadetakse JWT päringu osana serveri poole ning selle järgi tuvastav server autenditud kasutaja. Parima turvalisuse tagab andmete krüpteerimine HTTPS (*Hypertext Transfer Protocol Secure*) ja SSL (*Secure Sockets Layer*) kaudu, kuid antud töö raames seda ei saavutatud ning rakendused kasutavad HTTP (*Hypertext Transfer Protocol*) protokolli. Navigeerimine kasutajaliideses toimub läbi Vue Router komponendi [28]. Navigeerimisel kontrollitakse kasutaja autentimise staatust ning selle puudumisel suunatakse sisse logimise vaatesse.



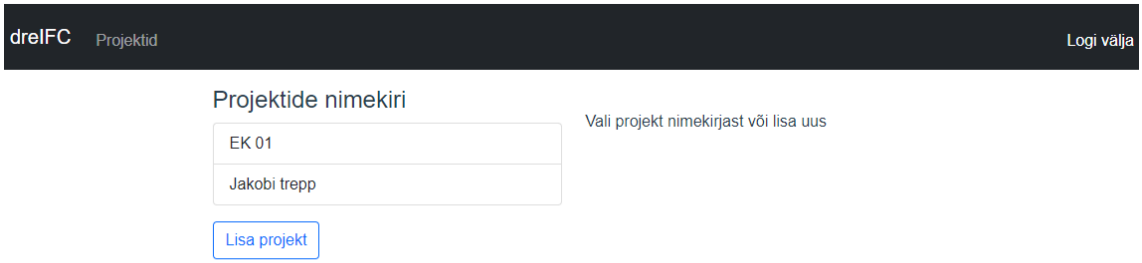
Joonis 10. Kasutaja loomise vaade.



Joonis 11. Sisse logimise vaade.

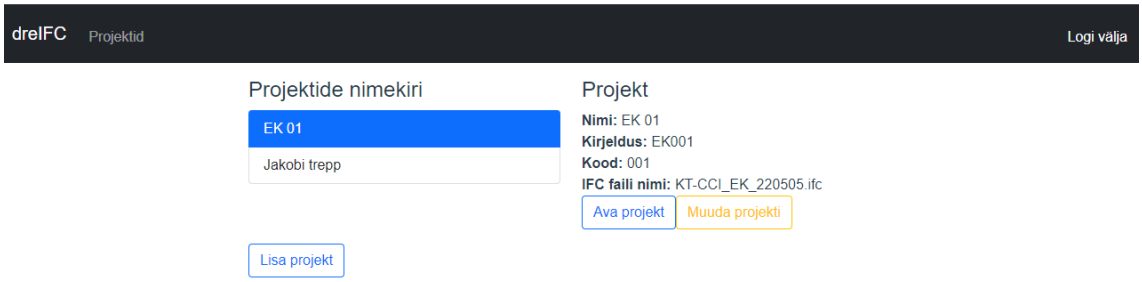
3.3.3 Kasutaja projektid

Eduka sisse logimise järel suunatakse kasutaja projektide nimekirja vaatesse, mida on kujutatud joonisel 12. Kasutajal on võimalik valida, muuta ja kustutada olemasolevaid projekte ning lisada uus projekt.



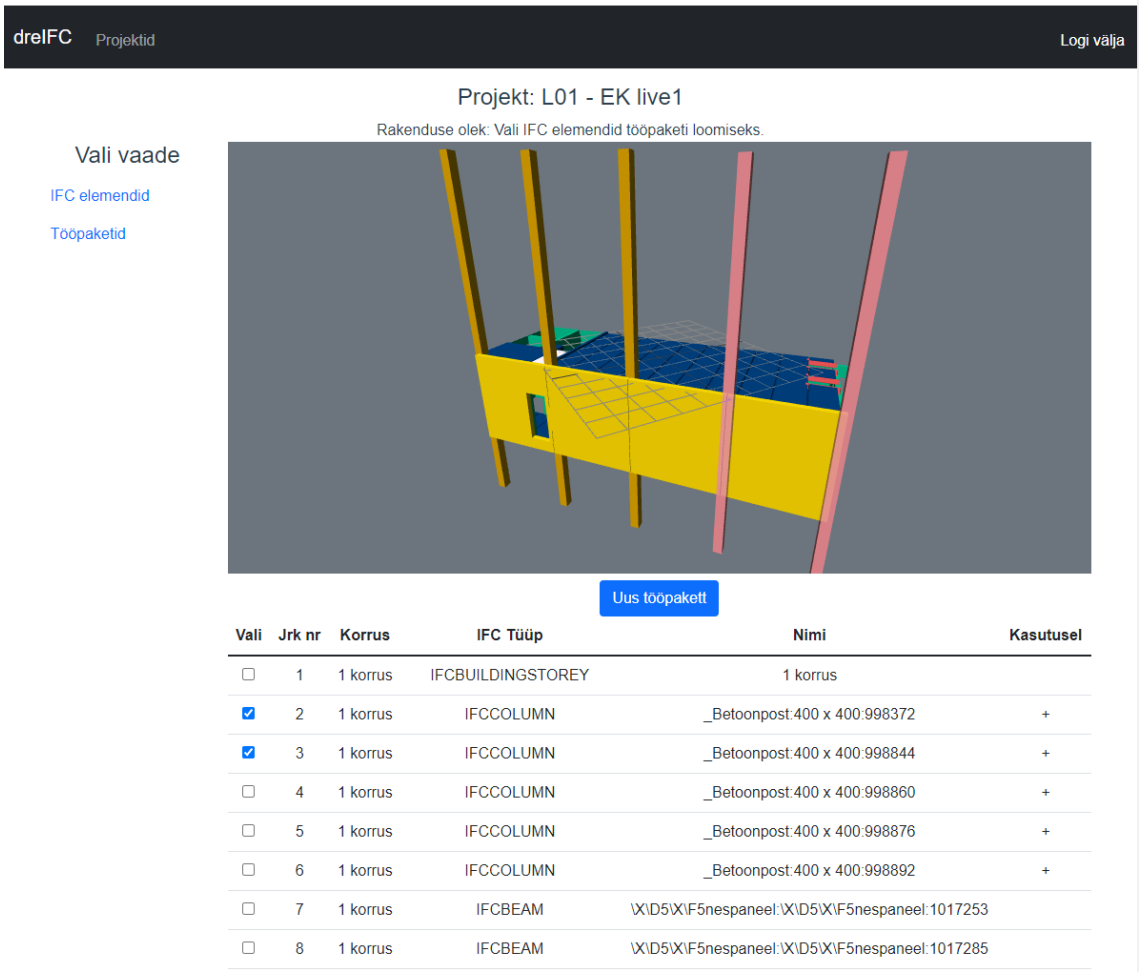
Joonis 12. Projektide nimekiri.

