TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Põldroos 185688IADB

Veebirakendus korteriühistute arvete ja aastaaruannete loomiseks

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: Kristjan Põldroos

24.04.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on arendada korteriühistu Kase Tee 5 jaoks arvete ja aastaaruannete koostamiseks mõeldud veebirakendus. Loodud rakendus peaks omama sarnaseid arvete koostamise võimeid, mis on olemasoleval tabeltöötluse lahendusel, lisades samal ajal täiendavaid funktsioone ja automatiseerimist, et muuta arvete koostamise protsessi efektiivsemaks.

Töö keskendub peamiselt praeguse süsteemiga seotud ebamugavustele ning püüab neid lahendada veebipõhise rakenduse loomisega, mis viib eksisteerivad funktsionaalsused üle veebipõhisele platvormile, kus vähendatakse või kõrvaldatakse enamik probleeme, samal ajal säilitades kasutajasõbralikkuse.

Bakalaureusetöö tulemusel valmis veebirakenduse algne versioon, milles keskendutakse arvete koostamise osale. Rakendus võimaldab kasutajatel läbida kogu koostamise protsessi, alates ühistu andmete sisestamisest kuni tegelike arvete loomiseni.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 5 peatükki, 13 joonist, 3 tabelit.

Abstract

Web application for creating bills and annual reports in apartment associations

The aim of the given thesis is to develop a web application for the apartment association Kase Tee 5, which is designed for creating bills and annual reports. The application should have all the creation capabilities of the existing table-based solution, while adding additional features and automation to streamline the process of creating bills.

This thesis primarily addresses the inconveniences associated with the existing system and tries to fix them by creating an application that takes the systems features and ports them over to a web-based platform, where most of the problems are either reduced or eliminated completely, while remaining simple to use and user friendly.

As a result of this thesis, the initial version of the web application was developed, with a primary focus on the bill creator side. The application enables users to manage the entire process, from entering the association's information to generating the actual bills

The thesis is in Estonian and contains 47 pages of text, 5 chapters, 13 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	A*pplication Programming Interface,* tarkvara liides, mis võimaldab erinevatel programmidel, rakendustel või süsteemidel omavahel suhelda.
Avatud lähtekood	Kood, mis on vabalt kättesaadav ja millel puudub piirangud muutmiseks ja kasutamiseks.
Backend	Tagarakendus, serveripoolne veebirakenduse osa, kus töödeldakse ja muudetakse andmeid.
CLI	*Command line interface*, tekstipõhine kasutajaliides
CSS	*Cascading Style Sheets*, keel, mida kasutatakse veebidokumentide kujundamiseks ja vormindamiseks.
DTO	*Data Transfer Object,* andmete edastamiseks loodud objekt.
Frontend	Eesrakendus, lõppkasutaja poolne veebirakenduse osa, millega inimene otseselt kokku puutub.
HTML	*Hyper Text Markup Language,* standardne keel veebidokumentide koostamiseks, mis on mõeldud veebibrauseris kuvamiseks.
JSON	*JavaScript Object Notation,* avatud standardiga formaat, mida kasutatakse pidevalt andmete edastamiseks.
JWT	*JSON Web Token,* avatud standard, kus kasutatakse JSON-objekte andmete turvaliseks vahetuseks.
Nõrgalt tüübitud keel	Keel, kus ei kontrollita programmeerimise ajal muutujate tüüpe.
PDF	*Portable Document Format,* mitmekülgne faili formaat, mis annab inimestele lihtsa ja usaldusväärse viisi dokumentide esitamiseks ja vahetamiseks.
REST	*Representational state transfer,* tarkvara arhitektuuristiil, mis on loodud arvutisüsteemide vahelise standardi tekitamiseks
Service	Teenus, mis teostab tagarakenduses äriloogikat.
Tarkvararaamistik	Abstraktne platvorm, mis pakub standardseid lahendusi ja struktuure tarkvaraprojektide arendamiseks.
Tugevalt tüübitud keel	Keel, milles on muutujate peal tugevad kontrollid, et tekitada vähem vigu programmeerimisel.
UCD	*User Centered Design*, disainiprotsess, mille keskmeks on võetud kasutaja.

UI	*User Interface, k*asutajaliides ehk visuaalne osa millega kasutaja otseselt kokku puutub.
UX	*User Experience*, kasutajakogemus.
ZIP	Failiformaat, mis võimaldab andmete arhiveerimisist.
YAML	Inimestele loetav andmete serialiseerimiskeel, mida kasutatakse konfiguratsioonifailide kirjutamiseks.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeva kiiresti arenevas maailmas omab tehnoloogia olulist rolli meie igapäevaelus. Kommunikatsioonist arveteni on meil võimalik läbi viia erinevaid toiminguid tänu erinevatele tehnoloogilistele lahendustele. Ometi, vaatamata tehnoloogia märkimisväärsele arengule, kasutavad paljud korteriühistud endiselt traditsioonilisemaid vahendeid. Eriti levinud on see väiksemates korteriühistutes, kus suuremate süsteemide kasutuselevõtt ei pruugi olla otstarbekas.

Käesolev lõputöö keskendub probleemile, kus Korteriühistu Kase Tee 5 otsib efektiivsemat lahendust ühistu haldamiseks ning arvete ja aruannete loomiseks. Praegu kasutusel olev tabeltöötlusprogramm, kuigi toimiv, nõuab arvete ja aastaaruannete koostamiseks palju aega, mis võib ulatuda terve päevani. Lisaks levitatakse mõningat infot endiselt paberkandjal.

Lõputöö eesmärgiks on analüüsida korteriühistu jaoks sobivamaid tehnoloogilisi lahendusi ja välja töötada veebirakendus, mis oleks kergesti õpitav, visuaalselt atraktiivne ning tagaks intuitiivsema kasutajakogemuse kui tavaline tabeltöötlusprogramm.

Arendamise käigus pööratakse erilist tähelepanu sellele, et see oleks lihtsalt mõistetav ja kiiresti omandatav, isegi neile, kes pole varem sarnaste veebirakendustega kokku puutunud. Visuaalselt atraktiivne kasutajaliides aitab paremini esile tuua olulist teavet, hõlbustades andmete haldamist ja analüüsimist samal ajal vähendades tõenäosust suuremate vigade tekkimisele.

Töö on jaotatud neljaks suuremaks peatükiks. Esimese peatükis keskendutakse korteriühistu praeguste probleemide mõistmisele ja veebirakenduse nõuete ning funktsionaalsuse väljatöötamisele. Teises peatükis tehakse ülevaade veebirakenduse tehnoloogiliste valikute, arhitektuuri, kasutajaliidese disaini ja kasutatavuse kohta. Kolmandas peatükis kirjeldatakse veebirakenduse loomise protsessi, alates serveripoolses ja kasutajaliidese lahendusest kuni testimisprotsessini. Viimases peatükis vaadatakse üle lõpp-produkt, selle efektiivsus ja kliendi tagasiside.

2 Probleemi analüüs

Käesolevas peatükis uuritakse põhjalikumalt bakalaureusetöös käsitletavat probleemi ning olemasolevaid lahendusi. Lõpuks täpsustab autor oma lähenemist probleemile.

2.1 Korteriühistu probleemide ülevaade

Korteriühistus Kase Tee 5 arvete loomine jaguneb ära peamiselt kahe erineva tabeli vahel. Esimene tabel, mis sisaldab kogu informatsiooni korterite kohta ning nende kuupõhiseid makseid. Teine tabel on mõeldud veemaksete arvutamiseks. Arvete koostamise hõlbustamiseks on loodud Word'i failina arvete mall, mis täidetakse vastavalt eelmainitud tabelite andmetele. Aruannete koostamiseks kasutatakse kolmandat tabelit, mis annab ülevaate korterite maksetest kokkuvõtlikul kujul.

Teades, kuidas korteriühistu arveid ja aruandeid koostab, saame uurida ebamugavusi, mis kaasnevad praeguse lahendusega. Allpool on esitatud peamised probleemid, mis tulenevad tabeltöötlusprogrammi ja olemasolevate protsesside kasutamisest:

Tabeltöötlus – korteriühistu kasutab andmete haldamiseks ja muutmiseks tabeltöötlusprogrammi Excel. Kuigi see on võimeline teostama erinevaid arvutusi ja pakub andmete ülevaadet, võib programm olla visuaalselt ülekoormav ja kasutajatele keeruline mõista.

Korteriühistu igakuiste arvete loomine – andmete ülekandmine tabeltöötlusest Word'i vormingus arvetesse on aeglane protsess, mis võib põhjustada viivitusi arvete koostamisel ja edastamisel.

Korteriühistu aastaruannete koostamine – aruannete loomiseks peab arvete koostaja looma täpse kokkuvõtte kõikide korterite kohta. See on ajakulukas tegevus, kuna andmeid, mida tuleb analüüsida, on palju ning nõuab suurt tähelepanu detailidele.

