TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Toivo Sults 202847IADB

Kinnisvarahalduse eelarvestaja veebirakendus

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

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: Toivo Sults

02.01.2024

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on pakkuda välja tehnoloogiline lahendus ettevõtte äriprotsessi mugavamaks ja kasutajasõbralikumaks täitmiseks ning luua prototüüp, mille abil valideerida loodav rakendus ja kinnitada edasine arendusprotsess.

Töö käigus tutvub autor ettevõtte äriprotsessiga, et välja selgitada ja kirjeldada loodava rakenduse nõuded. Analüüsitakse erinevaid tehnoloogilisi võimalusi lahenduse saavutamiseks ning teostatakse arenduseks sobivate tehnoloogiate valik.

Kirjeldatud nõuete alusel, määratud skoobi ulatuses, luuakse ning testitakse rakenduse prototüüp. Lisaks kirjeldab autor võimalikke edasiarendusi, mis ei kajastu kirjeldatud nõuetes kuid, mis aitaksid kaasa rakenduse edasisele arengule tulevikus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 6 peatükki, 6 joonist, 3 tabelit.

Abstract

Property Management Estimator Web Application

The aim of this bachelor's thesis is to propose a technological solution to make the company's business process more convenient and user-friendly, and to create prototype of application to validate and confirm the further development process.

Within the thesis, the author familiarizes himself with the company's business process in order to find out and describe the requirements of the application to be created. Various technological possibilities for achieving a solution are analyzed and the selection of suitable technologies for development is carried out.

Based on the described requirements, within the specified scope of thesis, an application prototype is created and tested. In addition, the author describes possible future developments, which are not reflected in described requirements, but which would contribute to the further development of application in the future.

The thesis is in Estonian and contains 27 pages of text, 6 chapters, 6 figures, 3 tables.

Lühendite ja mõistete sõnastik

.docx	Faililaiend, mis tähistab Microsoft Wordi dokumendiformaati
.NET	Tarkvararaamistik, mis pakub arendajatele keskkonda tarkvaraarenduste loomiseks ja käitamiseks
API	*Application Programming Interface*, tarkvarakomponentide komplekt või liides, mis võimaldab ühel tarkvararakendusel suhelda teise rakendusega
Cascade delete	Andmebaasi kontseptsioon, mis viitab olukorrale, kus andmebaasisüsteem kustutab automaatselt seotud andmed peakirje kustutamisel
Code first	Tarkvaraarhitektuuri mõiste, mis viitab andmebaasi modelleerimisele ja loomisele, võttes aluseks programmeerimiskeele objektide määratlused
DbContext	Andmebaasiga suhtlemiseks mõeldud Entity Framework raamistiku osa ehk klass millest lähtutakse, et suhelda andmebaasiga
DOM	*Document Object Model*, programmeerimisliides, mis kujutab HTML või XML dokumente objektidena
EF	*Entity Framework*, objekt-suhte kaardistamise raamistik .NET rakenduste jaoks
Excel	Arvutustabelite loomise, redigeerimise ja analüüsimise tarkvara
GET	HTTP päringumeetod ressursside pärimiseks veebiserverilt
IDE	*Integrated Development Environment*, tarkvararakendus, mis hõlbustab tarkvara arendamise protsessi
Mapper	Tarkvara komponent, mis aitab kaardistada ühe kihi objekte teise kihi objektideks
mobiilID	Identifitseerimise ja autentimise meetod, mis võimaldab kasutajatel kasutada oma mobiiltelefoni digitaalse isikutunnistusena
MVC	*Model-View-Controller*, tarkvara arhitektuuriline muster, mis eraldab komponendid vastutusalade järgi
PDF	*Portable Document Format*, Adobe Systems poolt välja töötatud failivorming
POST	HTTP päringumeetod andmete saatmiseks serverile, et luua või värskendada ressurssi.

Responsiivne disain	Veebidisaini lähenemine, mille eesmärk on parema kasutajakogemuse tagamine erinevate seadmete ja platvormide kasutajatele
SmartID	Lahendus isikutuvastuseks, mille abil on võimalik siseneda e-teenustesse ja allkirjastada dokumente
Stack Overflow	Veebipõhine küsimuste ja vastuste platvorm, mis on koondanud laiaulatusliku programmeerimisalase teabekogu
URL	*Uniform resource locator*, interneti-aadress, võrguaadress
W3C	*World Wide Web Consortium*, rahvusvaheline organisatsioon, mis tegeleb erinevate tehnoloogiate standardiseerimisega
Word	Tekstitöötlusprogramm

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

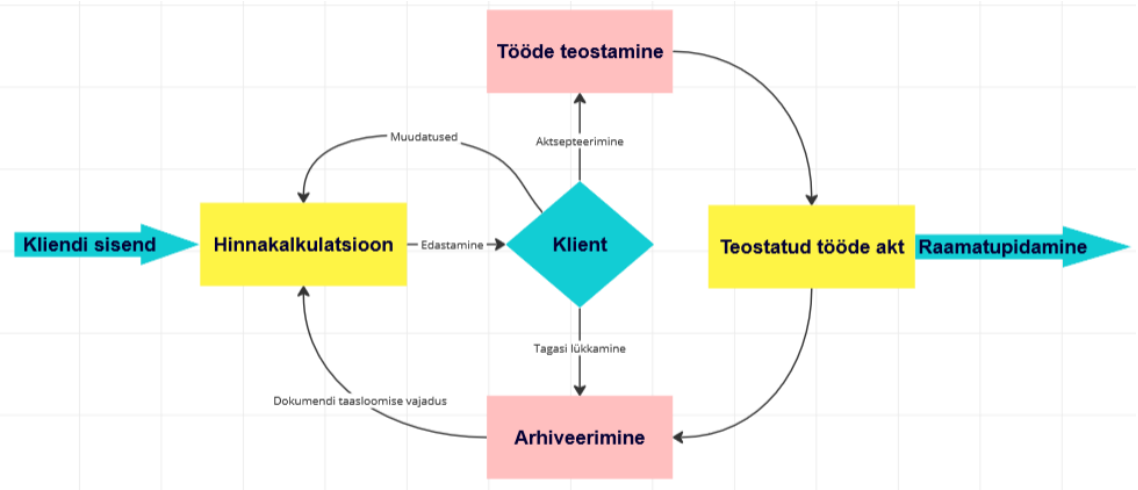
Ettevõte, mille juured ulatuvad eelmise sajandi teise poolde, on viimaste aastakümnete jooksul muuhulgas ühe ärisuunana tegelenud kinnisvara arendamisega. Tänase seisuga omab ettevõte peamiselt ärikinnisvara Eesti tuiksoonel – Harjumaa piirkonnas, mille rentimisega aktiivselt tegeletakse.

Rentnike arv on varieeruv, kuid peamiselt pikemaajaliste lepingutega ulatub see enam kui poolesaja stabiilse rentnikuni. Aeg ajalt rentnikud soovivad teostada oma rendipindadel ümberehitustöid ja tehnosüsteemide korrigeerimisi tulenevalt oma tootmise iseloomust. Sellega seonduvalt tuleb koostada klientidele (rentnikud) ümberehitustööde kalkulatsioone, neid klientidele esitada, pidada läbirääkimisi, teostada kalkulatsioonides muudatusi ja ümberkirjeldusi ning lõppfaasis, kokkulepitud tööde lõpetamisel, tuleb koostada teostatud tööde aktid, mis on rahalise arvelduse aluseks.

2 Hetkel rakendatud protsess

Ettevõtte protsessiga tutvumisel ja selle kaardistamisel on läbi viidud vabas vormis poolstruktureeritud intervjuu (Lisa 2) eelarvestaja rolli täitva isikuga [1] ning teostatud protsessi vaatlus. Intervjuu annab võimaluse suhelda vahetult protsessis osalejaga ning koguda vajadusel kiirelt täpsustavat lisainformatsiooni, tänu millele on võimalik paremini mõista ja tõlgendada intervjueeritava vastuseid nähes tema emotsioone kogemuste ja soovitud tulemuste kirjeldamisel.

Hinnapakkumiste koostamiseks ja teostatud tööde aktide vormistamiseks kasutatakse tekstitöötlusprogrammi ja tabelarvutuse rakendusi, millest lõpuks luuakse PDF (*Portable Document Format)* failivormingus dokument. Hetkel on ettevõttes kasutusel Microsofti Office kontoritarkvarapakett, millest kasutatakse Word (tekstitöötlusprogramm) ja Excel (tabelarvutus rakendus) rakendusi. Klientidega suhtlemine toimub e-kirjade teel.



Joonis 1. Äriprotsess.

2.1 Protsessi kirjeldus

Protsess algab hinnakalkulatsiooni koostamisega andmetabelina. Tööde loetelu (tööde kirjeldus, mahud, ühikuhinnad) sisestamine toimub käsitsi. Peale valmimist kopeeritakse kogu sisuline osa eelvormistatud, ettevõtte logode, kontaktide ja muu olulise infoga blanketti, mida hoitakse eraldi, eelvormistatud .docx ehk Microsoft Wordi

failivormingus. Sellest omakorda luuakse PDF failivormingus dokument, mida siis juba kliendile esitada.

Peale hinnakalkulatsiooni esitamist, rentnik, kas kinnitab selle või lükkab selle tagasi. Hinnakalkulatsiooni kinnitamisel rentniku või tema esindaja poolt tellitakse vajalikud materjalid tarnijatelt ja vajalikud tööd alltöövõtjatelt. Hinnakalkulatsiooni tagasilükkamisele järgneb üks kahest, selle täielikul tagasilükkamisel see arhiveeritakse tagasi lükatud hinnakalkulatsioonina või muudatussoovide korral asutakse teostama kalkulatsioonis vajalike muudatusi.

Peale tööde teostamist vormistatakse hinnakalkulatsioonidega sarnaselt andmetabelina paika pandud vormingus teostatud tööde aktid. Igakordselt on vajalik muuta tööde sisu, mis on üldjuhul võimalik kopeerida hinnakalkulatsioonist, kuid mitte alati. Käsitsi on vaja muuta ära kuupäevad, kontrollida ja muuta rentniku info – kontrollida rentniku aadress, registrinumber, kontaktandmed, tööde nimetused, volitatud vastuvõtja jne.

Edukalt läbitud protsessi korral edastatakse teostatud tööde akt raamatupidamisele, mis on aluseks arvete koostamisel.

2.2 Probleemi kirjeldus

Arvestades, et hinnakalkulatsioonide ja aktide koostamine toimub peamiselt käsitsi trükkides ning võib esineda hinnakalkulatsioone, mille ridade arv ületab poole tuhande piiri, siis on tegemist ajakuluka ja ühtlasi ka veaohtliku protsessiga, kus esineb võimalusi inimlike eksituste tekkimiseks. Suhtlus rentnike ja nende esindajatega toimub e-kirjade vahetamise teel. Pikemate suhtluste korral võib esineda võimalus, et kirjad jäävad tähelepanuta ning tööde kinnitamise või nende tagasilükkamise osas võib tekkida viivitusi. Samuti ei ole välistatud inimlikud eksimused või tehnilised probleemid, mille tõttu edastatakse rentnikule valed dokumendid. Tööde aktsepteerimisel vajalike aktide koostamisel sisuliselt kordub hinnakalkulatsioonide koostamisele sarnaselt manuaalne töö. Kuigi võib esineda võimalus, et tööde loetelu saab kopeerida hinnakalkulatsioonidelt, tuleb see kontrollida ning käsitsi lisada konkreetse projektiga seotud rentniku ehk tellija info.