Projekti vaatesse edasi liikumiseks või projekti muutmiseks tuleb märgistada projekt nimekirjast ning selle kõrvale ilmub kirjeldus ning nupud edasi liikumiseks. Märgistatud projektiga vaadet on kujutatud joonisel 13.



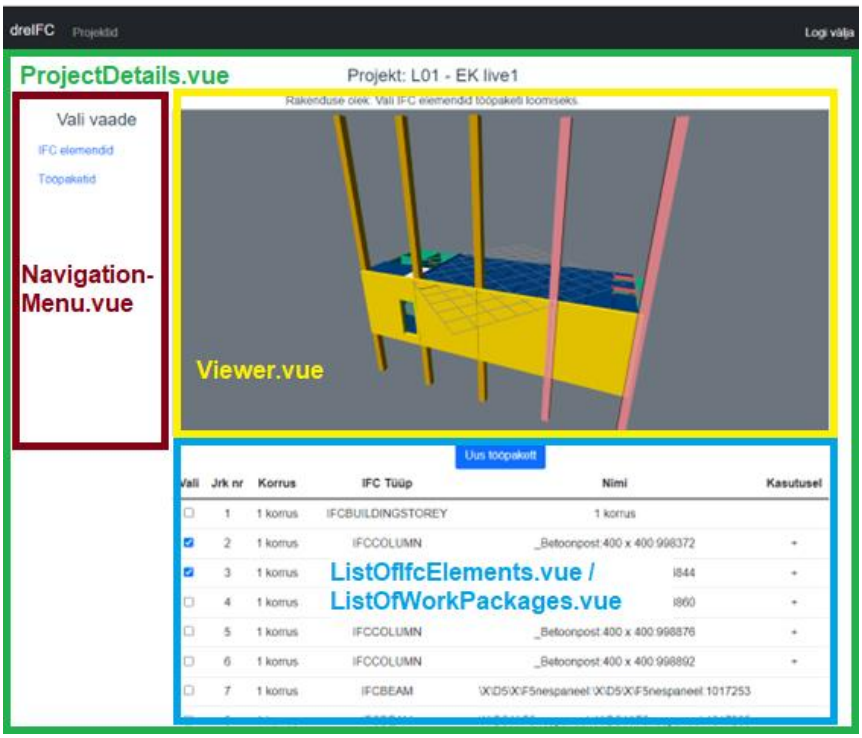
Joonis 13. Projektide nimekiri valitud projektiga.

Projekti avamisel suunatakse kasutaja edasi rakenduse põhivaatesse – projekti vaade, mille kuvatõmmis on esitatud joonisel 14.



Joonis 14. Peamise töövaate kuvatõmmis.

Projekti peamine töövaade on Vue.js raamistiku põhimõtete järgi jaotatud eraldi komponentideks ning seda jaotust on visualiseeritud joonisel 16. Komponendi NaigationMenu.vue sees on vasakul küljel asuv menüü, mille kaudu on võimalik valida, kas tabelis kuvatakse IFC elemente või tööpakette. Viewer.vue komponendi kaudu toimub 3D mudeli kuvamine ja andmete kätte saamine ehitusprojekti mudeli failist. Vaate alumises osas on sõltuvalt rakenduse olekust, kas ListOfIfcElements.vue, mis kuvab tabelit IFC elementidega või ListOfWorkPackages.vue, mis kuvab tabelit tööpakettidega. Eelnevalt loetletud osad asuvad ProjectDetails.vue komponendis, mis haldab nende omavahelist andmevahetust.



Joonis 15. Peamise töövaate komponendid.

3D kuva komponent Viewer.vue on ehitatud IFC.js teegi põhjal ning selle kaudu toimub ka info kättesaamine ehitusprojekti mudeli failist. Andmete sisse laadimine kutsutakse välja rakenduse kasutaja poolt ning rakenduse käivitub joonisel 16 esitatud meetod loadIfcData. Selle sees kutsutakse välja IFC.js komponendi integreeritud meetodit getSpatialStructure, mis tagastab kõik mudelis olevad elemendid puu kujulise andmekogumina.

```
1. async loadIfcData() {
2.   let ifcProject = await
   this.IFCManager.ifcLoader.ifcManager.getSpatialStructure(this.IFCManag
   er.scene.ifcModel.modelID);
3.   let elementArray = new Array;
4.
5.   await this.parseTree(ifcProject, elementArray);
6.
7.   this.$emit('ifcElementsArrayLoaded', elementArray);
8.
9.   this.dataLoaded = true;
10.
11.  if (this.savedFileLoaded == true){
12.          this.setIfcStatusReady();
13.  }
14.}
```