Arvete loomise protsessi õppimine on keeruline – eksisteeriva lahenduse omandamine ei ole kiiresti õpitav, eriti arvutivõõrale inimesele, mis teeb selle süsteemi üleandmise

keerukamaks. Uuel inimesel võib tekkida tõrkeid arvete koostamise ja haldamise protsessis.

Korteriomanike puudulik ülevaade – omanikel ei ole olemas otsest juurdepääs oma andmetele, mistõttu peavad nad pöörduma arvete koostaja poole, et saada täpsustusi.

Kommunikatsioon – korteriühistul puudub ühtne ja standardne platvorm, kus jagada ja levitada teavet korterite kohta, mis võib põhjustada infovahetuse viivitusi ja segadust.

Lähtuvalt eelnevalt kirjeldatud probleemidest oli vaja välja töötada lahendus, mis aitaks neid probleeme leevendada või kõrvaldada. Algse versiooni valmimiseks keskenduti esimesele kolmele probleemile: tabeltöötlusele, korteriühistu igakuiste arvete loomisele ja korteriühistu aastaruannete koostamisele.

2.2 Veebirakenduse nõuded ja funktsionaalsus

Korteriühistute täpsemate soovide mõistmiseks toodi välja vajalikud funktsionaalsed ja mittefunktsionaalsed nõuded. Rakendati kasutajalugusid, mis on informaalsed selgitused, kirjutatud lõppkasutaja perspektiivist. Kasutajalugudel on tarkvaraarenduse protsessis oluline roll, kuna need aitavad arendajatel mõista lõppkasutaja vaatenurgast tulenevaid vajadusi ja ootusi. Erinevalt tarkvaranõuetest keskenduvad kasutajalood inimestele ja nende vajadustele, kasutades lihtsasti mõistetavat keelt, et anda arendajatele konteksti [1].

2.2.1 Funktsionaalsed nõuded

Korteriühistu veebirakenduse efektiivseks ja sujuvaks toimimiseks on oluline tagada vastavad funktsionaalsed nõuded:

- Veebirakendus on võimeline looma uusi korteriühistuid ning lisama ühistutele maja.
- Veebirakenduses on võimalik lisada majale kortereid ja muuta korterite informatsiooni
- Veebirakenduses saab lisada ja eemaldada kommunaale
- Veebirakenduses saab koostada kuuarveid korteritele

- Veebirakenduses on võimalik koostada aruandeid

2.2.2 Mittefunktsionaalsed nõuded

Veebirakenduse jaoks tähtsad mittefunktsionaalsed nõuded:

- Veebirakendus peab olema lihtne kasutada ja kergesti mõistetav
- Veebirakenduse disain ja navigatsioon peavad olema loogilised ja intuitiivsed
- Veebirakendus peab pakkuma visuaalselt atraktiivset ja professionaalset välimust
- Veebirakendus peab tagama andmete turvalisuse ja privaatsuse

2.2.3 Kasutajalood

Teades rakenduse nõudeid, oli paika pandud nende põhjal vastavad kasutajalood, mis on välja toodud tabelis 1.

Tabel 1. Nõuete põhjal loodud arvete koostaja kasutajalood.

Kellena	soovin…	selleks, et…
Arvete koostajana	luua korteriühistuid	hallata oma maju ja kortereid tõhusalt.
Arvete koostajana	lisada maja korteriühistutele	paremini korrastada ja hallata erinevaid kortereid.
Arvete koostajana	lisada kortereid majale	saada täielik ülevaade ühistutes asuvatest korteritest ning nende haldamisest.
Arvete koostajana	uuendada korterite andmeid igakuiselt	hoida informatsiooni ajakohasena ning tagada täpne arvestus.
Arvete koostajana	lisada ja eemaldada kommunaale kuu kohta	arvestada korteritele kehtivate kohustustega ning hoida arvestus korras.
Arvete koostajana	koostada arveid korteritele	tagada nende õigeaegne väljastamine ning korrektne arveldamine.
Arvete koostajana	teha aasta kohta aruandeid	saada aasta kohta täpne ülevaade.

2.2.4 Tulevikus loodavad funktsionaalsused

Arendatava lahenduse õigeaegset valmimist silmas pidades, jäid mõned funktsionaalsused edasiarenduseks. Sinna hulka kuuluvad näiteks omanikele mõeldud funktsioonid, mis annaks neile viisi ülevaadete saamiseks ja kiirendaks veelgi arvete koostamise protsessi.

Omanikupoolsed funktsionaalsed nõuded:

- Omanik saab vaadata oma korteri arveid ja makseajalugu
- Omanik saab vaadata enda kohta infot ja nõuda arvete koostajalt muudatusi
- Omanik saab sisestada enda veenäite

Omanikupoolsed mittefunktsionaalsed nõuded:

- Rakendus peab olema kiire ja reageerima kasutaja toimingutele ilma viivitusteta
- Rakendus peab pakkuma mobiiliversiooni, mis võimaldaks mugavat kasutamist nutitelefonides ja tahvelarvutites
- Omanike isikuandmeid tuleb hoida turvaliselt

2.2.5 Kasutatavus

Arvestades olemasolevat tabeltöötluspõhist lahendust, sooviti välja töötada rakendus, mis säilitaks tabeltöötluse parimad küljed ja vähendaks selle puudusi. Selle saavutamiseks tuli tagada, et informatsiooni sisestamine oleks lihtne ja mugav, samal ajal kuvades ühel lehel suurem osa informatsioonist.

Kuigi perfektse lahenduse loomine pole tabeltöötluse puhul otseselt võimalik, kuna arvete koostamine nõuab juba iseenesest palju informatsiooni, saab siiski parandusi teha, et muuta protsess lihtsamaks ja arusaadavamaks. Lähenemiseks on võetud informatsiooni nõudmiste minimaliseerimine ning andmete jagamine selgemalt struktureeritud plokkidesse, et kasutajad saaksid esitatud teavet paremini mõista.

Kasutajasõbralik disain aitab tagada, et seda oleks lihtne õppida ja kasutada, vähendades seeläbi vajadust koolituse ja juhendamise järele. Selle tulemusena peaks loodav

rakendus olema meeldivam kasutada ning aega ja ressursse säästev lahendus korteriühistu jaoks.

2.3 Olemasolevad ja alternatiivsed lahendused korteriühistule

Eestis on mitmeid lahendusi, mis on suunatud korteriühistute raamatupidamisele, kuid ükski neist ei ole spetsiifiliselt loodud ainult korteriühistute arvete loomiseks. Samuti on Anton Šiškov välja töötanud bakalaureusetöö raames korteriühistu veebirakenduse, mis aitab raamatupidamisega seotud tegevusi hallata.

2.3.1 Korto

Korto on terviklik korteriühistute haldamise ja raamatupidamise süsteem, mis hõlmab mitmeid funktsioone, sealhulgas arvete loomist, finantsarvestusi, kinnisvara haldust ja ühistu liikmete suhtlemist. Korto aitab korteriühistutel hallata nii üldisi korteriga seotud ülesandeid kui ka raamatupidamist [2]. Kuigi arvete loomine on osa Korto süsteemist, ei ole see nende peamine fookus ning süsteem ei ole optimeeritud ainult selle ülesande jaoks.

2.3.2 Digimaja

Digimaja on veel üks korteriühistutele suunatud lahendus, mis katab laia valikut haldus- ja raamatupidamisülesandeid. Digimaja pakub lisaks arvete loomisele ka eelarve koostamist, korteriühistu dokumentide haldust ning liikmete ja juhatuse suhtlemise võimalusi [3]. Kuigi Digimaja võimaldab korteriühistutel luua arveid, ei ole see lahendus spetsiifiliselt keskendunud ainult arvete loomisele.

2.3.3 Ellerex

Ellerex on mitmekülgne korteriühistute haldamise ja raamatupidamise süsteem, mis hõlmab mitmesuguseid funktsioone nagu arvete loomine, eelarve koostamine ja lepingute haldamine [4]. Ellerex kasutajaliides on visuaalselt natuke ülekoormav, andes ekraanile väga palju infot korraga. Lisaks on arvete loomine vaid üks paljudest süsteemi funktsioonidest ning ei ole nende peamine fookusala.

2.3.4 Anton Šiškov bakalaureusetöö

Anton Šiškovi bakalaureusetöö eesmärk oli arendada välja korteriühistutele mõeldud veebirakendus, mis suudab hallata kahte maja korraga. Šiškov leidis, et Eesti turul

olevad lahendused ei vasta tema nõuetele ning mõnedes aspektides pakub tema arendatud veebirakendus rohkem funktsionaalsust [5]. Kuigi ta lõi mitmekülgse veebirakenduse, ei olnud kasutajaliidese disainile suurt rõhku pandud. Sarnaselt eelnevalt mainitud lahendustele ei vasta ka Šiškovi loodud veebirakendus täielikult ettenähtud kriteeriumidele.

2.4 Arendatav lahendus

Eksisteerivate lahenduste peamine probleem seisnes funktsionaalsuse üleküllus, mis muutis need keeruliseks ja kasutajatele raskesti mõistetavaks. Arendatud lahenduse eesmärgiks oli luua arvete koostamise programm, mis oleks lihtne ja kiiresti arusaadav, kuid samal ajal piisavalt mitmekülgne, et säilitada olulisi funktsionaalsusi. Selle saavutamiseks tuli leida optimaalne tasakaal kasutajaliidese disaini ja vajalike funktsioonide vahel.