Tagasilükatud pakkumiste korral need arhiveeritakse tagasilükatud hinnapakkumistena. Arhiiv on üles ehitatud kaustapuuna ühe isiku arvutis, mis tähendab, et ligipääs sellele

ilma vastava riistvarata ei ole võimalik, mistõttu on see ka üsna haavatav. Juhul, kui ühe konkreetse arvutiga midagi peaks juhtuma võib kogu arhiiv olla kaotatud. Arhiivis navigeerimine võib kujuneda üsna ajakulukaks kuna sellel puudub informatiivne otsimise võimalus ehk võimalus leida infot koostatud dokumentide siseselt kujul, mis annaks ilma suurema vaevata ülevaate näiteks mõne konkreetse rentniku teostatud töödest või tagasilükatud pakkumistest. Ette võib tulla olukord, kus tagasilükatud pakkumise leidmine on vajalik näiteks mõne aasta möödumisel, kuna rentnik võib mõnest juba kogu koostatud hinnakalkulatsiooniga tööst olla taas huvitatud.

2.3 Lõpplahenduse eesmärk

Probleemi kirjeldusest selgub, et hetkel kasutatava lahenduse juures kasutatakse palju käsitsi info sisestamist, mille tõttu on erinevaid veaohtlike olukordi ja võimalusi andmete kadumiseks. Rakenduse lõppeesmärk oleks eemaldada veaohtlikud olukorrad, mis tulenevad andmete käsitsi sisestamisest. Luua käideldav terviklik arhiiv, mis võimaldaks laialdasemalt ligipääsu andmetele ning mille abil oleks võimalik hõlpsamini navigeerida juba koostatud ning tagasi lükatud hinnakalkulatsioonide või siis juba aksepteeritud ning teostatud tööde aktidega seotud dokumentide vahel. Rakendus peaks muutma arhiivi paremini ja mugavalt käideldavamaks ehk kasutajasõbralikumaks. Rakendus võimaldaks pääseda volitatud isikutel sellele ligi erinevate seadmete ja internetiühenduste abil ka näiteks rentniku ruumides ringi jalutades ja töid kaardistades. Volitatud isikuks käesoleval juhul on isik, kellel on olemas kasutajakonto, millega pääseda ligi ettevõtte jaoks loodud rakenduse paigaldusele. Iga hinnakalkulatsiooniga seotud dokument võiks olla kajastatud ülevaatlikul vaatelehel.

Lisaks ettevõtte enda tööprotsessi mugavamaks ning vähem ajakulukaks muutmisele võiks loodav rakendus olla tulevikus tööriistaks ka rentnikele ja nende esindajatele, võimaldades selle abil, kas dokumentidega tutvuda, esitada ettepanekuid muudatusteks või aktsepteerida koostatud hinnakalkulatsioone.

Kokkuvõtvalt on loodava rakenduse eesmärk luua kasutajasõbralik keskkond, mille kasutamine on selge, lihtne ja mugav ning mille kasutamine ei eelda eriteadmisi või pikemalt juhenditega tutvumist.

2.4 Lõputöö skoop

Lõputöö skoopi kuulub sisendina esitatud äriprotsessi jaoks loodava rakenduse analüüs, selle kavandamine ning prototüübi loomine. Skoopi ei kuulu äriprotsessi analüüs ja selle muutmine.

Kui andmebaasi mudel ja tagarakendus luuakse arvestusega katta kogu kirjeldatud lähteülesanne siis prototüübi loomisel on peamine eesmärk anda võimalus ettevõttel teostada hinnakalkulatsioonide koostamist läbi veebirakenduse ning selle baasilt valideerida loodav rakendus. Prototüübi skoopi kuulub kinnisvara objektide loomine, nende osadeks jaotamine ja kirjeldamine, kontaktide lisamine ning ridadega hinnakalkulatsioonide loomine. Prototüübiga on üritatud edasi anda võimalikke lahendusi visuaalide näol, kuid tulenevalt piiratud ajaraamist ei ole selle eesmärk tagada läbimõeldud ülesehitusega kasutajasõbralikku keskkonda ning esmane eesmärk ei ole jälgida nõudeid ligipääsetavuse tagamiseks. Prototüüp on ühtlasi alus hilisema disaini loomisele, selle kasutamine annab selgema ülevaate ning võimaldab juba täpsemalt määrata milline võiks olla soovitud paigutus.

Käesoleva lõputöö skoopi ei kuulu täieliku kirjeldatud funktsionaalsuse loomine klientrakendusele, prototüübi skoop on ära määratud nõuete kirjeldamisel.

3 Rakenduse analüüs ja kavandamine

Käesoleva peatüki eesmärk on teha kindlaks ja kirjeldada loodava rakenduse nõuded.

3.1 Nõuete kirjeldus

Loodava veebirakenduse nõuete määramisel võeti aluseks rakenduse peamise kasutaja poolt intervjuu käigus kirjeldatud nõuded, mida vajadusel korrigeeriti vastavalt tehnilistele võimalustele.

Autorile teadaolevalt rakendatud äriprotsessiga ei kaasne täiendavaid nõudeid tulenevalt õigusaktidest või nõudeid, mis oleksid kirjeldatud ettevõtte sisemiste eeskirjade ja standarditega. Kuivõrd rakenduse loomisel on arvestatud intervjuu käigus selgunud nõuetega, siis nende realiseerimisel on üritatud läheneda võimalikult universaalselt.

Nõuded jagunevad funktsionaalseteks ja mittefunktsionaalseteks nõueteks. Funktsionaalsed nõuded on kirjeldatud kasutajalugudena, mis jagunevad omakorda kahte gruppi – nõuded, mis realiseeritakse prototüübis ning nõuded, mis ei kuulu prototüübi skoopi.

Ettevõtte poolelt ei soovitud eristada kasutajagruppe ja õigusi. Ühtlasi puudub esmase rakenduse raames vajadus luua kasutajakontosid ettevõtte klientidele, mis võib olla aga potentsiaalne võimalus edasiarenduste näol.

3.1.1 Funktsionaalsed nõuded

Järgnevalt on kirjeldatud funktsionaalsed nõuded, mis kuuluvad prototüübi skoopi:

- Eelarvestajana soovin lisada kinnisvara üksuseid;
- Eelarvestajana soovin teha kinnisvara osadeks jaotamisi;
- Eelarvestajana soovin näha lisatud kinnisvara üksuseid;
- Eelarvestajana soovin vajadusel muuta kinnisvara üksuseid ja nende osasid;
- Eelarvestajana soovin lisada hinnakalkulatsioone;
- Eelarvestajana soovin näha lisatud hinnakalkulatsioone;
- Eelarvestajana soovin teha muudatusi hinnakalkulatsioonides;
- Eelarvestajana soovin lisada kontaktisikuid ja kontakte;
- Eelarvestajana soovin näha lisatud kontaktisikuid ja kontakte;

- Eelarvestajana soovin teha muudatusi kontaktisiku andmetes ja kontaktides;
- Eelarvestajana soovin lisada ettevõtteid;
- Eelarvestajana soovin näha lisatud ettevõtteid;
- Eelarvestajana soovin teha muudatusi ettevõtete andmetes.

Järgnevalt on kirjeldatud funktsionaalsed nõuded, mis ei kuulu prototüübi skoopi:

- Eelarvestajana soovin määrata ettevõtte info ja sümboolika;
- Eelarvestajana soovin vajadusel ettevõtte infot ja sümboolikat muuta;
- Eelarvestajana soovin koostada hinnakalkulatsioonidest teostatud tööde akte;
- Eelarvestajana soovin teostada muudatusi kinnitamata teostatud tööde aktides;
- Eelarvestajana soovin alla laadida koostatud dokumente PDF kujul;
- Eelarvestajana soovin edastada hinnakalkulatsioone rentnikele;
- Eelarvestajana soovin edastada teostatud tööde akte rentnikele;
- Eelarvestajana soovin edastada kinnitatud teostatud tööde akte raamatupidamisele;
- Rentnikuna soovin näha minule koostatud hinnakalkulatsiooni;
- Rentnikuna soovin hinnakalkulatsiooni täielikult aktsepteerida;
- Rentnikuna soovin hinnakalkulatsiooni osaliselt aktsepteerida;
- Rentnikuna soovin hinnakalkulatsiooni ridade kohta lisada kommentaare;
- Rentnikuna soovin näha minule koostatud teostatud tööde akti;
- Rentnikuna soovin kinnitada minule koostatud tööde akti.

3.1.2 Mittefunktsionaalsed nõuded

- Rakendusel peab olema veebipõhine ligipääs;
- Eesrakendus peab olema responsiivse disainiga, kohandudes erinevate seadmetega, et tagada parem kasutajakogemus;
- Eesrakenduse loomisel peab olema arvestatud ligipääsetavuse standardite ja juhenditega [2].

3.2 Rakenduse tehnoloogiate valik

Loodava rakenduse jaoks vajaliku tehnilise lahenduse jaoks võib leida arendustehnoloogiaid programmeerimiskeelte ja nende raamistike ning andmebaasi lahenduste seast väga mitmeid. Iga aasta Stack Overflow, mis on populaarne veebipõhine

küsimuste ja vastuste platvorm, koondades laiaulatuslikku programmeerimisalast teabekogu, poolt korraldatavas uuringus, milles 2023 aastal osales pea 90 000 arendajat [3], selgus, et populaarsemate programmeerimiskeelte hulka kuuluvad Javascript, Python, Java ja C#. Nimetatud neljast on autoril vähesed kokkupuuted kõigiga. Populaarsemate veebiraamistike ja tehnoloogiate hulka kuuluvad Node.js, React, Angular ja ASP.NET Core.

Rakendust kasutava ettevõtte poolt konkreetseid nõudmisi esitatud ei ole, see tähendab, et puuduvad nõuded ühilduvuse osas. Oluline on, et rakendus oleks turvaline ja kättesaadav internetiühenduse abil. Arvestades limiteeritud aega ning tõsiasja, et väljatoodud lahendused on kõik loodava rakenduse teostamiseks kasutatavad, on valiku langetamisel üheks olulisemaks määrajaks eelnev kokkupuude erinevate programmeerimiskeelte ning raamistike kasutamisel või rakenduse loomiseks vajalikus ulatuses uue tehnoloogia õppimine ehk selle õppimiskeerukus.

3.2.1 Javascript

Javascript on laia kasutusalaga objektorienteeritud interpreteeritav programmeerimiskeel, mida peamiselt kasutatakse interaktiivsete veebisaitide loomisel kuid seda kasutavad ka mitmed veebilehitseja välised keskkonnad nagu Node.js, Apache CouchDB ja Adobe Acrobat. Javascripti lähtekoodi ei kompileerita masinkoodiks, seda loetakse rea põhiselt reaalajas. Levinud arusaama järgi on see peamiselt eesrakenduste loomiseks, kuid tänu sellistele tehnoloogiatele nagu näiteks Node.js, mis on avatud lähtekoodiga ning platvormide ülene [4], on seda võimalik samaväärselt kasutada tagarakenduse loomisel ehk muuhulgas näiteks veebiserverite loomiseks ja andmebaasipäringute teostamiseks. Javascripti põhitõed ei ole väga keerukad õppida, mille tõttu on see muutunud üheks enam levinud programmeerimiskeeleks ning on alustavate arendajate jaoks populaarne valik [5].