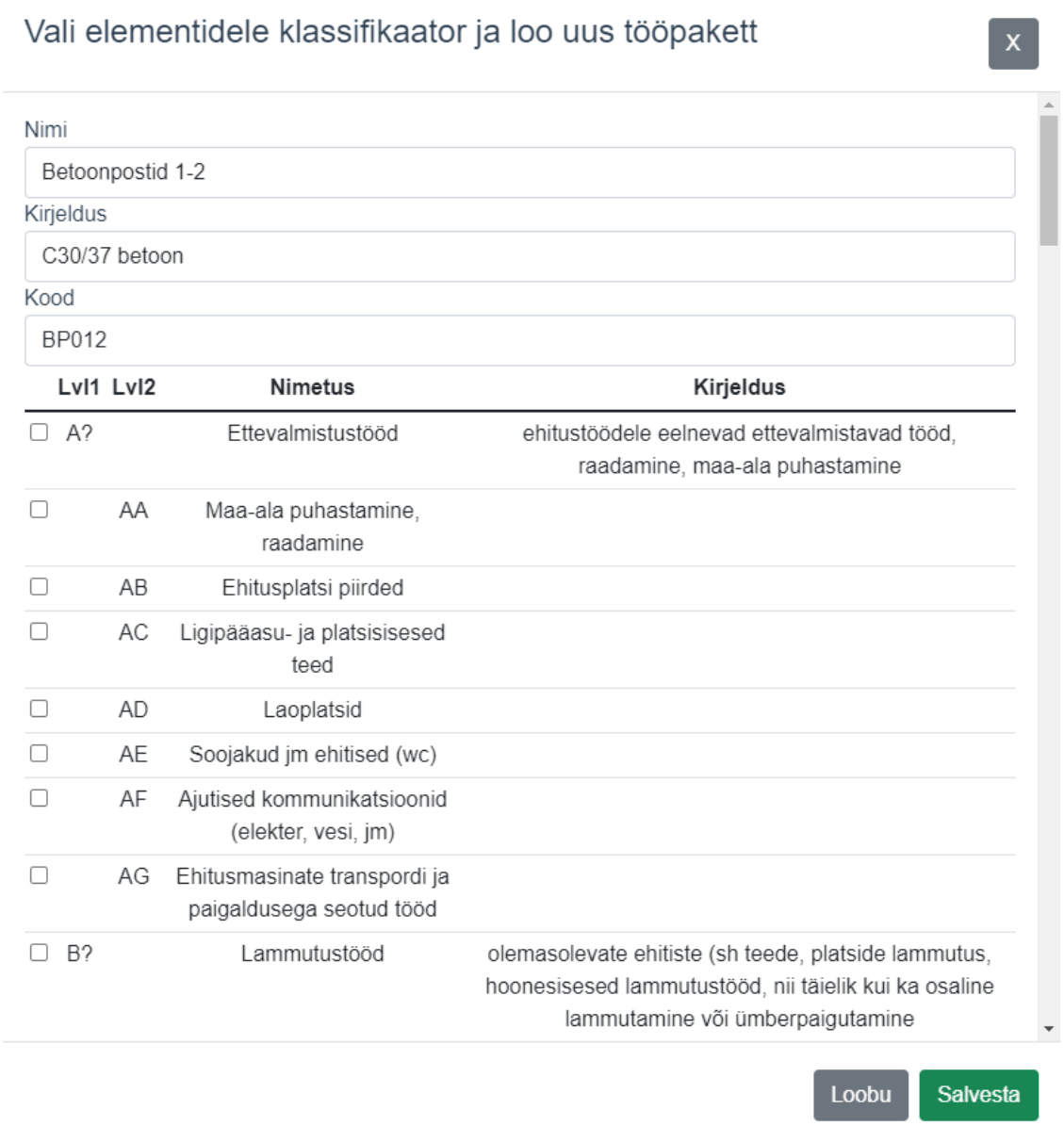
Joonis 16. Viewer.vue meetod loadIfcData.

Kasutajale andmete kuvamise eel töödeldakse andmeid joonisel 17 esitatud meetodis parseTree ning grupeeritakse elemendid korruste kaupa. Andmete puule rakendatakse rekursiivne töötlus, millega saab läbi käia kõik elementide tasemed. Igale elemendile kutsutakse omakorda välja IFC.js komponendi integreeritud meetod getItemProperties, mis lisab elemendile täiendava kirjeldava info. Lisaks filtreeritakse välja IFC abistavad elemendi tüübid, mille põhjal ei saa tööpakette luua. Tulemuseks on joonisel 15 esitatud elementide tabel vaate alumises osas.

```
1. async parseTree(ifcNode, buidingArray, storeyName){
2.   if (ifcNode.children.length === 0) {
3.     return buidingArray;
4.   } else {
5.     for (let i = 0; i < ifcNode.children.length; i++) {
6.       let element = ifcNode.children[i];
7.
8.       // Get detailed property object.
9.       let elementAllProps = await
   this.IFCManager.ifcLoader.ifcManager.getItemProperties(0,
   element.expressID, true);
10.
11.      if (element.type === 'IFCBUILDINGSTOREY') {
12.        storeyName = elementAllProps.Name.value;
13.      }
14.
15.      // Transfer IFC Type property to detailt property object.
16.      elementAllProps.ifcType = element.type;
17.
18.      // Add storey nr property.
19.      elementAllProps.buildingStorey = storeyName;
20.
21.      // Add prop for selection.
22.      elementAllProps.selected = false;
23.
24.      // Add object to parent building storey, filter out site and
   building elements
25.      if (element.type !== 'IFCSITE'
26.      && element.type !== 'IFCBUILDING'
27.      && element.type !== 'IFCSPACE'){
28.        buidingArray.push(elementAllProps);
29.      }
30.      // Call method recursively on child elements.
31.      await this.parseTree(element, buidingArray, storeyName);
32.    }
33.  }
34. }
```