Loodud lahenduse peamine fookus oli pakkuda kasutajasõbralikku kogemust, mis võimaldaks arvete koostajale tõhusamalt ja kiiremini arveid koostada. Selle saavutamiseks peab rakendus olema intuitiivne ja visuaalselt selge, et vähendada õppimisaega ja suurendada töö tõhusust.

Bakalaureusetöö raames keskenduti arendusetapis arvete ja aruannete koostamise osale ning lahendusele loodi ainult brauseripõhine versioon. See hõlmab arvete loomise tuge, korteriühistute ja korterite andmete haldamist ning ülevaadete ja aruannete genereerimist. Edasist arendust silmas pidades jäävad mõned mugavusfunktsioonid ja omanikele mõeldud osad arendamata, mis võiksid olla paremini kättesaadavad mobiiliversioonis.

3 Loodava veebirakenduse ülevaade

Järgnevas peatükis keskendutakse loodud lahenduse tehnoloogilistele valikutele, veebirakenduse arhitektuurile ja disainile, et anda lugejale ülevaade rakenduse ülesehitusest.

3.1 Tehnoloogilised valikud

Tänapäeval on tehnoloogia kiire areng toonud kaasa arvukate tehnoloogiate, tarkvararaamistike ja tööriistade kättesaadavuse, mis pakuvad erinevaid võimalusi ja eeliseid veebirakenduste arendamisel. Selles valdkonnas on välja kujunenud mitmesugused arhitektuuristiilid ning programmeerimiskeeled, mis aitavad kaasa tarkvaraarenduse paindlikkusele ja tõhususele.

Lisaks toetavad tänapäevased tehnoloogiad mitme platvormi kasutamist, mis võimaldavad arendajatel luua rakendusi, mis töötavad erinevatel seadmetel ja operatsioonisüsteemidel. Selle tulemusena on veebirakenduste arendamine muutunud kiiremaks, võimaldades tarkvaraarendajatel keskenduda oma rakenduste põhifunktsioonidele ning kasutajakogemusele.

Enamikud veebirakendused töötavad tänapäeval REST'i peal, mis on arhitektuuristiil veebiteenuste jaoks. See on väga populaarne tänu oma lihtsusele, suurele paindlikkusele ja skaleeritavusele. RESTful API-de kasutamine võimaldab arendajatel luua lihtsaid ja kergesti mõistetavaid liideseid, mis on platvormi- ja keeleülesed. See tähendab, et RESTful API-de abil saavad erinevate tehnoloogiatega loodud rakendused omavahel suhelda ja koostööd teha [6].

Erinevalt traditsioonilistest veebirakendustest, mis vajavad iga kasutaja tegevuse jaoks eraldi lehe päringut ja laadimist, on populaarseks saanud ühe lehe arendusmudel SPA, mis pakub kasutajatele sujuvat ja interaktiivset kasutajakogemust. SPA eesmärk on vähendada serveri ja brauseri vahelist suhtlust ning optimeerida veebirakenduse jõudlust. See parandab kasutajakogemust, vähendades lehtede laadimise aega ja

võimaldades kiiremat navigeerimist rakenduse vahel. Lisaks aitab SPA kaasa rakenduse arendamisele, kuna see võimaldab arendajatel keskenduda ühele koodibaasile, mis hõlbustab hooldust ja värskendusi [7].

Kuigi veebirakenduste arendamiseks on saadaval arvukalt erinevaid tehnoloogiaid ja raamistikke, keskendus antud analüüs ainult nendele tehnoloogiatele, millega autoril oli varasem kogemus. See lähenemine tagab, et valitud tehnoloogiad on juba tuttavad, mis omakorda soosib efektiivsemat arendusprotsessi ja aitab vältida võimalikke takistusi, mis võivad tekkida uute tehnoloogiatega kohanemisel.

3.1.1 Tagarakenduse programmeerimiskeele valik

Leidub palju erinevaid programmeerimiskeeli, mida saab kasutada veebirakenduste tagarakenduse loomiseks. Siiski, käesolevas töös keskendutakse kahele peamisele võimalusele: C# ja Java. Need keeled on populaarsed, laialdaselt levinud ning nende kasutamine veebirakenduste arendamiseks pakub märkimisväärseid eeliseid.

C# - objektorienteeritud programmeerimiskeel, mille on loonud Microsoft. Seda keelt kasutatakse laialdaselt erinevate rakenduste arendamiseks, alates veebirakendustest kuni mobiilirakendusteni. C# keel põhineb C++ ja Java keelte parimatel omadustel ning sisaldab lisaks ka teisi funktsioone ja võimalusi, mis toetavad kaasaegseid programmeerimise põhimõtteid [8].

Java – samuti objektorienteeritud programmeerimiskeel, mis on universaalne ja platvormist sõltumatu. Java eesmärk on pakkuda lihtsat, turvalist ja võimsat keelt, mida saab kasutada mitmesuguste rakenduste jaoks, alates väikestest mobiilirakendustest kuni suurte ettevõtte rakendusteni. Java filosoofia on "Kirjuta kord, käivita igal pool“, mis tähendab, et Java koodi saab kirjutada ühele platvormile ja seejärel käivitada mis tahes teisel platvormil, kus on olemas Java virtuaalmasin [9].

Keele valik mõlemal juhul ei mõjutanud autori otsust oluliselt, kuna oskused nii C# kui ka Java keeles on suhteliselt samaväärsed. Autori eelistus kaldub C# poole, sest C# keeles on tehtud mitmeid mugavusi Java keelega võrreldes. Siiski, enne lõpliku valiku tegemist oli tähtis uurida ka tagarakenduse raamistikke, mis märkimisväärselt aitavad kaasa arendusprotsessile ja pakuvad täiendavaid eeliseid mõlema keele puhul.

3.1.2 Tagarakenduse tarkvararaamistiku valik

Tagarakenduse raamistiku valik on oluline osa veebirakenduse arendamisel, kuna see määrab paljuski arendusprotsessi kiiruse, paindlikkuse ja stabiilsuse. Raamistikud pakuvad eelnevalt loodud komponente ja koodilahendusi, mis võimaldavad arendajatel kiiremini ja tõhusamalt rakendusi luua. C# ja Java programmeerimiskeeltel on mõlemal olemas head raamistikud veebirakenduse arendamiseks. Nendest lähemalt oli uurimiseks võetud .NET ja Spring Boot.

.NET – Microsofti loodud avatud lähtekoodiga raamistik, mis toetab erinevaid programmeerimiskeeli, sealhulgas C#. See annab arendajatele palju tööriistu, et luua rakendusi, mis nõuavad väiksemat arvutivõimsust, samal ajal pakkudes kiireid vastuseaegu ja tugevaid turvameetmeid [10].

Spring Boot – Java-põhine raamistik, mis võimaldab kiiret ja lihtsat veebirakenduste loomist. See on loodud Spring raamistiku põhjal, kaasates selle eeliseid, samal ajal lihtsustades konfiguratsiooni ja arendusprotsessi. Spring Boot'i eesmärk on vähendada projekti alustamise aega, pakkudes eelkonfigureeritud seadistusi ja sõltuvusi, mis aitavad arendajatel veebirakendusi kiiresti üles seada ja käivitada [11].

Kuigi .NET pakub arvukalt häid võimalusi ja eeliseid arendusprotsessis, on autoril märkimisväärselt suurem kogemus Spring Boot'iga. Sellest tulenevalt eelistab autor valida Spring Boot'i kui tagarakenduse raamistiku. Samuti mängib olulist rolli vastava raamistiku alustamise lihtsus, mis võimaldab arendusprotsessiga kiiremini alustada ning fookuse panna põhifunktsionaalsusele.

3.1.3 Eesrakenduse keel ja raamistikud

Eesrakenduse loomise keelevalik oli märkimisväärselt lihtsam. Stack Overflow 2022. aasta uuringus, kus küsitleti 70 000 osalejalt nende poolt kasutatavate raamistike kohta, selgus, et JavaScript on kümnendat aastat järjest kõige populaarsem programmeerimiskeel [12].

JavaScript on eesrakenduse loomiseks suurepärane valik, kuid sellel on ka mõned puudused. Näiteks on JavaScript'is nõrgalt tüübitud keel, mis võib viia vigadeni koodi jooksutamisel. Samuti võib JavaScript'i tüübitu süsteem põhjustada raskusi koodi loetavusega, eriti suuremates projektides [13].

Selleks et kõrvaldada mõningaid JavaScript'i puudusi, on loodud TypeScript. See on JavaScript'i põhjal koostatud programmeerimiskeel, millel on tugev tüübikontroll, mis omakorda aitab vähendada vigade esinemist koodi jooksutamisel, samal ajal parandades koodi loetavust. Võrreldes JavaScriptiga pakub TypeScript paremat toetust suuremate projektide jaoks ning võimaldab arendajatel kirjutada korralikumat ja turvalisemat koodi [13].