Järgnevalt on välja toodud mõned eelised, mida seostatakse Javascript programmeerimiskeelega [6]:

- Toetab kõiki modernseid veebilehitsejaid ja annab veebilehitsejate üleselt samaväärseid tulemusi. Võimaldab ehitada rakendust täies ulatuses, nii ees- kui tagarakenduse poolt;

- Põhitõdede õppimine ei ole keerukas, saadaval suurel hulgal materjale ja õppimisvõimalusi;
- Populaarne, millel on suur kogukond ning suurel hulgal avatud lähtekoodiga projekte, mida kasutada. Samuti globaalsete suurettevõtete toetus kogukonna arengule suuremate raamistike loomisega, näiteks Microsofti poolt Angular ja Facebooki loodud React.

Järgnevalt on välja toodud mõned puudused, mida seostatakse Javascript programmeerimiskeelega [6]:

- Suuremate rakenduste loomine võib olla keerukas;
- Kogu kood on nähtav kõigile;
- Javascripti vead võivad peatada kogu lehe laadimise.

3.2.2 Python

Python on interpreteeritav programmeerimiskeel, mis võimaldab erinevaid programmeerimisstiile nagu objektorienteeritud või funktsionaalset programmeerimist. Olemuselt peetakse seda lihtsaks keeleks, see on lihtsa ja selge süntaksiga, mille tõttu on see soovituslik valik programmeerimise õppimise alustamisel. Pythoni dünaamilised andmetüübid võimaldavad luua määramata tüübiga muutujaid, mis aitab suurendada võimalusi, kuid, mis teisest küljest on jällegi veaohtlik. Python on kasutatav väga laialdaselt, muu hulgas veebide loomiseks, skriptimiseks, serverites ja mängude loomisel. Python on samuti populaarne valik masinõppel [7].

Järgnevalt on välja toodud mõned eelised, mida seostatakse Python programmeerimiskeelega [8]:

- Mitmekülgne, kergesti õpitav, loetav ja kirjutatav;
- Avatud lähtekoodi ja suure aktiivse kogukonnaga, tänu sellele ka väga suur kolmanda osapoole moodulite ja teekide valik;
- Dünaamilised andmetüübid.

Järgnevalt on välja toodud mõned puudused, mida seostatakse Python programmeerimiskeelega [8]:

- Interpreteeritav programmeerimiskeel, aeglasem kui kompileeritavad keeled nagu C# või Java, mis tähendab, et selle kasutamine jõudlusmahukate ülesannete korral võib muutuda probleemseks;
- Python võib kasutada väga suurel hulgal mälu, eriti suurte andmekogude töötlemisel või keeruliste algoritmide kasutamise;
- Dünaamilised muutujad võivad käitusajal muutuda, mis muudab vigade püüdmise keeruliseks ning võib põhjustada programmivigu.

3.2.3 Java

Java on platvormist sõltumatu objektorienteeritud programmeerimiskeel, mida kasutatakse mobiili-, töölaua- ja veebirakenduste loomisel. Läbi aegade, oma eksisteerimise aja jooksul, on see olnud üheks populaarseimaks programmeerimiskeeleks. Baitkoodi kompileeritud Java lähtekoodi käivitatakse Java virtuaalmasina abil ning see on sõltumatu platvormist. Java virtuaalmasin sisaldab reaalajalist kompilaatorit, mis kompileerib baitkoodi masinkoodiks. See töötab otse protsessoril ilma vajaduseta baitkoodi jooksvalt interpreteerida [9].

Järgnevalt on välja toodud mõned eelised, mida seostatakse Java programmeerimiskeelega [10]:

- Töötab erinevatel platvormidel nagu Windows, Mac ja Linux. See on kiire, võimas ning turvaline;
- Selge struktuuriga objektorienteeritud programmeerimiskeel;
- Suure, aktiivselt panustava kogukonnaga, põhjalikult dokumenteeritud teekide kogu.

Järgnevalt on välja toodud mõned puudused, mida seostatakse Java programmeerimiskeelega [10]:

- Java õppimine võib olla keerukas, süntaks on paljusõnaline;
- Java kasutamine võib nõuda palju mäluressurssi ja on aeglasem kui näiteks C/C++ keel;

- Võib olla kulukas kasutada, tänu kõrgemate ressursinõuetele suureneb kulu riistavarale.

3.2.4 C#

C# on kaasaegne objektorienteeritud tüübikindel programmeerimiskeel, mis on mõeldud tarkvarakomponentide loomiseks hajusates keskkondades. See on loodud Microsofti poolt, et kasutada kõiki .NET tarkvararaamistiku võimalusi. C# on üldotstarbeline ning seda programmeerimiskeelt kasutatakse erinevate rakenduste loomisel nagu näiteks mobiilirakendused, töölauarakendused kui ka mängude loomisel. Üldiselt on see kergesti õpitav ning seda on kerge kasutada, selle tõttu on see väga paljude arendajate eelistuseks [11].

Järgnevalt on välja toodud mõned eelised, mida seostatakse C# programmeerimiskeelega [12]:

- Lihtne ja kergesti õpitav ning lai standardteekide valik;
- Platvormide ülene, võimaldab luua rakendusi nii Windowsi, Linuxi kui macOS tarbeks, samuti mobiili- ja veebirakendusi;
- Tugevalt tüübitud keel, mis tähendab, et andmetüüpide kontroll toimub kompileerimisel.

Järgnevalt on välja toodud mõned puudused, mida seostatakse C# programmeerimiskeelega [12]:

- C# ei ole väga paindlik, kuna sõltub suuresti .NET raamistikust;
- Arenenumate kontseptsioonide mõistmine C# keeles võib olla üsna keeruline;
- Vajalik koodi kompileerimine igakordsete muudatuste korral, mis võivad põhjustada vigu ja probleeme.

Tabel 1. Programmeerimiskeelte võrdlus.

Programmeerimiskeel	**Kogemused**	**Õppimiskeerukus**	**Kogukonna toetus**
C#	Hea	Keskmine [12]	Suur [12]
Java	Nõrk	Keskmine [10]	Suur [10]
Python	Nõrk	Madal [8]	Väga suur [8]
Javascript	Minimaalne	Keskmine [6]	Väga suur [6]

3.2.5 React

React on Facebooki poolt välja töötatud avatud lähtekoodiga Javascripti raamistik. React on mõeldud lihtsustamaks dünaamiliste veebirakenduste komponentide loomist ja nende haldamist. Kuna see kasutab virtuaalset DOM-i (*Document Object Model*) vaid muudetud osade värskendamiseks, siis aitab see parandada jõudlust. See võimaldab nii veebi- kui mobiilirakenduste loomist. React on eriti populaarne ühelehe-rakenduste loomisel, milles kasutajaliidese muutumine dünaamiliselt on seotud kasutaja tegevusega. Võrreldes teiste olemasolevate tehnoloogiatega on React küllaltki uus [13].

Järgnevalt on välja toodud mõned eelised, mida seostatakse React raamistikuga [14]:

- Parandab jõudlust tänu virtuaalse DOM-i kasutamisele;
- Seda on üsna lihtne õppida ja kerge kasutada;
- Reactil on lai kogukond ja palju kolmanda osapoole teeke ja komponente.

Järgnevalt on välja toodud mõned puudused, mida seostatakse React raamistikuga [15]:

- Sõltuvus kolmandate osapoolte teekidest;
- Algseadistuste tegemine võib olla keerukas.

3.2.6 Angular

Angular on Google poolt loodud ja hallatav avatud lähtekoodiga veebirakenduste raamistik, mis on loodud dünaamiliste ühelehe veebirakenduste ehitamiseks, kus sarnaselt Reactile kasutajaliides muutub ilma lehte uuesti laadimata. Angulari rakendused koosnevad taaskasutatavatest ja iseseisvatest komponentidest. Tegemist on ühe populaarseima veebirakenduste raamistikuga, mida kasutatakse laialdaselt suurte ja keerukate projektide arendamisel. Nagu paljude populaarsemate raamistike puhul on ka sellel lai kogukond ja ökosüsteem [16].

Järgnevalt on välja toodud mõned eelised, mida seostatakse Angular raamistikuga [17]:

- Suure kogukonna ja ökosüsteemiga;
- Tugevalt tüübitud, see on kirjutatud Typescriptis. Tänu tüübikontrollile aitab ennetada vigu arendusetapis;
- Saadaval on palju kolmanda osapoole mooduleid ja saadaval on hästi koostatud dokumentatsioon.

Järgnevalt on välja toodud mõned puudused, mida seostatakse Angular raamistikuga [17]:

- Angulari õppimine võib olla üsna keerukas, eriti algajatel;
- Suurema mahukusega failid, mis suurte rakenduste korral võivad põhjustada probleeme jõudlusega.

3.2.7 Node.js

Stack Overflow poolt 2023 aasta esimeses pooles läbi viidud uuringu [3] põhjal on tegemist populaarseima raamistikuga arendajate seas. Node.js on avatud lähtekoodiga, serveripoolne Javascripti raamistik, kuid seda on võimalik kasutada ka klientrakendustes. Node.js on mitmekülgne ja kasutusel laialdaselt alates väiksematest veebilehtedest kuni keerukate rakendusteni [18].

Järgnevalt on välja toodud mõned eelised, mida seostatakse Node.js raamistikuga [19]:

- Skaleeritav ja kõrge jõudlusega;
- Node.js kasutamine on üsna kergelt õpitav;
- Mahuka ja aktiivse kogukonna toega.

Järgnevalt on välja toodud mõned puudused, mida seostatakse Node.js raamistikuga [19]:

- Piiratud teekide tugi;
- Ebastabiilne API liides;
- Asünkroonne programmeerimismudel.

3.2.8 ASP.NET Core

ASP.NET Core on avatud lähtekoodiga ristplatvormiline kõrge jõudlusega raamistik, mida kasutatakse veebirakenduste arendamiseks. See on täiustatud ja kaasajastatud versioon ASP.NET raamistikust. ASP.NET Core on disainitud kiiruse, modulaarsuse ja kerge kasutuse saavutamiseks erinevates operatsioonisüsteemides. Tänu asünkroonsetele programmeerimismustritele aitab ASP.NET Core suurendada rakenduse jõudlust. Tänu MVC arhitektuurimustri kasutamisele, mis eraldab rakenduse loogika, andmete kihid ja kasutajaliidese, muudab see koodi organiseerimise lihtsaks [20].

Järgnevalt on välja toodud mõned eelised, mida seostatakse ASP.NET Core raamistikuga [21]:

- Väga kõrge jõudlusega, ristplatvormiline raamistik;
- MVC arhitektuur;
- Kiire arendusaeg.