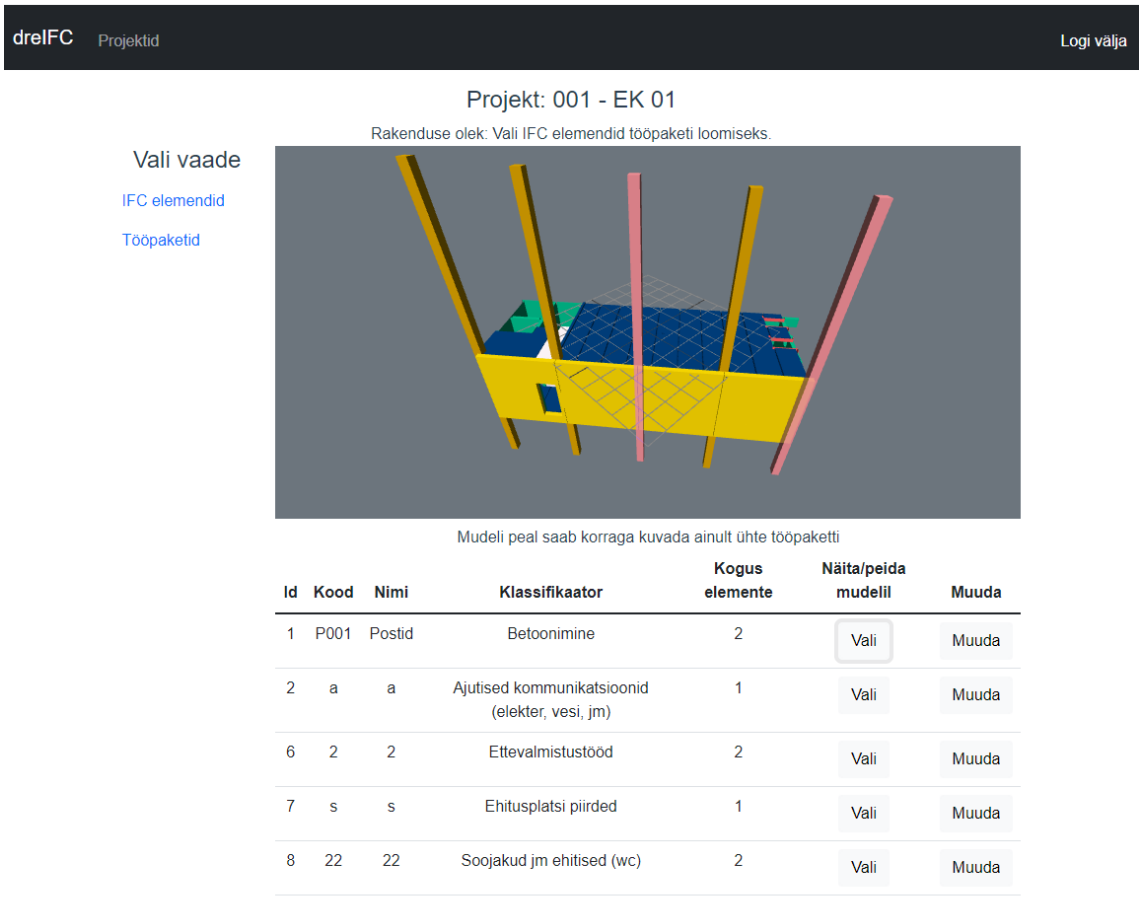
Joonis 17. Viewer.vue meetod parseTree.

Tööpaketi loomiseks on vaja märgistada vähemalt üks element. Elemente on võimalik valida nii tabelist kui ka 3D vaatest kahekordse hiire klikiga soovitud elemendil. Pärast märgistamist tuleb vajutada nuppu uue tööpaketi loomiseks ning avaneb tööpaketi loomise vaade, mis on esitatud joonisel 18. Tööpaketi loomiseks on vaja sisestada loodava paketi nimi ja kood ning valida sobiv klassifikaator. Võimalik on lisada ka kirjeldus, aga see pole kohustuslik.



Joonis 18. Tööpaketi loomise aken.

Loodud tööpakettide nägemiseks on vaja valida navigatsiooni menüüst tööpakettide vaade ning seejärel kuvatakse tööpakettide tabel 3D vaate alla nagu kuvatud joonisel 19. Tabelist on võimalik ka kuvada kõiki tööpaketis olevaid elemente 3D vaates.



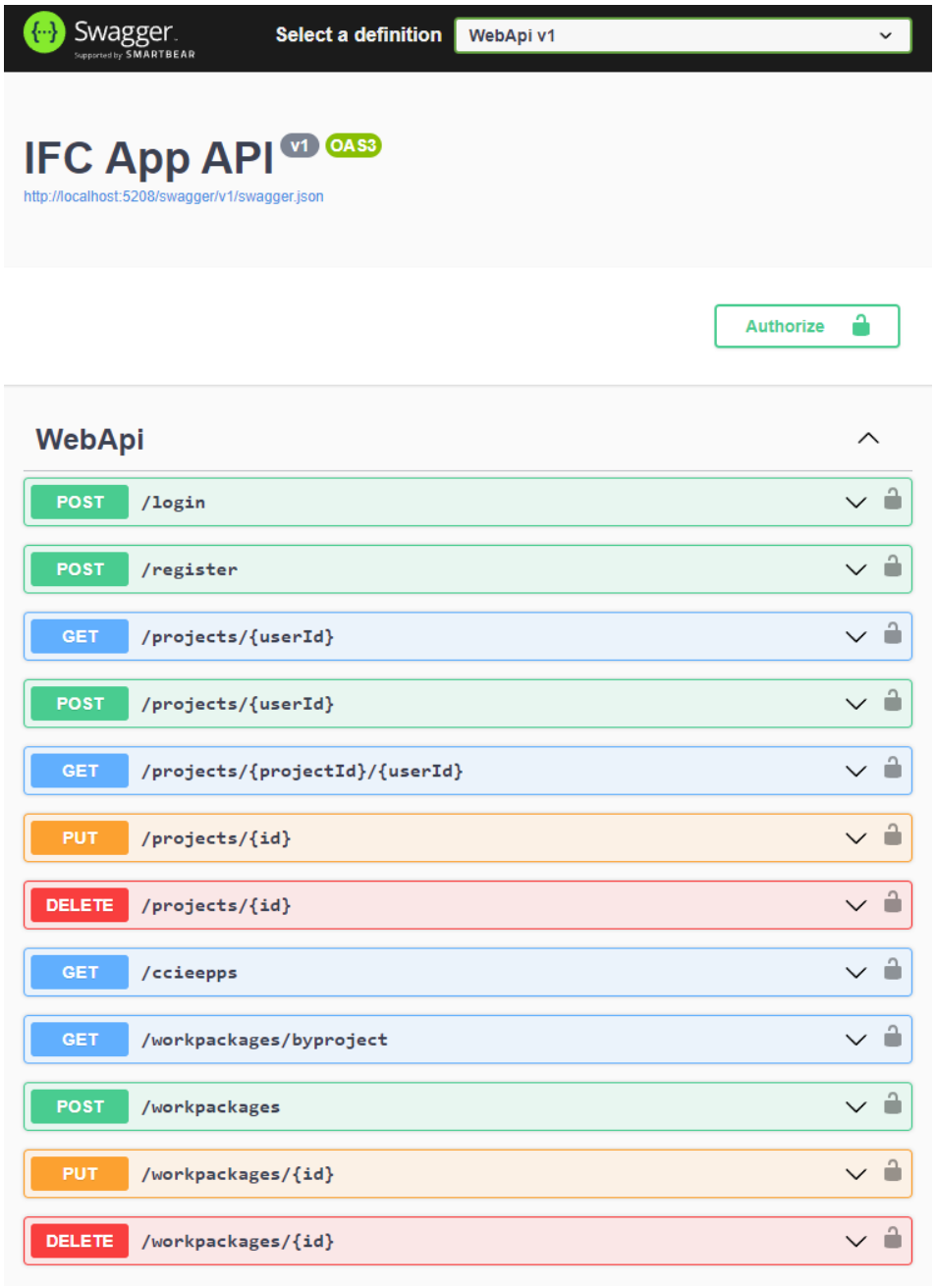
Joonis 19. Tööpakettide tabeli vaade.