Seega on bakalaureusetöö raames eesrakenduse keeleks valitud TypeScript. Kuid üksnes TypeScript'iga on keeruline luua eesrakendust. Selleks uuriti raamistikke ja teeke, mis kasutavad TypeScript'i ja millega on autoril endal kogemusi.

Vue – raamistik JavaScript'i ja TypeScript'i kasutajaliideste ehitamiseks. Selle eesmärk on abistada arendajaid kiire ja paindliku eesrakenduse loomisel, võimaldades neil hõlpsasti komponente luua ja nendega töötada [14].

React – avatud lähtekoodiga JavaScript'i teek, mis keskendub kasutajaliideste loomisele. Selle eesmärk on pakkuda optimeeritud lahendust komponentide abil, mis aitavad arendajatel ehitada keerukaid kasutajaliideseid, säilitades samas koodi loetavuse ja struktuuri [15].

Angular – platvorm ja raamistik, mis võimaldab luua tõhusaid ja skaleeritavaid veebirakendusi. Angular on loodud kasutajaliideste ehitamiseks komponentide abil, pakkudes paindlikku ja modulaarset arhitektuuri [16].

Bakalaureusetöö raames ei olnud eesmärk arendada väga suurt rakendust, seega polnud vajadust keskenduda kõige kiirema raamistiku või teegi valimisele. Seetõttu langetati otsus suuresti struktuuri põhjal. Angular'i struktuur eristus natuke Vue ja React'i struktuurist, kuna see jagab ühe komponendi kolmeks erinevaks failiks. Üheks faktoriks oli ka TypeScript. Kuigi Vue ja React võimaldavad TypeScript'i kasutamist, on Angular selle kasutamiseks spetsiaalselt üles ehitatud. Lisaks oli autoril kõige suurem kogemus Angular'iga, mis muutis selle valiku eelistatuks.

3.1.4 Andmebaasi valik

Korteriühistu andmeid on vaja hoida pikemas perspektiivis, seega oli vaja leida lahendus nende hoidmiseks. Probleemi lahendamiseks võeti kasutusele andmebaasid. Andmebaasid on organiseeritud kogumid struktureeritud informatsioonist ehk

andmetest, mis on salvestatud tavaliselt elektrooniliselt arvutisüsteemi. Tänapäeva kõige levinumates andmebaasides on andmed siestatud ridade ja veergudena tabelites, et muuta andmete töötlemine ja päringud efektiivseks. Seejärel saab andmeid kergesti hallata, muuta, uuendada ja kontrollida. Enamik andmebaase kasutab päringukeelt nimega SQL andmete kirjutamiseks ja pärimiseks [17].

Arvestades autori teadmisi, langes andmebaasi valik PostgreSQL'i kasuks. PostgreSQL on võimas, avatud lähtekoodiga andmebaasisüsteem, mis kasutab ja laiendab SQL keelt, ühendades palju funktsionaalsusi, mis võimaldavad andmeid turvaliselt salvestada ja vähendada töökoormust. PostgreSQL on teeninud tugeva maine tänu oma arhitektuurile, usaldusväärsusele, andmete terviklikkusele, laialdasele funktsioonide kogumile ning avatud lähtekoodiga kogukonna pühendumusele, kes tagab järjepidevalt toimivaid ja uuenduslikke lahendusi [18].

Andmebaasi tabelite loomiseks ja haldamiseks otsustati kasutada Liquibase'i. Liquibase on andmebaasi skeemi muudatuste haldamise tööriist. See võimaldab arendajatel teha andmebaasis kiiremini muudatusi, seda turvalisemalt välja töötada, alates arendusest kuni tootmiseni [19].

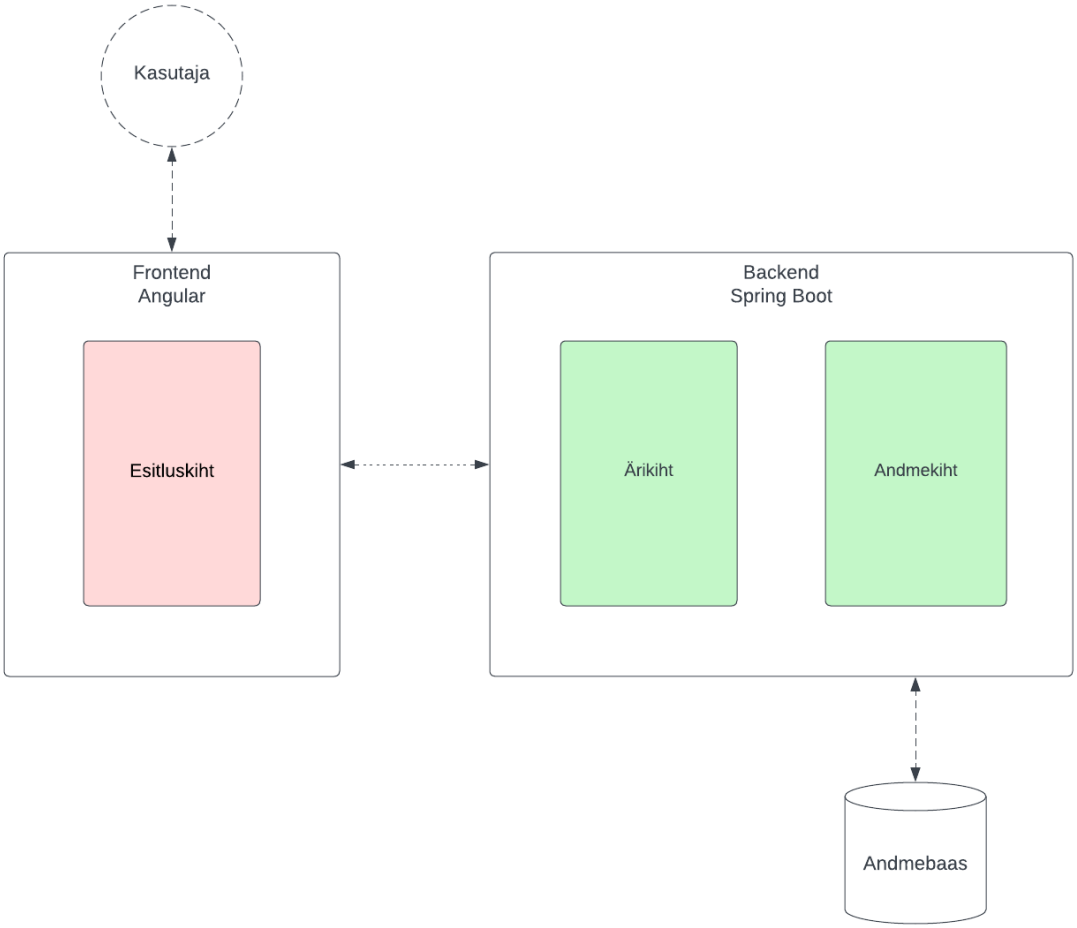
3.1.1 Docker

Selleks et lihtsustada arendust ning tulevikus ka veebirakenduse väljastamist, on kasutusele võetud Docker. Docker on avatud lähtekoodiga platvorm, mis võimaldab arendajatel rakendusi isoleeritud konteinerites arendada, käivitada ja hallata. Konteinerid on kergesti kohandatavad ning platvormist sõltumatud, tagades suurema tarkvara ühilduvuse ja paindlikkuse erinevate keskkondade vahel. Dockeri peamiseks eesmärgiks on standardiseerida rakendus konteinerite abil, et rakendus töötaks igal pool mitte ainult lokaalselt [20].

Töö raames kasutati Docker'it andmebaasi eraldiseisva komponendina hoidmiseks, kuid tulevikus on võimalik seda platvormi kasutada ka veebirakenduse väljastamiseks. Sel juhul pakitakse kogu rakendus konteineritesse, mis omakorda hõlbustab nende käivitamist teistes serverites.

3.2 Veebirakenduse arhitektuur

Veebirakenduse arhitektuur põhineb suhteliselt lihtsal struktuuril, mis koosneb kolmest peamisest kihist. Esimeseks kihiks on esitluskiht, mis kuulub *frontend'*i alla ning vastutab kasutajaliidese eest. Teiseks ja kolmandaks kihiks on äriloogika kiht ning andmetele juurdepääsu kiht, mis mõlemad asuvad *backend'*i all. Joonis 1 visualiseerib nende kihtide vahelist suhet.



Joonis 1. Veebirakenduse arhitektuur.

Esimene kiht ehk esitluskiht, on mõeldud suhtlemiseks kasutaja ja äriloogika kihi vahel. See kiht sisaldab kogu visuaalset sisu, mida kasutaja näeb ning võimaldab kasutajal REST API kaudu suhelda tagarakenduses oleva äriloogika kihiga. Teine kiht ehk ärikiht, vastutab suhtluse eest esitluskihi ja andmebaasi juurdepääsu kihi vahel. Selles kihis toimub kogu andmete haldamine ja töötlemine. Kolmas kiht ehk andmebaasi juurdepääsu kiht, vahendab suhtlust äriloogika kihi ja andmebaasi vahel. Selle kihi

eesmärk on saata andmeid andmebaasi ning hankida andmeid sealt vastavalt äriloogika kihi vajadustele ja soovidele.