Järgnevalt on välja toodud mõned puudused, mida seostatakse ASP.NET Core raamistikuga [21]:

- Õppimiskeerukus ja vähene kogukond, tegemist on üsna värske raamistikuga;
- Puudub universaalne lahendus veebirakenduse majutamisel, valida tuleb konkreetsete nõudmiste baasilt.

Tabel 2. Veebiraamistik/ -tehnoloogia võrdlus.

Veebiraamistik/ -tehnoloogia	**Kogemused**	**Õppimiskeerukus**
React	Nõrk	Madal [14]
ASP.NET Core	Hea	Keskmine [21]
Node.js	Puudub	Keskmine [22]
Angular	Puudub	Kõrge [17]

Rakenduse loomisel eraldatakse ees- ja tagarakendus, et muuta need teineteisest sõltumatuks ning võimaldada vajaduse korral tulevikus ühe või teise tehnoloogia välja vahetamist või muutmist tulenevalt võimalikest muutuvatest ärilistest vajadustest. Tagarakendus teostatakse kasutades C# keelt ning ASP.NET Core raamistiku võimalusi, mille kombineerimisel saavutab autor suurima võimaliku kogemustebaasiga aluse. Eesrakendus luuakse React raamistiku abil, milles autoril küll kogemused on vähesed, kuid tänu dokumentatsioonile ja õppematerjalidele vajaliku rakenduse loomiseks piisavalt kiiresti õpitav. Viimane väide on autori subjektiivne hinnang. React raamistikule on plaanis integreerida Bootstrap, mille abil on võimalik luua responsiivse disainiga veebileht. Bootstrap on vabavaraline CSS raamistik, mis sisaldab endas Javascripti pluginaid, HTML- ja CSS-kujundusmalle [23]. Kuigi puudub nõudmine, siis eesrakenduse hilisemate arendustööde käigus jälgitakse tulevikule mõeldes W3C (*World Wide Web Consortium*) poolt kirja pandud standardeid ja juhiseid [2], et tagada selle ligipääsetavus.

Arendustööriistadena on kasutatud koodiredaktoreid, JetBrains Rider [24] tagarakenduse loomiseks ning Visual Studio Code [25] eesrakenduse loomiseks. JetBrains Rideri, tagarakenduse loomiseks, on autor valinud põhjendusel, et see toetab C# programmeerimiskeelt ja sellel on täielik .NET Core tugi. JetBrains Rider võimaldab teostada testimist otse IDE-st (tarkvararakendus, mis aitab hõlbustada arendusprotsessi) ning see pakub koodianalüüsi ja veateadete esitamist, mis aitab ennetada vigu ning parandada programmikoodi kvaliteeti. Visual Studio Code valikul on lähtutud IDE ühilduvusest valitud tehnoloogiaga. Sellel on mugav kasutajaliides ning laialdane laienduste valik, mis hõlbustab arendusprotsessi. Lisaks on see tasuta kättesaadav kõigi jaoks. Üldisemalt on arendustööriistade valikul olnud määrav, et need toetaksid võimalikult laialdaselt kiiret prototüübi valmimist.

3.2.9 Andmebaasi valik

Tulenevalt tagarakenduse jaoks valitud tehnoloogiast võib leida, et tänu Entity Framework raamistikule on toetatud erinev valik relatsioonilisi andmebaase, mille vahel on võimalik valik teha [26]. Kuivõrd andmebaaside valikuid on erinevaid, siis käesoleva rakenduse mõistes ei ole ette näha suuri mahte ning pidevat suurt koormust andmebaasile, mille tõttu võib pidada võimalusi üsna samaväärseteks.

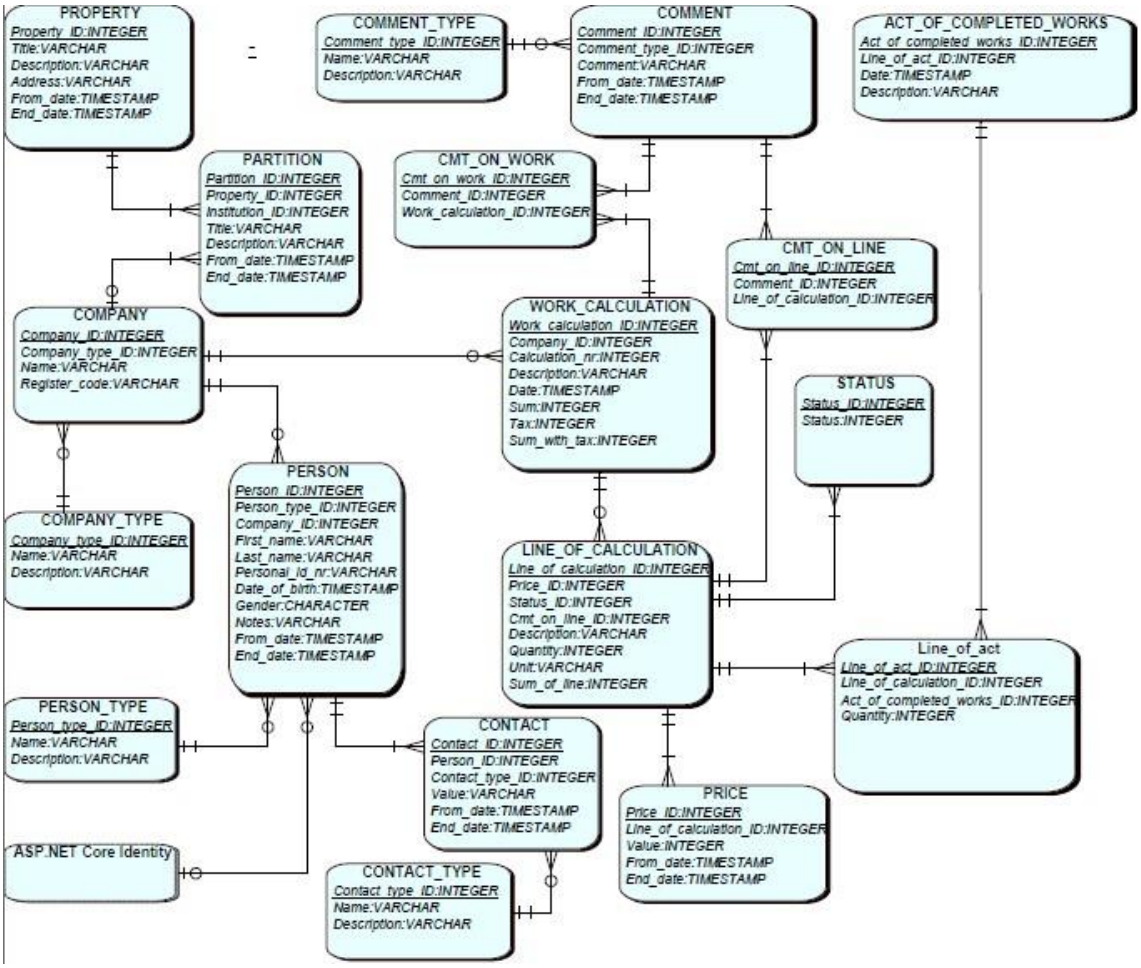
Autor on otsustanud teha valiku PostgreSQL kasuks. PostgreSQL on valitud, kuna see on tasuta kättesaadav, avatud lähtekoodiga ning laialdase funktsionaalsusega. Valiku langetamisel on autor ühtlasi pidanud oluliseks andmebaasi skaleeritavust võimalike edasiarenduste korral rakenduse elukaare jooksul.

4 Lahenduse teostus

Peatükk kirjeldab projekti prototüübi loomist.

4.1 Andmebaas

Andmebaasi mudeli loomisel on võetud arvesse esialgse intervjuu käigus selgunud tööprotsessi osasid ja nende komponente. Andmebaasi mudel on esitatud kontrollimisele ka ettevõtte esindajale ja vastavalt esitatud lisainformatsioonile ning tehnilistele võimalustele on teostatud kohendamisi, mille tulemusel on valminud esmane vajalik andmebaasi mudel.



Joonis 2. Olemi-suhte diagramm.

4.1.1 Olemite semantika

Järgnevalt kirjeldatakse loodud andmebaasiprojekti olemite semantika.

Tabel 3. Andmebaasiprojekti olemite semantika.

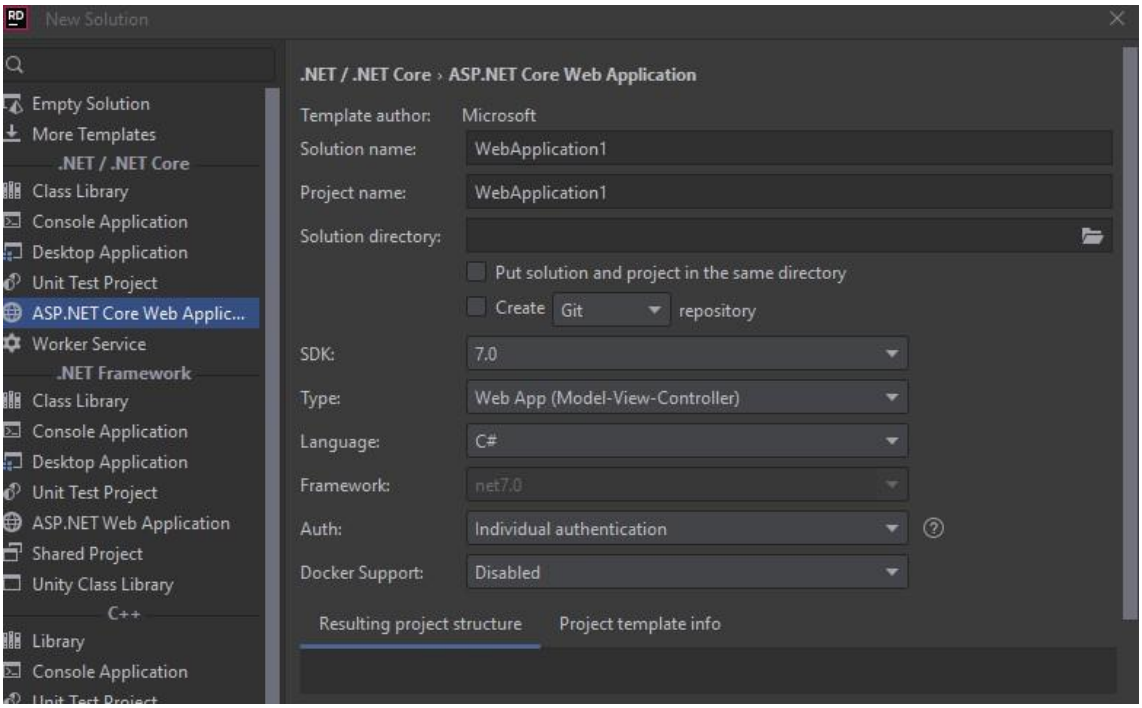
Tabeli nimi	**Semantika**
PROPERTY	Tabelis hoitakse andmeid kinnisvara kohta.
COMPANY	Tabelis hoitakse andmeid ettevõtete kohta.
COMPANY_TYPE	Tabelis hoitakse andmeid ettevõtte tüüpide kohta.
PERSON_TYPE	Tabelis hoitakse andmeid isikutüüide kohta.
PARTITION	Tabelis hoitakse andmeid jaotatud kinnisvara osade kohta.
PERSON	Tabelis hoitakse andmeid isikute kohta.
COMMENT	Tabelis hoitakse andmeid kalkulatsioonide ja aktidega seotud kommentaaride kohta.
COMMENT_TYPE	Tabelis hoitakse andmeid kommentaari tüüpide kohta.
ACT_OF_COMPLETED_WORKS	Tabelis hoitakse andmeid teostatud tööde aktide kohta.
CMT_ON_WORK	Tabelis hoitakse andmeid teostatud tööde aktide ja kommentaaride seoste kohta.
CMT_ON_LINE	Tabelis hoitakse andmeid ridade ja kommentaaride seoste kohta.
STATUS	Tabelis hoitakse andmeid dokumentide staatuste kohta.
LINE_OF_ACT	Tabelis hoitakse andmeid teostatud tööde aktide ja hinnakalkulatsioonil loodud ridade seoste kohta.
PRICE	Tabelis hoitakse andmeid rea hindade kohta.
CONTACT	Tabelis hoitakse andmeid kontaktide kohta.
CONTACT_TYPE	Tabelis hoitakse andmeid kontaktide tüüpide kohta.
LINE_OF_CALCULATION	Tabelis hoitakse andmeid kalkulatsiooni ridade kohta.
WORK_CALCULATION	Tabelis hoitakse andmeid kalkulatsioonide kohta.