3.4 Rakendusliides

Serveripoolse rakenduse eesmärk on REST (*Representational State Transfer*) API põhimõttel andmebaasiga suhtlemine. Failide struktuur on esitatud lisas 3. Klassid ja failid on loogiliselt eraldatud parema loetavuse mõttes, kuid ehitatud üles minimalistlikult ja kompaktselt.

3.4.1 Rakendusliidese arhitektuur

Eelnevalt valitud Minimal API raamistiku peamine loogika asub failis Program.cs, kus on nii seadistused ja sätted kui ka REST API ligipääsetavad meetodid. Joonisel 20 on kujutatud kõik klientrakenduse poolt kasutatavad meetodid.



Joonis 20. API kasutatavad meetodid.

API projekti salvestamise meetodi näide on esitatud joonisel 21 ning seal on näha, et tegemist on väga minimalistliku lähenemisega. Ei ole loodud eraldi loogika kihti andmebaasiga suhtlemiseks, kuna autor hindas selle ajakulu ebamõistlikuks käesoleva skoobi jaoks.

```
1. app.MapPost("/projects/{userId}", [Authorize(AuthenticationSchemes =
   JwtBearerDefaults.AuthenticationScheme)]async (string userId, Project
   project, AppDbContext dbContext) =>
2. {
3.     project.AppUserId = userId;
4.     dbContext.Projects.Add(project);
5.     await dbContext.SaveChangesAsync();
6.
7.     return Results.Created($"/projects/{project.ProjectId}", project);
8. });
```

Joonis 21. API projekti salvestamise meetod.

3.4.2 Rakendusliidese turvalisus

Rakenduse kasutamine anonüümselt on võimalik ainult sisse logimiseks või uue kasutaja registreerimiseks. Kogu järgnev tegevus on piiratud. Kasutaja andmemudel on loodud Microsoft AspNetCore Identity integreeritud kasutaja põhjal ning seda on kujutatud joonisel 22 [29].

```
1. using Microsoft.AspNetCore.Identity;
2. using System.ComponentModel.DataAnnotations;
3.
4. namespace WebApi.Models.Identity
5. {
6.     /// <summary>
7.     /// App user model.
8.     /// </summary>
9.     public class AppUser : IdentityUser
10.    {
11.        [MinLength(1)]
12.        [MaxLength(128)]
13.        public string? FirstName { get; set; }
14.
15.        [MinLength(1)]
16.        [MaxLength(128)]
17.        public string? LastName { get; set; }
18.
19.        public ICollection<Project>? Projects { get; set; }
20.    }
21. }
```

Joonis 22. Rakenduse kasutaja andmemudeli kood.

Kasutaja tuvastamiseks kasutatav JWT on avalikult kättesaadav standard, mis kirjeldab kompaktset viisi turvaliseks andmevahetuseks kahe osapoole vahel kasutades JSON (*JavaScript Object Notation*) tüüpi objekti. Seda infot on võimalik usaldada ja kontrollida, kuna see on krüpteeritud kasutades avaliku ja salajase võtmel põhinevat meetodit [6]. Joonisel 23 on esitatud rakenduses JWT loomiseks kasutatav koodi lõik.

```
22. public static string BuildToken(LoginDto dto, IConfiguration
    configuration)
23. {
24.   var claims = new List<Claim>
25.   {
26.           new Claim(ClaimTypes.Email, dto.Email!)
27.   };
28.
29.   var key = new
    SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["jwt:key"]))
    ;
30.   var creds = new SigningCredentials(key,
    SecurityAlgorithms.HmacSha256);
31.
32.   var expiration = DateTime.UtcNow.AddHours(3);
33.
34.   var token = new JwtSecurityToken(
35.           issuer: configuration["Jwt:Issuer"],
36.           audience: configuration["Jwt:Audience"],
37.           claims: claims,
38.           expires: expiration,
39.           signingCredentials: creds);
40.
41.   return new JwtSecurityTokenHandler().WriteToken(token);
42. }
```

Joonis 23. JWT loomine.

3.4.3 Näidisandmed

Rakenduse korrektseks töötamiseks on vajalik selle esmakordsel käivitamisel andmebaasi kirjutada klassifikaatorite andmed. Selleks on loodud abistav klass DataInitializers, mis kutsutakse välja, kui andmebaasis puuduvad klassifikaatorid. Näidisandmed on teisendatud klassifikatsioonisüsteemist CCI-EE tootmise osast ning salvestatud JSON kujul.

3.5 Testimine

Automaattestimist ei ole arendatud, kuna töö ajalisse skoopi ei mahtunud automaattestimise raamistike uurimine ning nende rakendamine. Autor kasutas kümmet erinevat ehitusprojekti faili peamise funktsionaalsuse manuaalseks testimiseks.