3.2.1 Andmebaasi struktuur

Arvestades eelnevalt käsitletud korteriühistute probleeme ja kasutajalugusid, oli võimalik luua andmebaasi struktuur, mis toetab nende lahendamist ja vajadusi. Andmebaasis oli kavandatud tabelid nii bakalaureusetöö funktsionaalsuste jaoks kui ka võimalike edasiarenduste jaoks, et tulevaste arendustööde teostamine oleks sujuvam ja lihtsam. Andmebaasi diagramm, mis koostati programmiga Vertabelo, on nähtaval peatükis Lisa 2.

3.3 Kasutajaliidese disain ja kasutatavus

Veebirakenduse arendamisel oli oluline tagada kasutajasõbralik ja intuitiivne kasutajakogemus, selleks oli võetud kasutusele UCD. UCD ehk *User Centered Design* on disainiprotsess, mille käigus keskendutakse kasutajate vajadustele, eelistustele ja ootustele, et luua tõhusaid ja kasutajasõbralikke tooteid või süsteeme. UCD protsess hõlmab mitut etappi, sealhulgas kasutajate uurimist, disaini ideede loomist, prototüüpide arendamist ja testimist ning iteratsioonide kaudu pidevat täiustamist. UCD põhineb ideel, et parem kasutajakogemus saavutatakse, kui toodete disainimisel ja arendamisel pööratakse erilist tähelepanu kasutajate vajadustele ja ootustele [21].

Kuigi tegemist on arvete koostamise programmiga, kus vajalikku informatsiooni on suhteliselt palju kuvada, on arendusprotsessis pööratud maksimaalselt tähelepanu UI (*User Interface*) ja UX (*User experience*) disaini printsiipidele. Teades, et tegemist on programmiga, mis on alles algversioonis, püütakse siiski luua võimalikult sarnaseid elemente ja struktuure lõplikule tootele.

Tulevikus plaanitakse programmi disaini veelgi arendada, võttes arvesse tagasisidet ja kasutajate vajadusi, et muuta korteriühistu veebirakendus võimalikult meeldivaks.

4 Veebirakenduse arendus

Selles peatükis käsitletakse veebirakenduse arendusprotsessi, mis põhines analüüsile, probleemide ülevaatamisele ja kasutajavajaduste lahendamisele.

4.1 Serveripoolne lahendus

Serveripoolne lahendus ehk tagarakendus hõlmab ärikihti ja andmekihti. See vastutab suurema osa andmete töötlemise eest. Selles osas teostatakse andmete haldamine, muutmine ja edastamine veebirakenduse eri komponentide vahel. Lisaks hõlmab tagarakendus andmebaasi, kus hoitakse rakendusega seotud andmeid.

4.1.1 Tagarakenduse ülesehitus

Veebirakenduse tagarakenduse arendamisel on kasutusele võetud Gradle. Gradle on avatud lähtekoodiga ehitusautomaatika tööriist, mis on piisavalt paindlik erinevat tüüpi tarkvara ehitamiseks [22].

Kasutatud on ka Spring Boot'i soovitatud kausta struktuuri [23], kus jagatakse Java failid seose, mitte eesmärgi järgi.

4.1.2 Andmebaasi loomine

Andmebaasi haldamiseks on esmalt loodud Docker'i konteiner, mis sisaldab PostgreSQL andmebaasi. Konteiner pakub täiendavat serverit andmebaasi majutamiseks ning seda kasutatakse kahe peamise eesmärgi saavutamiseks. Esiteks võimaldab see andmebaasi säilitamist ilma pideva kustutamisvajaduseta, mis tähendab, et arendusprotsessi käigus ei ole vaja korduvalt andmeid sisestada. Teiseks pakub see tulevikus lihtsamat võimalust veebirakenduse tootmisesse viimiseks ja selle uuendamiseks.

Selleks et Docker mõistaks, kuidas konteinerit koostada, tuleb luua docker-compose.yml fail (Joonis 3), mis kasutab YAML struktuuri. YAML on lühend sõnadest

YAML Ain't Markup Language, mis on inimesele loetav andmete serialiseerimiskeel, mida sageli kasutatakse konfiguratsioonifailide kirjutamiseks [24].

```
services:
  apartment-postgres:
    container_name: apartment-postgres
    image: postgres:latest
    restart: unless-stopped
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
  logging:
   options:
     max-size: 10m
     max-file: "3"
   ports:
     - "5432:5432"
   volumes:
     - apartment-postgres-volume:/var/lib/postgresql/data
volumes:
  apartment-postgres-volume:
```

Joonis 2. Fail docker-compose.yml.

Docker’i konteineri loomisel koos PostgreSQL andmebaasiga kasutati Liquibase'i. Liquibase'i kasutamiseks koostati andmebaasi struktuurifail, mis määrab tabelite sisu ja nende reeglid. Andmebaasi struktuurifailide loomiseks on mitmeid erinevaid võimalusi, millest arendusprotsessis valiti sarnaselt Docker’ile, YAML-failide kasutamine.

4.1.3 Veebirakenduse andmed

Tagarakenduse ja andmebaasi suhtlemiseks oli loodud vastavad olendid, mis peegeldavad andmebaasi tabelite struktuuri. See tagab andmete korrektse liikumise edasi ja tagasi. Olendite loomisel on kasutusele võetud Java teek nimega Lombok. See pakub mitmeid annotatsioone, mis automatiseerivad tavalisi Java klasside funktsioone [25]. Näide korteri olendist on joonisel 4.

```
@Builder
@Getter
@Setter
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "apartment")
@SequenceGenerator(name = "apartment_id_seq", sequenceName =
                          "apartment_id_seq", allocationSize = 1)
public class Apartment {

    @Id
    @Column(name = "id", nullable = false)
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
                               "apartment_id_seq")
    private Long id;

    @Column(nullable = false)
    private String number;

    @Column(nullable = false)
    private Long floor;

    @Column(nullable = false)
    private Double size;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "apartment")
    private Set<Bill> bills = new HashSet<>();

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "building_id", referencedColumnName = "id", nullable =
                        false)
    private Building building;

    @ManyToOne
    @JoinColumn(name = "owner_id", referencedColumnName = "id", nullable =
                        false)
    private Owner owner;
}
```

Joonis 3. Korteri olendi näide.

Sõltuvalt olukorrast võib olla vajalik saata eesrakendusele kas rohkem või vähem informatsiooni. Selleks kasutatakse andmete edastamiseks ja vastuvõtmiseks spetsiaalseid objekte, mille nimeks on DTO ehk *Data Transfer Object*. DTO-de abil on võimalik muuta andmeid sobivamaks eesrakendusele saatmiseks, ilma et see mõjutaks tagarakenduse ja andmebaasi vahelist suhtlust. Korteri DTO on välja toodud joonisel 5.

```
@Data
@Builder
public class ApartmentDto {

    private Long id;

    @Size(max = 256)
    private String number;

    @NotNull
    private Long floor;

    @NotNull
    private Double size;

    @NotNull
    private Long buildingId;

    @NotNull
    private Long ownerId;
}
```

Joonis 4. Korteri DTO näide.

Lisaks on võimalik DTO-dele rakendada valideerimist, et tagada vastuvõetud info vastavus ettemääratud piirangutele. See aitab kaasa süsteemi stabiilsusele ning kaitseb puudulike andmete eest, mis võiksid mõjutada rakenduse tööd. Seega võimaldavad DTO-d andmete sujuvama ja turvalisema liikumise taga- ja eesrakenduse vahel.

4.1.4 Andmete saatmine ja vastuvõtmine

Ees- ja tagarakenduse vahelise andmevahetuse tagamiseks kasutatakse REST API-d. Spring Boot raamistikus saavutatakse see REST kontrollerite abil, mis võtavad eesrakenduselt pärit andmed vastu ja suunavad need edasi äriloogikasse, kus neid töödeldakse vastavalt nõuetele. Spring'i kontrollerid on üldiselt lihtsad ning nende peamine eesmärk on kontrollida, kas kasutajal on vastavad õigused ja andmete järgivad DTO nõudeid. Joonis 6 näitab, milline on ühistu REST kontroller.

```
@RestController
@RequiredArgsConstructor
@RequestMapping(path = "api/")
public class AssociationController {

    private final AssociationService associationService;

    @PreAuthorize("@securityService.isAssociationAllowed(#username,
                                                         #request)")
    @PostMapping(path = "association")
    public void insertAssociation(@RequestParam String username,
                                  @RequestBody @Valid AssociationDto
                                                      associationDto,
                                  HttpServletRequest request) {

        associationService.insertAssociation(associationDto, username);
    }

    @PreAuthorize("@securityService.isAssociationAllowed(#username,
                                                         #request)")
    @GetMapping(path = "association")
    public List<AssociationDto> getAssociation(@RequestParam String username,
                                               HttpServletRequest request) {

        return associationService.getAssociation(username);
    }

    ...
}
```

Joonis 5. Lõige failist AssociationController.java.