4.2 Tagarakendus

Tagarakenduse loomisel on kasutatud kihilist arhitektuuri, milles on eraldatud ärimudeli kiht, andmeedastusloogika, äriloogika ja avalik API andmeedastus. Tänu kihilisele arhitektuurile on võimalik selle erinevaid kihte välja vahetada ilma teisi kihte mõjutamata, see võimaldab kihtide tasemel aidata kaasa skaleeruvusele ning aitab hõlbustada testimist. Projekti struktuur on selgelt määratletud, mistõttu see on kergemini mõistetav ja dokumenteeritav [27]. Tagarakenduse projekt koosneb omakorda üheksateistkümnest väiksemast projektist, kasutatud on mitmeid erinevaid liideseid ja abstraktseid klasse kindlustamaks andmete terviklikkus. Rakendus käivitub WebApp projektist.

Andmete kihtide vaheliseks tõlgendamiseks on loodud *mapper*id, mis aitavad mudelite baasil tõlgendada andmeid ühest kihist teise. Rakenduse loomise raames on kasutatud kihtide vahel suhteliselt samaväärseid andmeid, see tähendab, et peamiselt on olemite ülesehitus kihtide vahel samaväärne.

Projekti on alustatud ASP.NET Core Web Application loomisega, mis tänu kasutatud JetBrains Rider tarkvarale kasutajaliidesele saavutatav üsna kergelt.



Joonis 3. JetBrains Rider versioon 2022.3 ASP.NET Core Web Application projekti loomise vaade.

Kogu suhtlus ees- ja tagarakenduse vahel toimub läbi API otspunktide.

Projekti järgnevas faasis on loodud andmebaasi mudelile vastav olemite ülesehitus koodi põhiselt. Rakenduse andmebaas on loodud pidades silmas *code first* lähenemist ehk andmebaas on üles ehitatud olemite klassidena, mille alusel hilisemalt andmebaasi migratsioon on loodud.

```
using System.ComponentModel.DataAnnotations;
using Domain.Base;
namespace Domain.App;
public class Property : DomainEntityId
{
	[MinLength(1)]
	[MaxLength(64)]
	public string Title { get; set; } = default!;
	[MinLength(1)]
	[MaxLength(256)]
	public string? Description { get; set; } = default!;
	[MinLength(1)]
	[MaxLength(128)]
	public string? Address { get; set; } = default!;
	public DateTime FromDate { get; set; }
	public DateTime? EndDate { get; set; }
	public ICollection<Partition>? Partitions { get; set; }
}
```

Joonis 4. Olemi Property klassi koodinäide.

Andmebaasi tunnus id jaoks on loodud ja kasutatud abstraktset DomanEntityId klassi, mis annab kõikidele seda klassi laiendavatele alamklassidele ühise omaduse identifikaatori näol.

Ühendus andmebaasiga on loodud kasutades DbContext klassi, mis on osa EF raamistikust. Loodud prototüübi peal kasutatakse andmete kustutamisel *cascade delete* ehk andmebaasist kustutamisel kustutatakse kõik selle väljaga seotud teised andmed.

Peale esmaste mudelite loomist on loodud iga olemi jaoks Controller klassid, mille abil on loodud võimekus teostada esmane testimine ning mille tulemusel on võimalik veenduda mudeli toimimises. Lisaks käsitsi testimisele on läbi viidud integratsioonitestimine, mille abil on kontrollitud iga ühenduspunkti toimimine HTTP päringumeetodite GET ja POST abil. GET päringu puhul on kontrollitud andmete edastamine ning POST päringus on kontrollitud süsteemi võimekus andmeid vastu võtta ja neid töödelda ehk teostatud uue objekti lisamine. Testimise läbi viimiseks on loodud uus projekt tests, mis sisaldab testi iga ühenduspunkti kohta.

Kuigi esmajoones puuduvad erinevused äriloogika ning domeenimudeli vahelisel kihil, siis on loodud võimekus äriloogika vahekihi lisamiseks. Mõningane erinevus mudelite vahel tekib avalikus, API poolt opereeritavas kihis. Näiteks on mõnel juhul välja jäetud eesrakenduse jaoks ebavajalikud parameetrid nagu näiteks loomise kuupäev.

4.2.1 API otspunktid

Eesrakendust teenindab 17 erinevat API ühenduspunkti. Üks ühenduspunkt kasutajakonto info jaoks ning 16 rakenduse funktsioneerimiseks mõeldud API ühenduspunkti.

Kuna rakendus on loodud esmajoones prototüübina ja ei ole mõeldud täieulatusliku äriprotsessi teostamiseks, siis võib loodud API ühenduspunkte pidada rahuldavaks ja eesmärgipäraseks. Prototüübi loomisel API ühenduspunktide poolt tagastatud info on üsna samaväärne loodud domeenimudeliga, mis tähendab, et erispetsiifilisi kohandamisi prototüübi loomisel teostatud ei ole. Siiski on API ühenduspunktide puhul kasutusel avalikud andmeedastusobjektid, mis on saadud domeenimudelite kaardistamisel äriloogilisele mudelile ning omakorda nende kaardistamisel API poolt avalikeks mudeliteks. Sarnaselt, vastupidises suunas, toimub ka andmeedastusobjektide vastuvõtmine ning nende vastandamine domeeniobjektideks.

Kõik API ühendus otspunktid on turvatud autoriseerimise nõudega ehk kõikide rakendusloogiliste otspunktide toimimiseks on vajalik kasutaja autoriseerimine kasutajakontole logimisega.

API ühenduspunktid on dokumenteeritud Swagger abil, mis on avatud lähtekoodiga formaat mille abil kirjeldada ja dokumenteerida API ühenduspunkte [28], nendega on võimalik tutvuda käivitades tagarakenduse ning liikudes laiendusele /swagger.

4.3 Eesrakendus

Eesrakendus on teostatud kasutades React raamistiku koos Bootstrap CSS raamistikuga. Selle arendusel, sarnaselt tagarakendusega on kasutatud kihilist arhitektuuri.

Joonis 5. Eesrakenduse struktuur.

- „assets“ kaust sisaldab projektiga seotud meediat nagu pildid ja videod.
- „components“ kaust on mõeldud rakenduse komponentide jaoks, see sisaldab lehe päist ja jalust, samuti kliendi tuvastamiseks vajalikku IdentityHeaderit, mis tegeleb JWT päringutega ja vastustega.
- „css“ kaustast leiab eesrakendusega seotud CSS failid.
- „domain“ kaust sisaldab liideseid, mis kirjeldavad olemite põhjal eesrakenduse objekte.
- „dto“ kaust sisaldab andmeedastusobjektide liideseid.
- „routes“ kaust sisaldab eesrakenduse sisufaile nagu näiteks avalehe vaade, veateavitus leht ja ka objektide vaateid, näiteks kinnisvara, mis annab ülevaate kõikidest objektidest, uue kinnisvara lisamine, mis sisaldab kinnisvara lisamiseks vormi.
- „services“ kaust sisaldab iga objekti jaoks API andmevahetust.

Eesrakenduses on loodud iga olemi kohta liides, mis kirjeldab klasside vajalikud omadused, mis seda liidest implementeerivad. Sarnaselt tagarakendusele, laiendavad

need kõik IBaseEntity liidest, mis määrab igale alanevale liidesele identifikaatori omaduse.

Päringute tegemiseks on loodud akbstraktne BaseService klass, mille abil on määratud kõikidele päringutele ühine baas URL ehk võrguaadress ja muud vajalikud seaded ning iga ühenduspunkti tarbeks oma klass, mis tegeleb konkreetse API ühenduspunktiga suhtlemisel.

Päringute ülesehitamiseks on vahekihina loodud BaseEntityService klass, mis vastavalt päringule formuleerib vajaliku URL aadressi, mille poole pöördutakse.

Vajalike olemite ulatuses on loodud funktsionaalsed komponendid ehk komponentide vaated, mis tegelevad andmete esitamisega. Iga vajaliku olemi objekti kohta on lisatud kolm vaate komponenti – kõikide objektide kuvamine, uue lisamine ja ühe objekti vaade. Nende kuvamine sõltub kasutaja poolt navigeeritavast marsruudist, mis on määratud rakenduse sisenemispunktis, index.tsx failis.

4.4 Prototüübi rakendamine

Esmase kasutustestimise teostamiseks, ees- ja tagarakenduse prototüübi valmimise järgselt, on autor teinud veebirakenduse kättesaadavaks Azure pilveteenuses, mis on Microsofti poolt pakutav pilveteenuste kogum [29].

Nii ees- kui tagarakendus on konteineriseeritud Dockeri abil [30]. Selleks on loodud mõlemast konteinerpildid ja laetud need Docker Hubi, mis on avalik repositoorium konteinerpiltidele ning seotud need loodud Azure teenustega, et piltide uuendamisel teostataks automaatselt ka nende laadimine pilveteenusesse. Pilveteenuses opereeriva prototüübi andmebaasi tarbeks on loodud samas keskkonnas uus andmebaasi ressurss ning seotud see tagarakendusega.

Prototüübi ülesseadmise järgselt on autor veendunud rakendustevahelise andmevahetuse toimimises ning edastanud selle ettevõtte esindajale esmasele kasutustestimisele.

Lõputöö raames loodud ees- ja tagarakenduse kood on tehtud kättesaadavaks versioonihalduskeskkonnas GitHub ja nende seotud lingid on saadaval lõputöö lisades (Lisa 3).

5 Tulemuste analüüs

Käesoleva peatüki eesmärk on anda hinnang loodud lahendusele ning kirjeldada rakenduse võimalikud edasiarendused, mis ei ole määratud tänase äriprotsessi jaoks loodava rakenduse analüüsis.