4 Hinnang loodud veebirakendusele

Loodud veebirakendust on võimalik hinnata saavutatud kasutatavuse ning tuleviku edasi arendamise potentsiaali järgi.

4.1 Saavutatud kasutatavus

Rakenduse ehitamise käigus said lahendatud kõik funktsionaalsed ja mittefunktsionaalsed nõuded, et tagada minimaalne kasutatavus. Arendusprotsessi käigus avastatud täiendavad parendamise meetmed rakenduse kvaliteedi tõstmiseks lahendatakse tulevase edasi arenduse käigus, et saavutada maksimaalne potentsiaal.

Rakenduses on võimalik luua uus kasutaja või logida sisse olemasoleva kasutajaga. Kasutaja saab hallata olemasolevaid ja luua uusi projekte. Projekti vaates on võimalik kuvada ehitusprojekti 3D mudel ning väljavõte selles olevatest mudeli elementidest. Elementidest on võimalik luua tööpakette ning neid kuvada 3D mudelil. Visuaalne disain on minimalistlik ning seda on võimalik edasi arendada.

Rakendusliides omab vajalikku loogikat turvaliseks suhtlemiseks kasutajaliidesega ja võimaldab andmevahetust andmebaasiga.

4.2 Võimalused edasiarenduseks

Edasi arendamise võimalused jagatakse järgenvates peatükkides kaheks. Esmalt antakse ülevaade konkreetse rakenduse puudujääkide parendamiseks ning seejärel suuremas skoobis veebikeskkonna loomisest selle ümber.

4.2.1 Rakenduse edasiarendus

Autor hindab esimeseks prioriteediks edasi arendamisel automaattestimise lisamist. Tuleb uurida erinevaid olemasolevaid testimise raamistikke ning valida kõige sobivamad, mis võimaldavad katta ja kontrollida maksimaalsel hulgal kogu rakenduse loogikat. Tõenäoliselt on vaja eraldi raamistikke kliendi ja serveri poolele, kuna need on ehitatud erinevate tehnoloogiate peale. Testide lisamine võimaldaks avastada olemasolevaid puudujääke ning vähendaks riski olemasoleva funktsionaalsuse lõhkumisele uue funktsionaalsuse lisamise käigus.

Järgnevalt tuleb rakendus üle viia HTTP pealt turvalise HTTPS protokolli peale, kuna toimub andmete vahetus ja hea tava näeb ette sellisel juhul andmevahetuse krüpteerimise. Selleks on vaja osta SSL sertifikaat või luua see tasuta kasutades näiteks Let's Encrypt keskkonda [30]. Seejärel on vajalik teha muudatusi rakenduses ja muuta virtuaalmasina peal veebiserveri seadistusi.

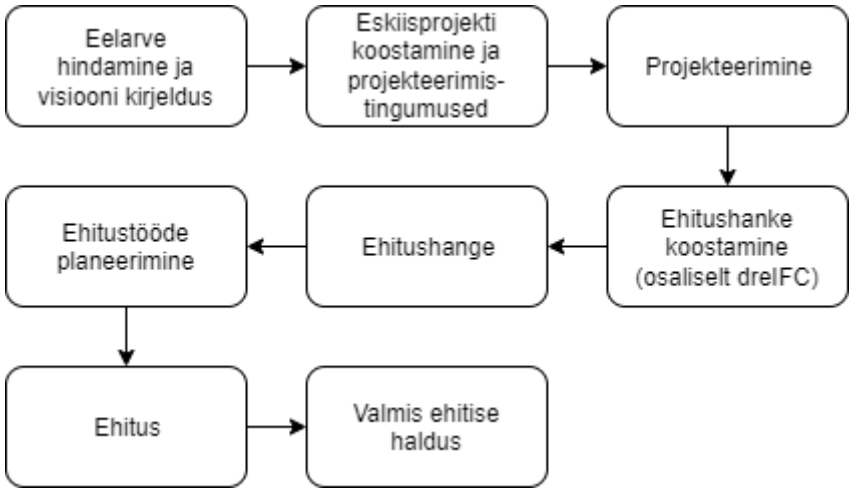
Selleks, et rakenduse potentsiaalset kasutajate arvu tõsta tuleb lisada kasutajaliidesele erinevate keelte tugi. Vue.js raamistikule on olemas erinevaid liideseid, mis seda võimaldavad. Suur osa rakendusest on vaja ümber kirjutada, et kasutajale kuvatav tekst ei oleks staatiliselt koodi sisse kirjutatud, vaid kasutaks muutujaid, millele vastavaid tõlkeid saaks hoida andmebaasis.

Tänapäeval on kasutaja mugavuse suurendamiseks kasutusel võimalused ennast registreerida teiste enamlevinud platvormi kasutajate kaudu näiteks Google või Facebook. Selleks tuleb rakendusse integreerida näiteks OAuth raamistik [31].

Käesoleva töö raames on tööpakettide loomiseks valitud CCI-EE standardi tootmise klassifikaatorid, kuid need ei kata kõiki töö liike. Standardis on käsitletud täiendavaid jaotusi ning rakendus võiks tulevikus võimaldada valida nende vahel või lisada kasutaja poolt loodud kohandatud struktuuri.

4.2.2 Platvormi ehitamine

Sissejuhatuses kirjeldatud suurema skoobi probleemi terviklikuks lahenduseks on üks võimalus luua kogu ehitusprotsessi haldamise platvorm ühtse veebikeskkonnana. See keskkond koosneks erinevatest rakendustest, mis loogilises järjekorras asetseks nii antud töös loodud rakenduse ees kui ka järel. Üks võimalus platvormi koostamiseks eraldi rakendustena on skeemina antud joonisel 24, kus käesolevas töös loodud rakendus on osa ehitushanke koostamise etapist.



Joonis 24. Platvormi skeem.