REST API kasutab nelja erinevat tüüpi päringuid: POST, GET, PUT ja DELETE. Need on vastavalt andmete sisestamine, pärimine, uuendamine ja kustutamine. API päringute eristamiseks on iga päringu ees lisatud prefiks */api*. Kontrolleritele ligipääsuks on vaja kokku panna baasaadress, API-prefiks ja aadressi lõpp.

Veebirakenduse arendamisel oli vaja luua mitmeid erinevaid päringuid, et tagada erinevate funktsionaalsuste töötamine. Päringute nimekiri on esitatud tabelis 2.

Tabel 2. Veebirakenduse API päringud.

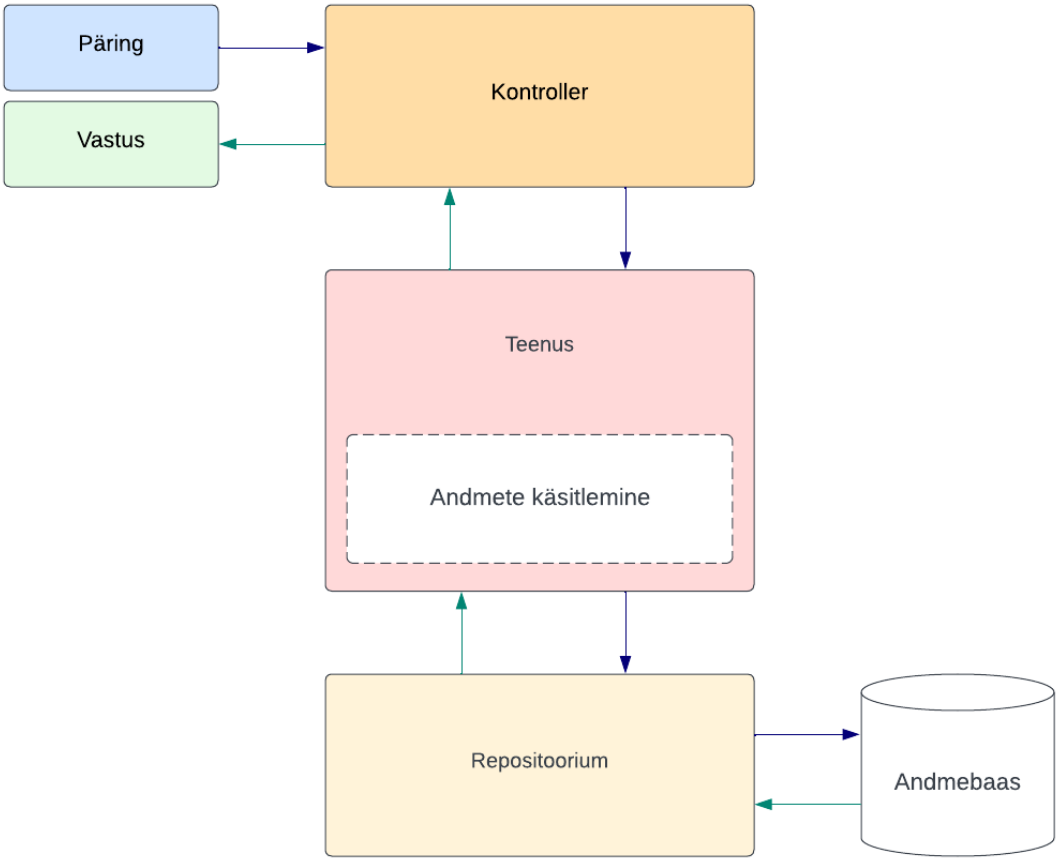
Päringu kirjeldus	Päringu tüüp	Aadressi lõpp
Kasutaja		
Sisselogimine	POST	/login
Registreerime	POST	/register
Ühistu		
Ühistu loomine	POST	/association
Ühistu pärimine	GET	/association
Maja		
Maja loomine	POST	/building
Korter		
Korterite uuendamine	PUT	/apartment
Arve Info		
Arve info pärimine	GET	/bill-info
Arve info uuendamine	PUT	/bill-info
Arve info sisestamine	POST	/bill-info
Arve info kuude pärimine	GET	/bill-info/dates
Arve infode kokkuvõte pärimine	GET	/bill-info/summary
Arved		
Arvete uuendamine	PUT	/bills
Valikud		
Valikute pärimine	GET	/selected
Valikute uuendamine	PUT	/selected
Valikute värskendamine	GET	/selected/refresh

4.1.5 Andmete käsitlemine

Andmete käsitlemiseks veebirakenduses kasutatakse *service*'id ehk teenuseid. Teenused võtavad vastu andmed või käsklused kontrolleritelt ning seejärel töötlevad, muudavad ja uuendavad andmeid vastavalt vajadusele. Teenuste abil viiakse läbi suurem osa andmete muutmise ja uuendamise protsessidest.

Kui käsklus on saadud ja andmeid on vaja edastada või hankida andmebaasist, suunab teenus selle päringu repositooriumile, mis omakorda teeb andmebaasi vastu vajalikud päringud. Repositooriumid tagavad sujuva suhtluse andmebaasiga, et hõlbustada andmete haldamist rakenduses.

Selle andmete käsitlemise protsessi toimimine on illustreeritud joonisel 7. Joonisel kujutatakse, kuidas andmed liiguvad kontrolleritest teenustesse, sealt repositooriumisse ja lõpuks andmebaasi ja tagasi.



Joonis 6. Andmete käsitlemise protsess.

4.1.6 Turvalisus

Korteriühistute andmete turvalise hoidmise tagamiseks kasutatakse Spring Security teeki. See pakub lahendusi kahele probleemile: autentimine ja autoriseerimine. Spring Security sisaldab filtrite süsteemi, mis kontrollib enne päringu töötlemist, kas kasutajal on piisavad õigused päringu teostamiseks [26].

Veebirakenduses kasutatakse vastavat tehnoloogiat kahe peamise eesmärgi saavutamiseks. Esiteks tagatakse, et tundmatud isikud ei saaks rakendust kasutada. Teiseks, et erinevad arvete koostajad ei saaks muuta üksteise andmeid. Tulevikus tuleb

lisada juurde filtreid, mis välistaks omanike võimaluse muuta arvete koostaja andmeid, kuid see põhineb samal printsiibile, mis kehtib nende vahel.

Autentimise tagamiseks kasutatakse koos Spring Security'ga JWT tehnoloogiat. JWT ehk *JSON Web Token* on avatud standard, mis on kompaktne viis turvalise teabe edastamiseks erinevate osapoolte vahel JSON-objektina. Nende turvalisuse tagab digitaalselt allkirjastamine, mis lubab neid verifitseerida [27]. Rakenduses kasutatakse kahte tüüpi JWT-sid: tavalist tokenit ja värskendus tokenit. Sisselogimisel väljastatakse mõlemad korraga, kuid neil on erinevad aegumise ajad.

Tavaline token, mis kestab 15 minutit, on mõeldud rakenduse üldiseks kasutamiseks ning selle abil saab teostada erinevaid toiminguid rakenduses. Kui keegi peaks tavalise tokeni kätte saama, on tal võimalik muuta andmeid ainult 15 minutit. Värskendustoken, mille pikkus on 30 päeva, on seevastu loodud spetsiaalselt kasutusmugavuse tagamiseks, et rakendust kasutades ei peaks kasutaja pidevalt uuesti sisse logima. Kui tavalise tokeni kehtivus lõppeb, küsitakse uut tokenit värskendustokeni abil, et tagada kasutajale katkematu ja turvaline kasutuskogemus.

4.2 Kasutajaliidese lahendus

Kasutajaliidese lahendus ehk eesrakendus, hõlmab esitluskihti ja vastutab lõppkasutaja poolt nähtava ja kasutatava sisu eest. Selles kihis kujundatakse visuaalsed elemendid ja interaktsioonid, mis võimaldavad kasutajatel rakendusega suhelda. Lisaks on esitluskihi ülesanne võtta kasutaja poolt sisestatud andmed ning edastada need ärikihile edasiseks töötlemiseks.

4.2.1 Eesrakenduse ülesehitus ja raamistikud

Eesrakenduse põhiraamistiku valikul lähtuti analüüsist ja otsustati kasutada Angular'i kõige uuemat versiooni, mis bakalaureusetöö koostamise ajal oli Angular 15. Angular'i projekti loomiseks ja haldamiseks kasutati Angular CLI-d. See on käsurea tööriist, mis võimaldab luua kiiresti projekti baaskonfiguratsiooni [28].

Rakenduse kujundamisel kasutati Bootstrap'i, mis on võimas *frontend* raamistik veebirakenduste loomiseks [29]. Täpsemalt *ngx-bootstrap'*i varianti, mis on

originaalsest Bootstrap'ist kohandatud versioon, hõlbustades selle kasutamist Angular'i projektides [30].