5.1 Hinnang loodud lahendusele

Lõputöö raames valminud prototüüpi võib pidada eesmärgipäraseks, kuivõrd autor hindas prototüübi loomist saavutatavana pigem ilma märkimisväärset ajalist ressurssi panustamata, siis lõppkokkuvõttes, määratud skoobi saavutamiseks, oli see üsna väljakutseid pakkuv ülesanne, mille tõttu peab autor tulemusi väga heaks.

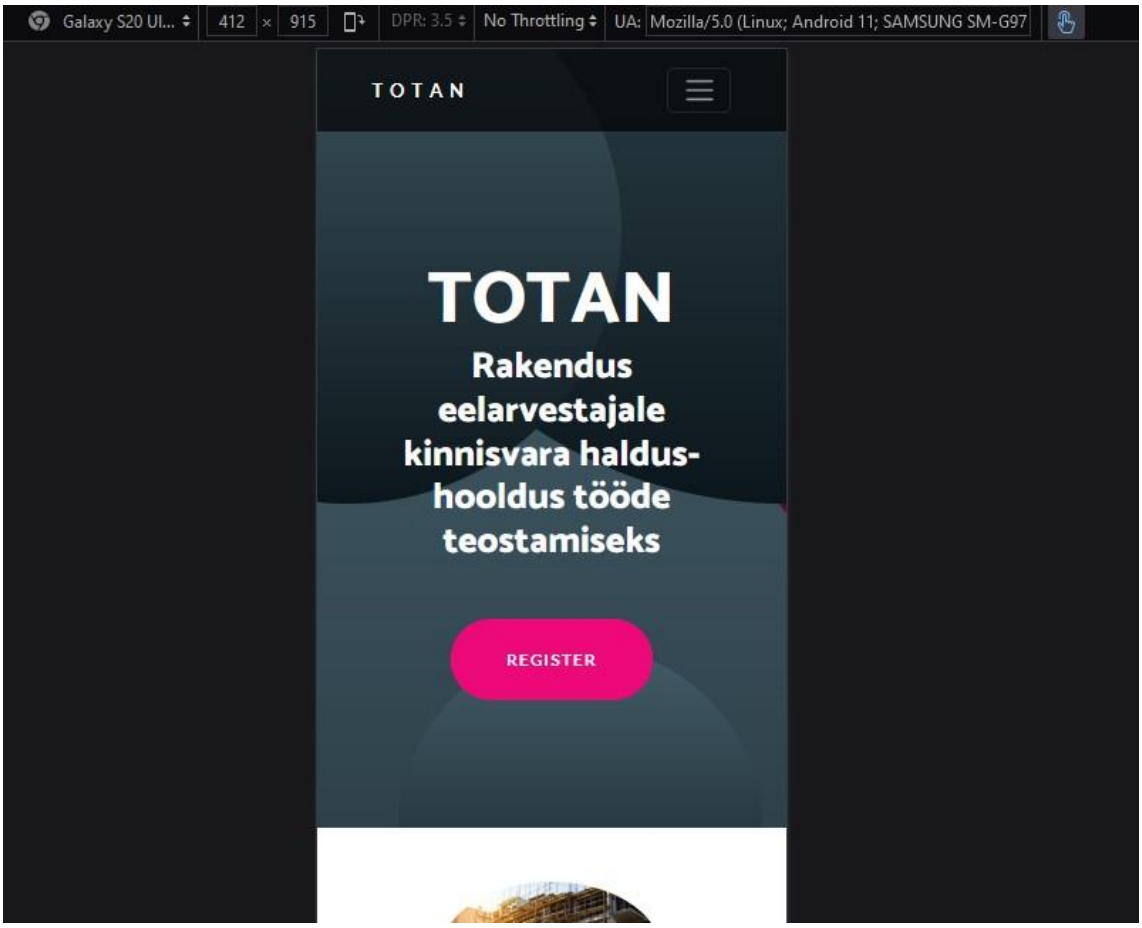
Kõik funktsionaalsused, mis prototüübile määrati said realiseeritud ning selle tulemusel õnnestus teostada rakenduse kasutustestimine. Kuigi rakenduse visuaalid ei olnud põhjalikult planeeritud ja kirja pandud ning olid autori nägemus võimalikust rakenduse disainist, siis andsid need juba hea pildi loodava rakenduse võimalikust ülevaatlikust loomust (Lisa 4).

Rakenduse abil on võimalik luua kasutajakonto ja logida selle abil veebirakendusse. Veebirakenduses on võimalik navigeerida päises asuva menüü abil kuhu on loodud lingid ettevõtetele, kontaktidele, kinnisvarale ja kalkulatsioonidele. Igal mainitud lehel on võimalik näha andmebaasi salvestatud infot ja teostada uue lisamist. Kontaktisikule kontaktide, kinnisvarale jaotuste ja kalkulatsioonidele ridade lisamiseks tuleb eelnevalt need sisestada baasinfoga, peale seda tekib konkreetse kontaktisiku, kinnisvara või kalkulatsiooni vaatesse võimalus kontaktide, jaotuste ja ridade lisamiseks. Kõiki lisatud andmeid on hilisemalt võimalik vajadusel muuta või kustutada.

Autori poolt läbiviidud kasutustestimise raames rakenduse tööd peatavaid veateateid ei esinenud, küll aga leidus mõttekohtasid võimalike täienduste tegemiseks, näiteks teavituste kuvamine salvestamisel või kustutamisel. Testimine teostati kahes erinevas seadmes, nii sülearvuti veebilehitseja (Mozilla Firefox) kui mobiiliseadme veebilehitseja

(Google Chrome) käitumine oli samaväärne ja võis veenduda, et veebirakendusele rakendatud responsiivne disain oli saavutatud.

Lisaks mobiiliseadme kasutamisele emuleeris autor erinevate seadmete ekraanisuurusi kasutades selleks veebilehitseja responsiivse disaini režiimi tööriista.



Joonis 6. Veebirakenduse avalehe emuleeritud mobiilne vaade.

Ettevõtte esindaja poolt läbiviidud kasutustestimise raames veenduti rakenduse võimalikus otstarbekuses ja soovis rakendada seda äriprotsessi teostamisel, eeldusel, et kõik kirjeldatud funktsionaalsed ja mittefunktsionaalsed nõuded on kasutusvalmis rakenduse esimeses versioonis täidetud ja saadaval.

Prototüübi põhiselt on täpsustavalt välja toodud, et hinnakalkulatsioonide koostamisel võiks rakendus hinnakalkulatsiooni numbri määrata automaatselt vastavalt mõnele esitatavale loogikale, kuid see lahter võiks jääda muudetavaks, et vajadusel saaks seda korrigeerida.

Uue hinnakalkulatsiooni, või ka edaspidiselt muude dokumentide, loomisel, võiks olla vaikimisi kuupäev määratud päevale, millal dokumenti luuakse. Sarnaselt hinnakalkulatsiooni numbri määramisele, võiks see jääda muudetavaks vastavalt vajadusele.

Tulenevalt aktuaalsest uuest maksumäärast on välja toodud ka variant kaaluda maksumäära määramise ja muutmise võimaluse lisamist rakenduse seadete kaudu.

Lisaks on jäänud prototüübis realiseerimata reapõhine summa kalkuleerimine, mis aga peaks olema kalkulatsioonide ja aktide ridadel kajastatud.

5.2 Edasiarendused

Kuna lahendus loodi prototüübina valideerimaks selle edasise arendusprotsessi otstarbekust, siis esmase ülesandena tuleks luua täieliku, kirjeldatud nõuetest lähtuva, funktsionaalsusega rakendus. Nõuded rakenduse funktsionaalsuse tagamiseks on kirjeldatud käesoleva lõputöö raames.

Kuna ettevõtte eesmärgiks on tööprotsessi raames, lisaks enda töö efektiivsemaks muutmisele, ka kliendi jaoks protsessi mugavamaks ja ülevaatlikumaks muutmine siis näeb autor ühe olulise võimalusena rakenduse jaoks kliendi vaate arendamise. Täna e-kirjade teel vahetatav kalkulatsioonide, muudatuste, teostatud tööde aktide ja muu taoline infovahetus oleks võimalik informatiivselt kokku koondada ka klientide jaoks. Selle jaoks tuleks rakendusele lisada aga näiteks kasutajagrupid, et võimaldada erinevate rollide põhiseid kasutajaid, mis võimaldaksid eeltuvastatud isikutel tutvuda endaga seotud materjalidega. Võimalusi selle loomiseks on erinevaid, üks võimalus oleks neid manuaalselt lisada ettevõtte poolt, kuid halduskoormuse vähendamiseks oleks mõistlik klientide kontaktisikud siduda isikukoodidega ning tagada ligipääs tänapäevaste identifitseerimise ja autentimise lahendustega nagu SmartID või mobiilID kasutamistega.

Lisaks informatiivsele kliendi vaatele oleks eesmärk lisada rakendusele ka dokumentide allkirjastamise funktsionaalsus kalkulatsioonide või teostatud tööde aktide mugavamaks allkirjastamiseks.

Kuigi rakenduse eesmärgiks on tekstitöötlus ja tabelarvutus rakenduste kasutamise eemaldamine tööprotsessist, võiks rakendusel olla ka lisafunktsionaalsus, mis võimaldaks

töödokumentide importimist ja eksportimist. See annab võimaluse vajadusel neid kasutada kui selleks peaks vajadus esinema. Ühtlasi annab see võimaluse arhiivi kannete teostamiseks, näiteks täna eksisteeriva arhiivi laadimine andmebaasi.

Rakendus on küll loodud ettevõtte soovil, kuid ei ole loodud ettevõtte jaoks tellimustööna, mis tähendab, et selle õigused jäävad autorile. Rakenduse funktsionaalsused on kirjeldatud tulenevalt ettevõtte äriprotsessist, kuid läbi viidud intervjuu käigus on selgunud, et see on levinud, üldkasutatav mudel, mida kasutatakse üsna laialdaselt samas valdkonnas tegutsevate teiste ettevõtete poolt. Sellest tulenevalt on tulevikus seda rakendust võimalik kohandada selliselt, et seda saaks rakendada ärilistel eesmärkidel ka teiste turul tegutsevate ettevõtete jaoks.

Rakenduse edasiarendamisel ärilistel eesmärkidel oleks aga autori arvates kindlasti vajalik viia läbi juba põhjalikum analüüs, mis hõlmaks endas turul saadaval olevate, teiste samalaadsete, rakenduste analüüsi.

Oluline oleks põhjalikumalt analüüsida, miks on meetodid laialdasemalt juba pikemaajaliselt kasutusel ning läbi viia turuuuring, mille abil selgitada rakenduse võimalik potentsiaal.

6 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli analüüsida ja kavandada rakendus, mis aitaks lihtsustada ettevõtte tööprotsessi käigus teostatavaid tegevusi. Töö käigus viidi läbi analüüs, mis aitas paremini mõista ja kirjeldada probleemi ning määrati ära töö skoop. Rakenduse analüüsi käigus kirjeldati loodava rakenduse tarbeks funktsionaalsed ja mittefunktsionaalsed nõuded. Teostati võimalike tehnoloogiate omavaheline võrdlus ja rakenduse teostamiseks vajalike tehnoloogiate valik.