Joonisel 24 toodud skeem on üks võimalus, milliseid ehituse etappe võiks tulevikus platvorm katta. Enne selle arendamise algust tuleb läbi viia põhjalik analüüs ning seda vajadusel ümber teha. Sellist laadi lahendus võimaldab optimeerida kogu ehituse protsessi tõstes tõhusust ja kvaliteeti ning vähendada kulusid. Nõudlus sellisele lahendusele on olemas globaalsel tasandil ning sellel võiks olla äriline potentsiaal. Täpsema väärtuse hindamine vajaks eraldi ärilist analüüsi erinevate ehitusprojektide näitel. Digitaalehituse klastri juhatuse liikme Hendrik Park hinnangul kannab antud töö väga hästi edasi digitaalehituse klastri missiooni ja visiooni, milleks on läbi tehnoloogilise innovatsiooni tõsta sektoris usalduslikkust ja tootlikkust.

5 Kokkuvõte

Diplomitöö eesmärk oli luua veebirakendus ehitusprojekti ehitusmahtude töötlemiseks. Esimese sammuna oli vajalik projekti mudeli failist info kätte saamine ja selle kuvamine. Teiseks sammuks info töötlemine ja selle põhjal grupeerimine tööpakettideks. Kolmandaks sammuks loodud tööpakettide kuvamine.

Analüüsi osas sai paika pandud probleemi lahendamise skoop ning määratud nõuded. Lisaks uuriti erinevaid olemasolevaid tehnoloogiaid ja valiti välja sobivad raamistikud eesmärgi saavutamiseks. Rakenduse arendust kirjeldavas peatükis sai antud ülevaade arhitektuurist ning üldisest funktsionaalsest loogikast.

Tööd võib lugeda õnnestunuks, kuna loodud rakendus täidab kõiki kirjeldatud nõudeid ning on praktiliselt kasutatav. Loodud rakendusel on palju edasi arendamise võimalusi ning tugev äriline potentsiaal.

Kasutatud kirjandus

[1] Eesti Standardimis- ja Akrediteerimiskeskus MTÜ, „EVS 928:2016 Ehitusinformatsiooni modelleerimise (BIM) terminid,“ 2016.

[2] ET Infokeskuse AS, CCI-EE versiooni number 2022.01.0.1, Tallinn, 2022.

[3] P. Raspel. [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1.%20Loengumaterjalid/. [Kasutatud 26 03 2022].

[4] „Industry Foundation Classes (IFC) - An Introduction,“ [Võrgumaterjal]. Available: https://technical.buildingsmart.org/standards/ifc. [Kasutatud 22 03 2022].

[5] Douglas Crockford, [Võrgumaterjal]. Available: https://www.json.org/json-en.html. [Kasutatud 26 03 2022].

[6] auth0, [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 14 11 2022].

[7] Red Hat Inc, [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Kasutatud 16 11 2022].

[8] ssl.com, [Võrgumaterjal]. Available: https://www.ssl.com/faqs/faq-what-is-ssl/. [Kasutatud 12 11 2022].

[9] Riigi Kinnisvara AS, [Võrgumaterjal]. Available: https://nouded.rkas.ee/. [Kasutatud 13 02 2022].

[10] R. K. AS, „Ehitusinformatsiooni juhtimine (BIM),“ [Võrgumaterjal]. Available: https://www.rkas.ee/et/panus-uhiskonda/bim. [Kasutatud 22 03 2022].

[11] The Advanced Work Packaging Institute, „What is a work package?,“ [Võrgumaterjal]. Available: https://www.workpackaging.org/single-post/what-is-a-work-package. [Kasutatud 24 03 2022].

[12] ET Infokeskuse AS, [Võrgumaterjal]. Available: https://ehituskeskus.ee/cci/cci-ja-excel-tabelid/. [Kasutatud 26 03 2022].

[13] M. Randlepp. [Võrgumaterjal]. Available: https://www.veebimajutus.ee/blogi/ekspert-mis-on-persona-ning-miks-on-see-sulle-vaartuslik. [Kasutatud 13 02 2022].

[14] WHATWG (Apple, Google, Mozilla, Microsoft), [Võrgumaterjal]. Available: https://html.spec.whatwg.org/. [Kasutatud 15 11 2022].

[15] IFC.js, [Võrgumaterjal]. Available: https://ifcjs.github.io/info/. [Kasutatud 14 11 2022].

[16] xBIM team, [Võrgumaterjal]. Available: https://docs.xbim.net/XbimWebUI/. [Kasutatud 13 02 2022].

[17] xBIM team, [Võrgumaterjal]. Available: https://github.com/xBimTeam/. [Kasutatud 13 02 2022].

[18] xeolabs, [Võrgumaterjal]. Available: https://xeokit.io/. [Kasutatud 14 11 2022].

[19] IFC.js, [Võrgumaterjal]. Available: https://github.com/IFCjs. [Kasutatud 26 03 2022].

[20] xeolabs, [Võrgumaterjal]. Available: https://github.com/xeokit. [Kasutatud 26 03 2022].

[21] Stackoverflow, [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2021. [Kasutatud 26 03 2022].

[22] Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis?view=aspnetcore-6.0. [Kasutatud 26 03 2022].

[23] Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data. [Kasutatud 26 03 2022].

[24] MariaDB Tutorial, [Võrgumaterjal]. Available: https://www.mariadbtutorial.com/. [Kasutatud 26 03 2022].

[25] GitHub, [Võrgumaterjal]. Available: https://education.github.com/pack. [Kasutatud 12 11 2022].

[26] IFCjs, [Võrgumaterjal]. Available: https://github.com/IFCjs/examples/tree/main/simple-vue. [Kasutatud 29 10 2022].

[27] Vue.js, [Võrgumaterjal]. Available: https://vuex.vuejs.org/guide/. [Kasutatud 29 10 2022].

[28] Vue.js. [Võrgumaterjal]. Available: https://router.vuejs.org/. [Kasutatud 29 10 2022].

[29] Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio. [Kasutatud 15 11 2022].

[30] Let's Encrypt, [Võrgumaterjal]. Available: https://letsencrypt.org/. [Kasutatud 17 11 2022].

[31] A. Parecki. [Võrgumaterjal]. Available: https://oauth.net/2/. [Kasutatud 17 11 2022].