4.2.2 Komponendid

Komponendid on Angular'i rakenduse kõige põhilisemad kasutajaliidese ehitusplokid. Nimelt terve Angular'i rakendus koosneb komponentide hierarhiast, kus komponendid on omavahel seotud ja moodustavad rakenduse struktuuri [30]. Komponendid võivad hõlmata terveid lehekülgi või ainult üksikuid lehekülje osi.

Iga Angular'i komponent koosneb kolmest põhiosast: peafailist, mis on kirjutatud TypeScript'is ja sisaldab komponendi loogikat, HTML failist, mis kirjeldab komponendi visuaalset struktuuri ning CSS failist, mis määrab HTML elementide visuaalse stiili. Komponendid on paindlikud ja võimaldavad luua keerukaid lahendusi, säilitades samal ajal ühtset ja selget struktuuri.

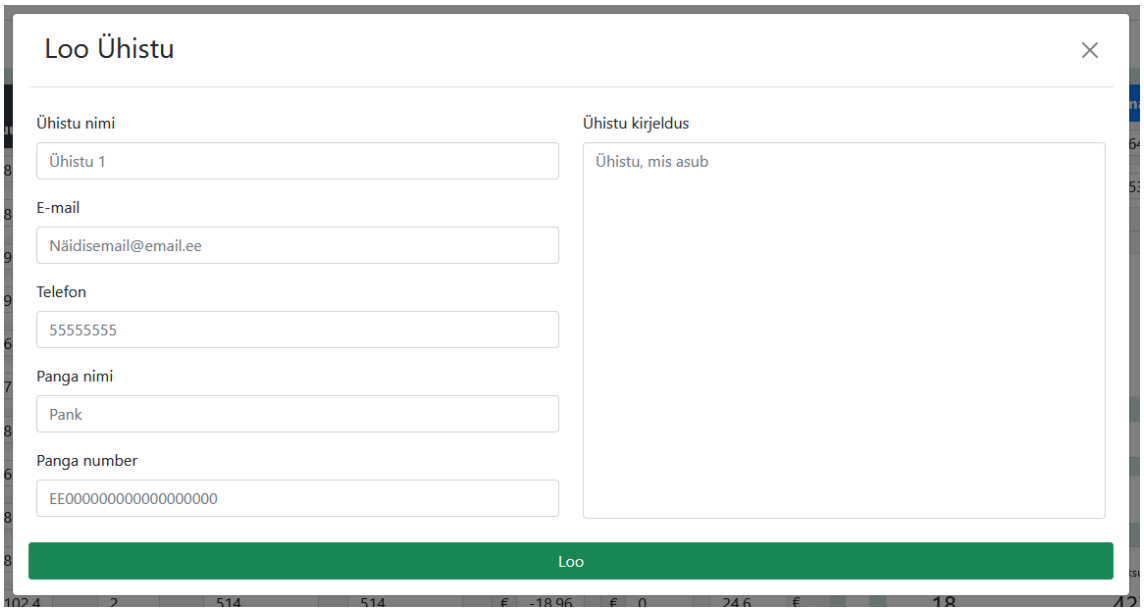
4.2.3 Teenused

Angular'i teenused on kitsa eesmärgiga klassid, mis on loodud konkreetsete ülesannete täitmiseks. Erinevalt komponentidest, mille peamine eesmärk on pakkuda kasutajakogemust, kasutatakse teenuseid ülesannete jaoks, mis ei hõlma vaate ega rakenduse loogika elemente. Sellisteks ülesanneteks võivad olla andmete hankimine serverist, kasutaja sisendi valideerimine või otse konsooli logimine [31].

Korteriühistu rakenduses kasutatakse teenuseid peamiselt tagarakendusega suhtlemiseks, kuid rakenduses on ka mitmeid teisi teenuseid, mis hõlmavad erinevaid funktsioone ja ülesandeid.

4.2.4 Korteriühistu vaade

Korteriühistu vaade on koht, kus toimub kogu korteriühistu protsess alates korteriühistu loomisest kuni arvete koostamiseni. Selle võimaldamiseks kasutatakse eesrakenduses mitut kontseptsiooni, sealhulgas hüpikaknaid. Hüpikaken on veebilehe elemendid, mis ilmuvad lehekülje peale ja inaktiveerivad kõik muud veebilehe komponendid. Neid kasutatakse selleks, et juhtida kasutaja tähelepanu olulistele tegevustele [31]. Korteriühistu veebirakenduses kasutatakse hüpikaknaid (Joonis 8) ühistu, maja ja arvete loomiseks, tagades sellega sujuvama kasutajakogemuse ilma lehekülje vahetamiseta iga loomise etapi juures.



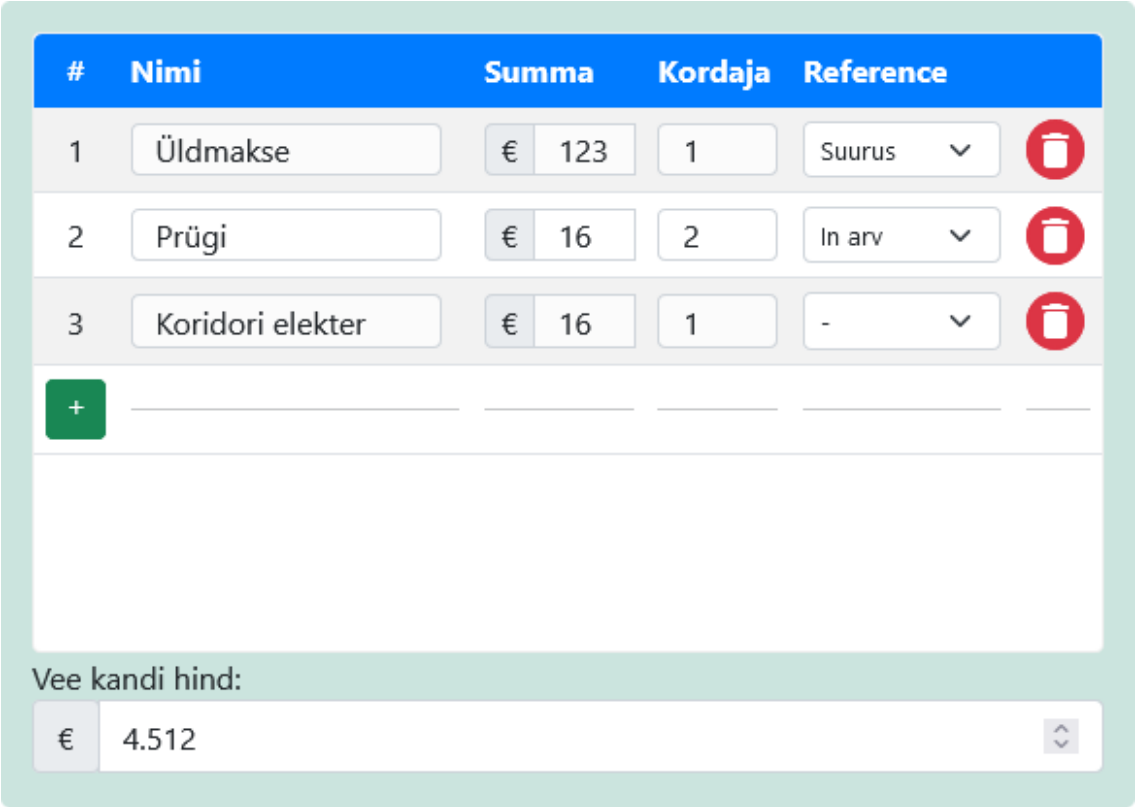
Joonis 7. Korteriühistu loomise hüpikaken.

Pärast ühistu ja maja loomist jaguneb vaade kolmeks suuremaks plokiks: kuu korterid, kommunaalid ja arveinfo. Kuu korterite plokk (Joonis 9) sisaldab kõikide korterite andmeid kuu kohta, kuhu saab vastavalt vajadusele sisestada ja muuta nende infot.

Krt nr	Omanik	Suurus	In Arv	Vee Algnäit	Vee Lõppnäit	Ettemaks	Makstud	Tasumisele kuulub
1	Test nimi 1	82.7	1	124.500	126.000	€ 0	€ 1	24.07 €
2	Test nimi 2	82.7	3	1234.500	1234.523	€ 0	€ 2.34	20.33 €
3	Test nimi 3	82.7	2	321.212	321.250	€ 0	€ 13.44	18.94 €
4	Test nimi 4	83.5	4	4321.211	4321.900	€ 0	€ 123	24.93 €
5	Test nimi 5	83.5	1	567.630	568.000	€ 0	€ 60	19.11 €
6	Test nimi 6	83.5	2	5678.920	5679.834	€ 0	€ 60	23.02 €
7	Test nimi 7	89.9	6	765.120	769.200	€ 0	€ 54	44.24 €
8	Test nimi 8	89.9	2	8765.940	8769.950	€ 0	€ 23	38.08 €
9	Test nimi 9	89.9	1	99999.000	99999.000	€ 0	€ 32	18.53 €

Joonis 8. Kuu korterite plokk.