Analüüsi ja nõuete kirjeldamise järgselt loodi prototüüp, mille abil ettevõtte esindajaga koostöös oli võimalik läbi viia esmane kasutustestimine, et valideerida loodav rakendus ja kinnitada või tagasi lükata arendusprotsessi jätkamine. Kirjeldati arendusprotsessi ja ees- ja tagarakenduse ülesehitust.

Püstitatud eesmärgid said täidetud, kirjeldatud said vajalikud nõuded, mis katavad ettevõtte tänast äriprotsessi, loodud sai eraldiseisvate ees- ja tagarakendusega prototüüp, mis katab määratud skoobi ulatuses funktsionaalsed nõuded ning mille abil oli võimalik teostada kasutustestimine täisfunktsionaalse rakenduse arendamise otstarbekuse määramiseks.

Lisaks tänase äriprotsessi katmisele kirjeldati ka võimalikud edasiarendused, mis annaksid rakendusele lisandväärtust ning, mis aitaks rakenduse muuta ühtlasi mugavaks tööriistaks ka ettevõtte klientide jaoks.

Kasutatud kirjandus

[1] S. Virkus, „Intervjuu liigid,“ 2016. [Võrgumaterjal]. Available: https://www.tlu.ee/~sirvir/Intervjuu_vaatlus_ja_sisuanals/intervjuu_liigid.html. [Kasutatud 10 10 2023].

[2] „Making the Web Accessible,“ [Võrgumaterjal]. Available: https://www.w3.org/WAI/. [Kasutatud 07 12 2023].

[3] Stack Overflow, „2023 Developer Survey,“ 05 2023. [Võrgumaterjal]. Available: https://survey.stackoverflow.co/2023/. [Kasutatud 30 11 2023].

[4] „Node.js,“ [Võrgumaterjal]. Available: https://nodejs.org/en. [Kasutatud 21 10 2023].

[5] MDN contributors, „JavaScript,“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Kasutatud 11 10 2023].

[6] „Advantages and Disadvantages of JavaScript,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/. [Kasutatud 21 10 2023].

[7] „Python,“ [Võrgumaterjal]. Available: https://www.python.org/about/. [Kasutatud 21 10 2023].

[8] „Python Language advantages and applications,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/python-language-advantages-applications/. [Kasutatud 21 10 2023].

[9] „Java,“ [Võrgumaterjal]. Available: https://www.java.com/en/. [Kasutatud 24 10 2023].

[10] „Advantages and Disadvantages of Java,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-java/. [Kasutatud 25 10 2023].

[11] „A tour of the C# language,“ 04 05 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 26 10 2023].

[12] „Introduction to C#,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/introduction-to-c-sharp/. [Kasutatud 26 10 2023].

[13] „React,“ [Võrgumaterjal]. Available: https://react.dev/. [Kasutatud 26 10 2023].

[14] „Pros and Cons of ReactJS,“ [Võrgumaterjal]. Available: https://www.javatpoint.com/pros-and-cons-of-react. [Kasutatud 01 11 2023].

[15] C. Goralski, „Pros and Cons of React,“ 26 05 2022. [Võrgumaterjal]. Available: https://thecodest.co/blog/pros-and-cons-of-react/. [Kasutatud 23 12 2023].

[16] „Angular,“ [Võrgumaterjal]. Available: https://angular.io/. [Kasutatud 19 11 2023].

[17] B. K. Ragala, „Advantages and Disadvantages of Angular,“ 22 09 2023. [Võrgumaterjal]. Available: https://www.knowledgehut.com/blog/web-development/advantages-and-disadvantages-of-angular. [Kasutatud 03 12 2023].

[18] „Node.js,“ [Võrgumaterjal]. Available: https://nodejs.org/en. [Kasutatud 03 12 2023].

[19] V. Agarwal, „Advantages and Disadvantages of Node.js,“ 24 09 2023. [Võrgumaterjal]. Available: https://www.codingninjas.com/studio/library/advantages-and-disadvantages-of-nodejs. [Kasutatud 05 12 2023].

[20] „ASP.NET Core,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/apps/aspnet. [Kasutatud 01 12 2023].

[21] N. Khanna, „ASP.NET Core Web Application: Cost, Features, Advantages and Disadvantages,“ 01 06 2023. [Võrgumaterjal]. Available: https://www.ebizneeds.com/blog/asp-net-core-web-application/. [Kasutatud 05 12 2023].

[22] „How Difficult is it to Learn Node.js?,“ [Võrgumaterjal]. Available: https://www.nobledesktop.com/learn/node-js/how-difficult-is-it-to-learn-node-js. [Kasutatud 23 12 2023].

[23] „Build fast, responsive sites with Bootstrap,“ [Võrgumaterjal]. Available: https://getbootstrap.com/. [Kasutatud 07 12 2023].

[24] „Rider,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/rider/. [Kasutatud 04 10 2023].

[25] „Visual Studio Code,“ [Võrgumaterjal]. Available: https://code.visualstudio.com/. [Kasutatud 04 10 2023].

[26] „Database Providers,“ 22 11 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli. [Kasutatud 25 11 2023].

[27] „Common web application architectures,“ 07 03 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures. [Kasutatud 08 12 2023].

[28] „Swagger UI,“ [Võrgumaterjal]. Available: https://swagger.io/tools/swagger-ui/. [Kasutatud 26 12 2023].

[29] „Azure,“ [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/. [Kasutatud 01 01 2024].

[30] „Use containers to Build, Share and Run your applications,“ [Võrgumaterjal]. Available: https://www.docker.com/resources/what-container/. [Kasutatud 01 01 2024].

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

Mina, Toivo Sults

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kinnisvara halduse eelarvestaja veebirakendus“, 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.

02.01.2024

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

Lisa 2 – Poolstruktureeritud intervjuu

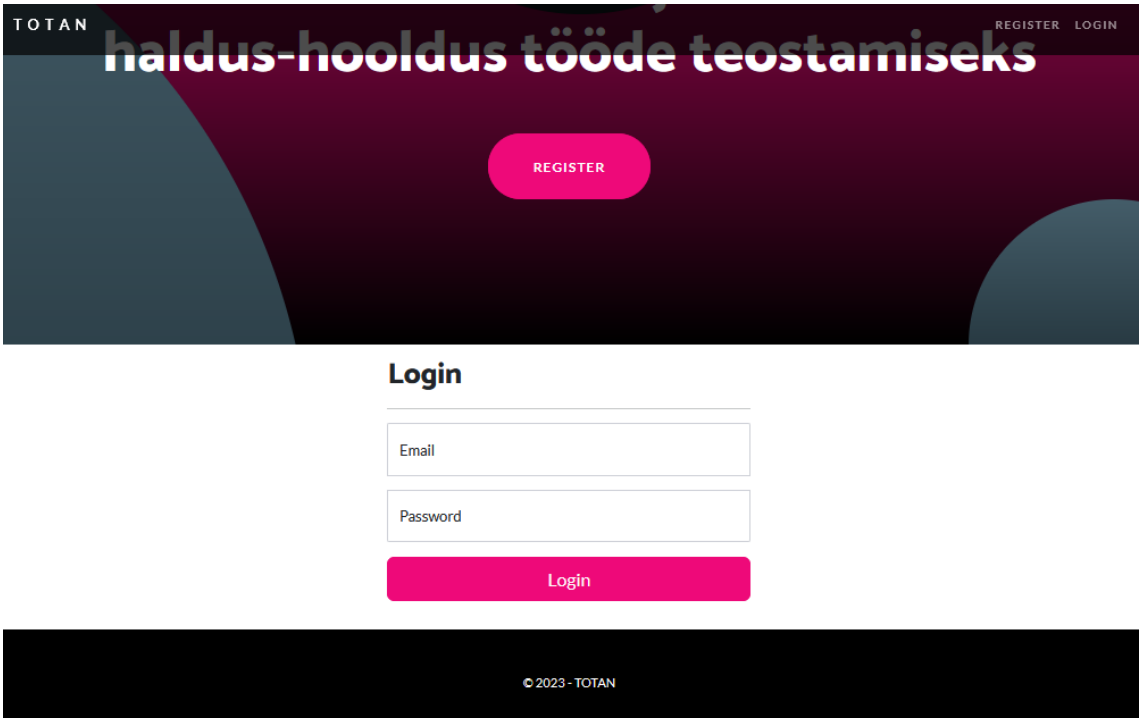
1. Palun kirjeldage üldsõnaliselt tööprotsessi ja eesmärki.
2. Kas töid teostatakse vaid enda ettevõtte poolt rendile antud ruumides?
3. Palun kirjeldage dokumentide koostamist.
   3.1. Hinnakalkulatsioonide koostamine.
   3.2. Teostatud tööde aktide koostamine.
4. Palun kirjeldage millised on nõuded dokumentidele.
   4.1. Nõuded hinnakalkulatsioonidele.
   4.2. Nõuded teostatud tööde aktidele.
5. Kas tööprotsessi mõne osaga on seotud nõudeid tulenevalt seadusandlusest ja/ või on ettevõtte enda poolt kehtestatud nõudeid sisemistele tööprotsessidele vmt.?
6. Palun kirjeldage soovitud kalkulatsioonide ja teostatud tööde aktide ridade sisu
7. Kas toote/ teenuste read on omatoode, see tähendab kas toodete üle on vaja pidada laoarvestust?
8. Kas teostatud tööde aktid on rahalise arvestuse aluseks?
9. Palun kirjeldage millised on teie peamised mured seoses dokumentide koostamisega.
10. Palun kirjeldage kuidas toimub dokumentide edastamine ja suhtlus rentnikega
11. Palun kirjeldage kuidas toimub koostatud kalkulatsioonide aksepteerimine?
12. Palun kirjeldage kuidas säilitate dokumente.
13. Palun kirjeldage millised on teie peamised mured seoses arhiiviga.
14. Kas olete kaalunud tänase protsessi kiirendamist mõne olemasoleva tarkvara abil?
15. Kas loodava rakenduse puhul on tarvis eristada rolle?
16. Kas loodavas rakenduses on vajalik pidada arvestust kinnisvara kohta?
17. Kas toote ja teenuse read peavad loodavas rakenduses olema eelsalvestatud?