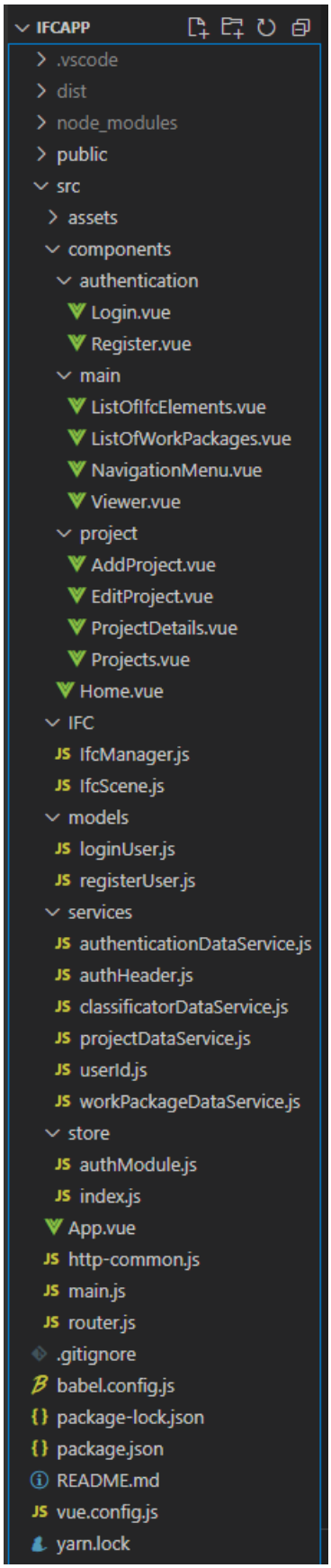
[32] „EVS 928:2016 „Ehitusinformatsiooni modelleerimise (BIM) terminid““.

Lisa 1 – Versioonihaldus

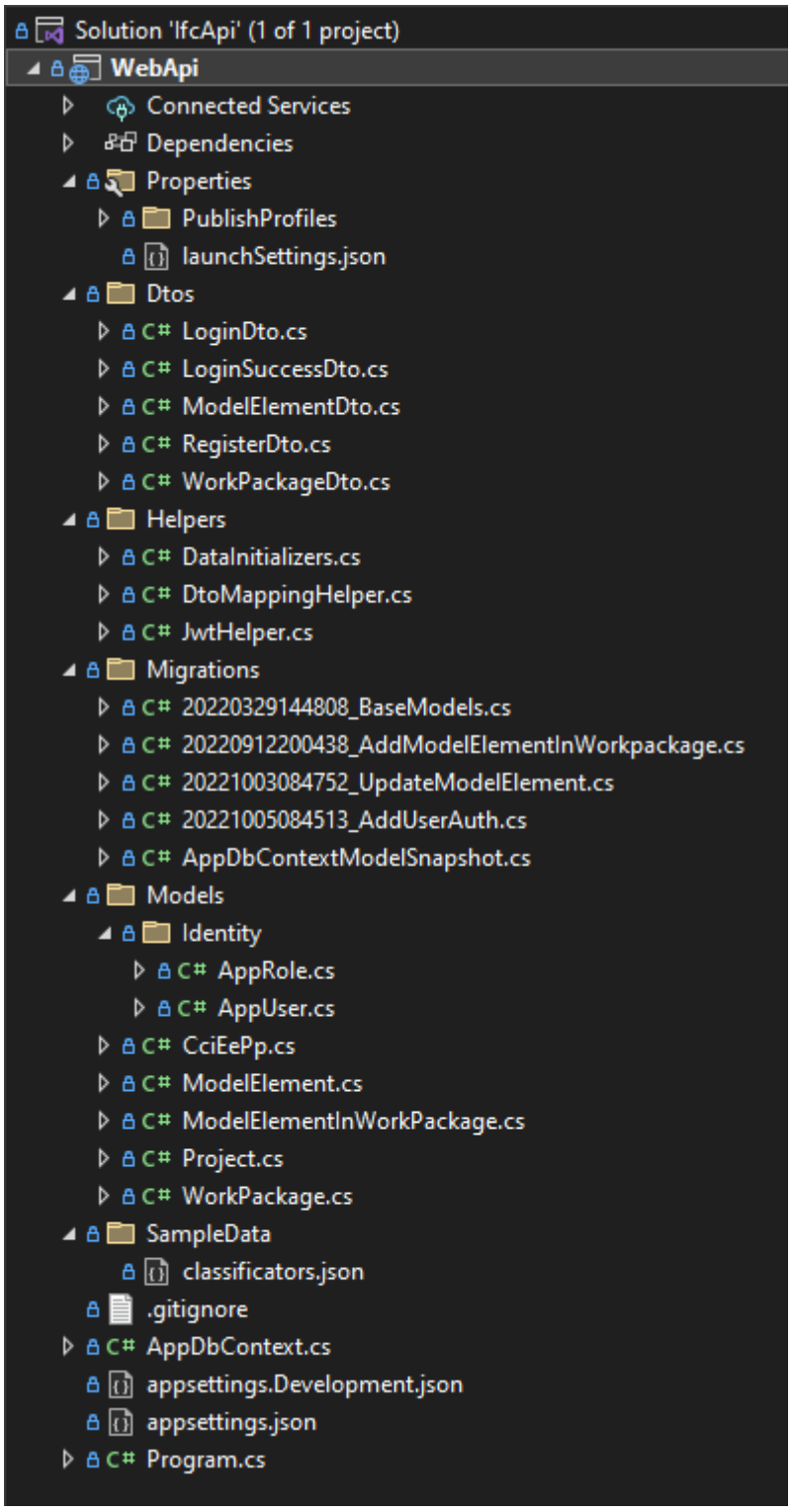
Kasutajaliides: https://github.com/andreivass/ifcapp/

Rakendusliides: https://github.com/andreivass/ifcapi/

Lisa 2 – Kasutajaliidese failid VS Code rakendusest.



Lisa 3 – Rakendusliidese failide puu Visual Studio rakendusest.



Lisa 4 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Andrei Vassiljev

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Töömahtude töötlemise rakendus ehitusprojektidele“, mille juhendaja on Meelis Antoi
   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.

02.01.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.