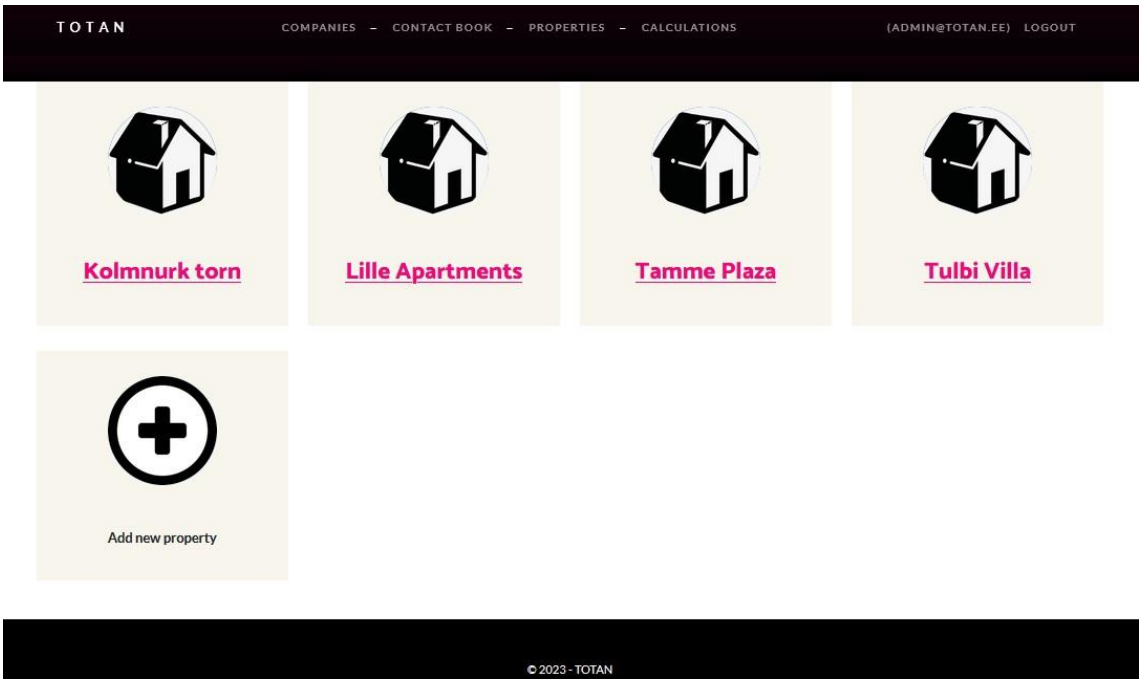
Lisa 3 – Rakenduse versioonihalduskeskkonnad

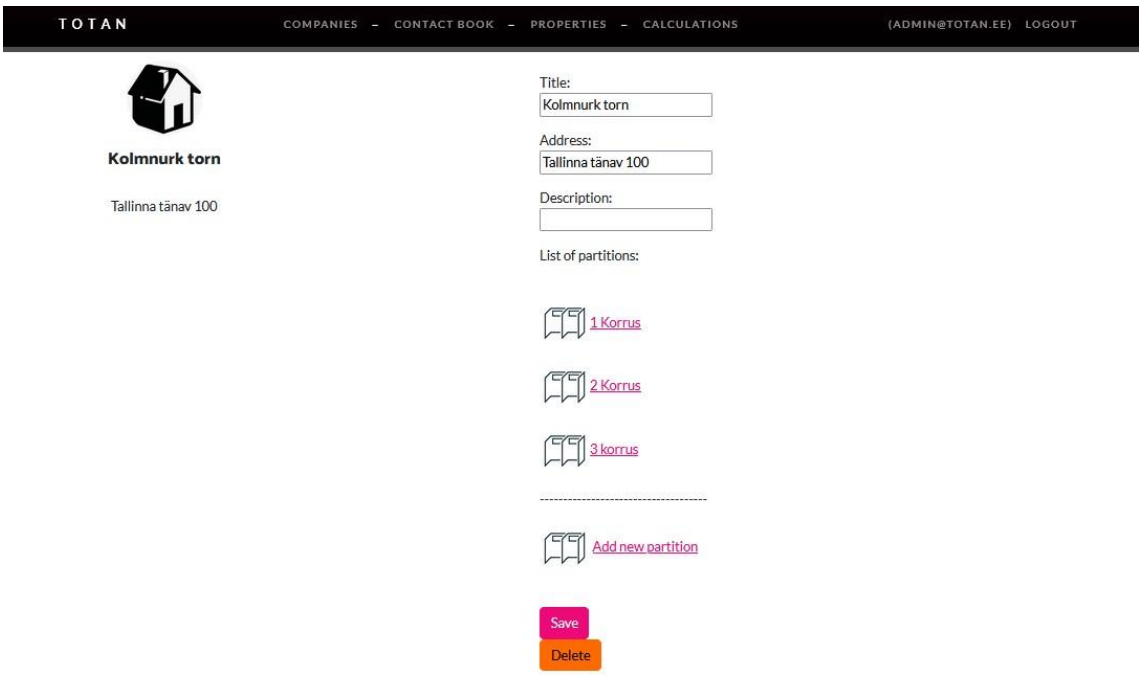
Eesrakendus: https://github.com/tosult/totanfront

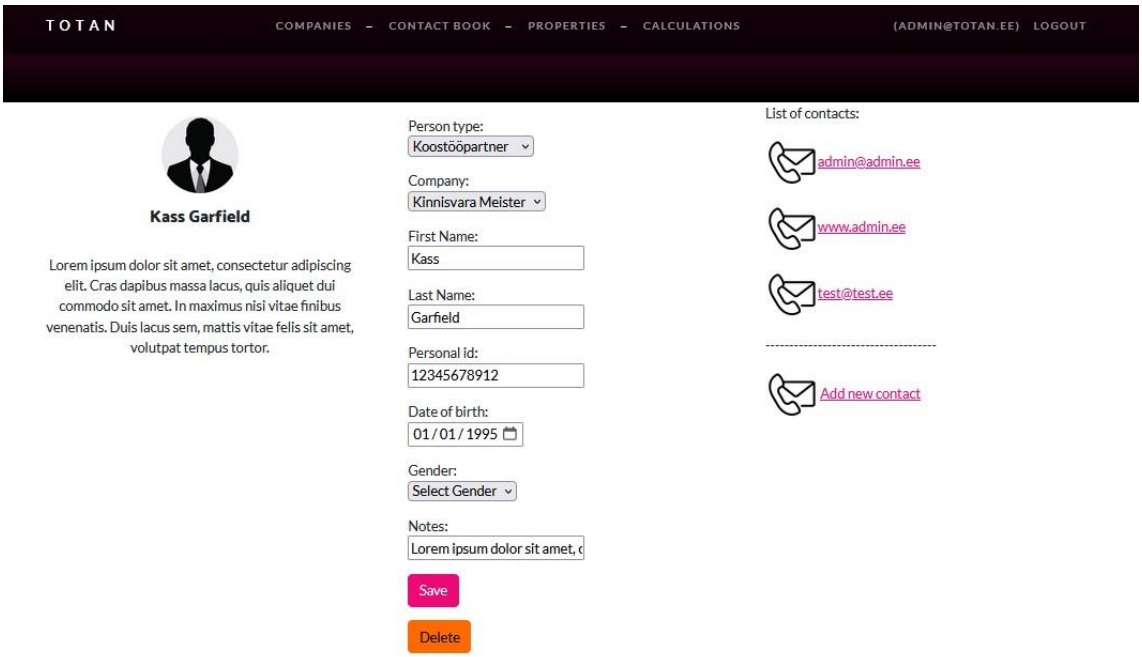
Tagarakendus: https://github.com/tosult/totanback

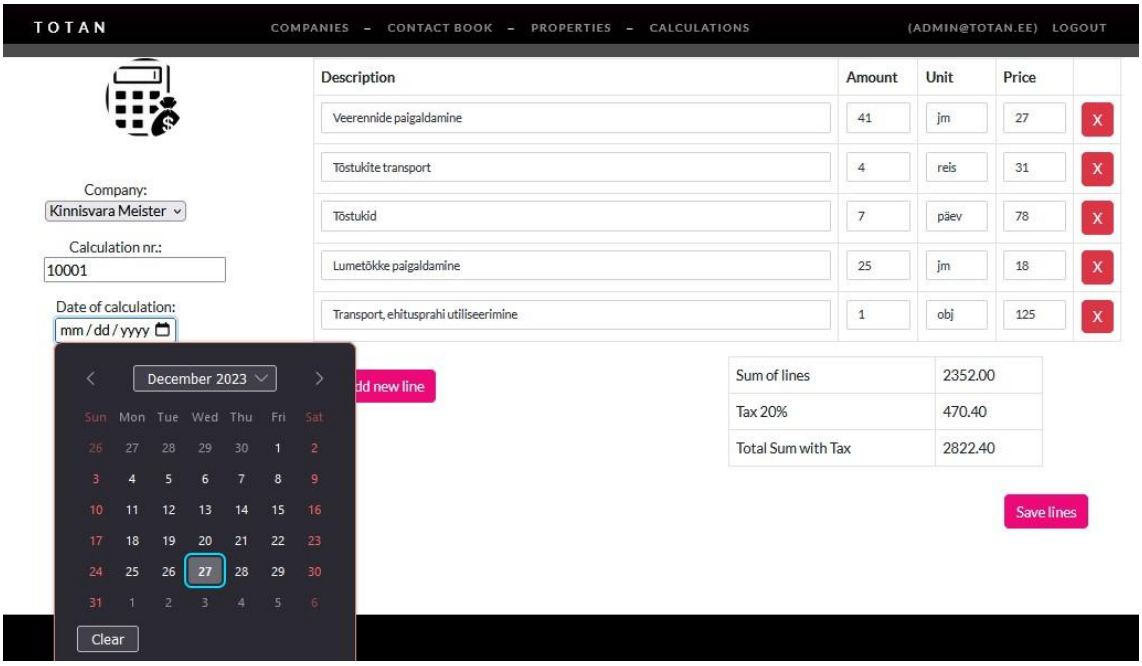
Lisa 4 – Prototüübi vaated




TOTAN
COMPANIES – CONTACT BOOK – PROPERTIES – CALCULATIONS
(ADMIN@TOTAN.EE) LOGOUT
Kolmnurk torn
Lille Apartments
Tamme Plaza
Tulbi Villa
Add new property
© 2023 - TOTAN


TOTAN
COMPANIES – CONTACT BOOK – PROPERTIES – CALCULATIONS
(ADMIN@TOTAN.EE) LOGOUT
Kolmnurk torn
Tallinna tänav 100
Title:
Kolmnurk torn
Address:
Tallinna tänav 100
Description:
List of partitions:
1 Korrus
2 Korrus
3 korrus
Add new partition
Save
Delete


TOTAN
COMPANIES – CONTACT BOOK – PROPERTIES – CALCULATIONS
(ADMIN@TOTAN.EE) LOGOUT
Kass Garfield
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras dapibus massa lacus, quis aliquet dui commodo sit amet. In maximus nisi vitae finibus venenatis. Duis lacus sem, mattis vitae felis sit amet, volutpat tempus tortor.
Person type:
Koostööpartner
Company:
Kinnisvara Meister
First Name:
Kass
Last Name:
Garfield
Personal id:
12345678912
Date of birth:
01/01/1995
Gender:
Select Gender
Notes:
Lorem ipsum dolor sit amet,
Save
Delete
List of contacts:
admin@admin.ee
www.admin.ee
test@test.ee
Add new contact


TOTAN
COMPANIES – CONTACT BOOK – PROPERTIES – CALCULATIONS
(ADMIN@TOTAN.EE) LOGOUT
Company:
Kinnisvara Meister
Calculation nr.:
10001
Date of calculation:
mm / dd / yyyy
December 2023
Sun Mon Tue Wed Thu Fri Sat
Clear
Description
Amount
Unit
Price
Veerennide paigaldamine 41 jm 27
Tõstukite transport 4 reis 31
Tõstukid 7 päev 78
Lumetõkke paigaldamine 25 jm 18
Transport, ehitusprahi utiliseerimine 1 obj 125
dd new line
Sum of lines 2352.00
Tax 20% 470.40
Total Sum with Tax 2822.40
Save lines