Arvete koostamisel tekib veergu nimega "tasumisele kuulub" väärtus kommunaalide ploki kaudu (Joonis 10). Selles plokis on võimalik sisestada, muuta ja kustutada kuu kommunaale, mille järel arvutab rakendus vastava summa.



Joonis 9. Kommunaalide plokk.

Viimane plokk, kuu info (Joonis 11), on koht, kus kuvatakse kuu andmeid. Selles osas on võimalik alustada arvete koostamist, mis avab veel ühe hüpikakna, kus saab sisestada viimast lisainfot, mis on arvete loomisel vajalik.

Joonis 10. Arvete info plokk.

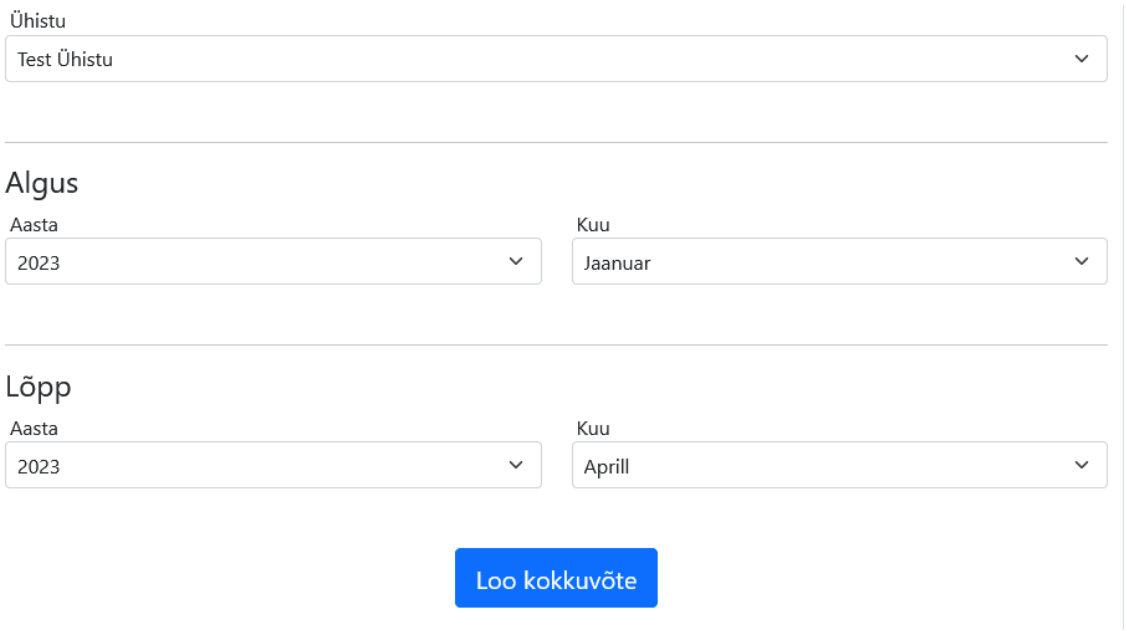
Lõpuks asub vaate ülaosas valitud kuu informatsioon, nupp järgmise kuu koostamiseks ja kuu muudatuste salvestamise võimalus (Joonis 12).

Joonis 11. Kuu lisanupud.

Järgmise kuu koostamisel luuakse uus kuu, kasutades eelneva kuu andmeid. Sellega tõstetakse üle eelmiselt kuult nii kommunaalid kui ka korterite info. Ettemaksu lahtrid täidetakse automaatselt, kasutades selleks makstud veerus olevat summat ja tasumisele kuuluvat väärtust. Lisaks sellele viiakse eelneva kuu vee lõppnäit üle uue kuu vee algnäiduks.

4.2.5 Kokkuvõtte vaade

Kokkuvõtte vaade koosneb kahest peamisest osast: kuupäevade vahemiku valimisest ning kokkuvõtte tabelist. Vaate vasakpoolses osas (Joonis 13) saab kasutaja valida ajavahemiku, mille põhjal koostatakse kokkuvõtteid. Väljad on dünaamilised, muudutes vastavalt eelnevatele lahtritele ja kuvades olemasolevaid kuid ja aastaid.



Joonis 12. Kokkuvõtte koostamise väljad.

Kui on kuupäevade vahemik valitud, siis genereerib rakendus tabeli (Joonis 14), kus näidatakse vahemiku põhjal korterite numbreid, praeguseid omanikku ja palju on nad kokku maksnud.

Kokkuvõte vahemikus: 2023 Jaanuar - 2023 Aprill

Krt nr	Omanik	Makstud kokku
1	Test nimi 1	5 €
2	Test nimi 2	2.34 €
3	Test nimi 3	13.44 €
4	Test nimi 4	128 €
5	Test nimi 5	60 €
6	Test nimi 6	60 €
7	Test nimi 7	54 €
8	Test nimi 8	23 €
9	Test nimi 9	32 €
Kokku		**377.78 €**

Joonis 13. Kokkuvõtte tabel.

4.2.6 Arvete koostamine

Arvete edastamiseks omanikele oli vajalik luua arvete koostamise teenus. Selleks kasutati avatud lähtekoodiga teeki nimega jsPDF, mis on loodud kliendipoolsete PDF-failide genereerimiseks [32]. Teegi abil on loodud peaaegu identne koopia korteriühistu varasemast arve PDF-ist. Teenus kogub arved ühte ZIP faili ning nimetab need vastava aasta ja kuu järgi, hõlbustades arvete haldamist ja saatmist. Arve näidis on oma esitatud peatükis Lisa 3.

4.3 Testimine

Veebirakenduse testimise aluseks võeti töö algfaasis määratud funktsionaalsed nõuded. Rakenduse arendamise käigus viidi pidevalt läbi teste Google Chrome'iga, mis oli selle hetke arvete koostaja enim kasutatud brauser. Viimase versiooni testi tulemused on nähtaval tabelis 3.

Tabel 3. Veebirakenduse viimase testi tulemused.

Funktsionaalne nõue	Korras
Veebirakenduses on võimalik luua uusi korteriühistuid	Jah
Veebirakenduses on võimalik lisada ühistutele maja.	Jah
Veebirakenduses on võimalik lisada majale kortereid	Jah
Veebirakenduses on võimalik muuta korterite informatsiooni	Jah
Veebirakenduses on võimalik lisada kommunaale	Jah
Veebirakenduses on võimalik muuta kommunaale	Jah
Veebirakenduses on võimalik kustutada kommunaale	Jah
Veebirakenduses on võimalik koostada kuuarveid korteritele	Jah
Veebirakenduses on võimalik koostada aruandeid	Osaliselt

Viimase testi põhjal oli näha, et veebirakendus vastab suuremale osale funktsionaalsetest nõuetest. Ainus erand on aastaaruannete koostamine, mille osaline funktsionaalsus on täpsemalt käsitletud kliendi tagasiside peatükis. Sellegipoolest näitasid testitulemused, et rakendus üldjuhul vastab määratud nõuetele.

5 Lõpp-produkti hindamine

Järgnevas peatükis analüüsitakse bakalaureusetöös arendatud veebirakenduse toimivust ning võrreldakse seda eelneva lahendusega. Käsitletakse kasutajate tagasisidet arendusprotsessi vältel kui ka selle lõpus, pakkudes ülevaadet sellest, kuidas klient rakendust vastu võttis ja milline oli tema kogemus. Lisaks arutatakse rakenduse edasiarendamise võimalusi, keskendudes võimalikele täiustustele ja uutele funktsioonidele.

5.1 Veebirakenduse efektiivsus

Uue veebirakenduse efektiivsuse hindamiseks võrreldakse seda vanema tabeltöötluse lahendusega, analüüsides erinevate toimingute sooritamist mõlemas süsteemis ja uurides visuaalset kujundust.

Uus andmete sisestamise protsess erineb natuke vanast lahendusest. Algse info saamiseks tuleb mõlemas lahenduses sisestada kõik andmed, mis võimaldavad arvutusi teostada. Uue kuu loomisel oli varasemas lahenduses vajalik arvete koostaja jaoks kopeerida eelmine kuu uude tabelisse ning seejärel teostada vajalikud arvutused. Veebirakenduses viiakse kõik eelneva kuu andmed järgmisesse kuusse automaatselt ühe nupuvajutusega ning tehakse kõik vajalikud väljavahetused, näiteks tõstetakse eelneva kuu vee lõppnäit algnäidu peale ja arvutatakse ettemaks.

Kommunaalide haldamine oli vanas lahenduses oluliselt aeglasem. Uue kommunaali lisamiseks pidi arvete koostaja muutma tabeli funktsionaalsusi, et tagada maksude korrektne arvutamine. Veebirakenduses tuleb uue kommunaali lisamiseks ainult lisada kommunaal korrektsesse plokki ning valida selle seadistused vastavalt vajadusele.

Arvete loomise protsess oli varasemas süsteemis keerulisem, kuna arvete koostaja pidi võtma andmed tabelitest ning täitma ära arvete mallid. Uues lahenduses on vaja ära täita ainult arvete info nagu tähtaeg ning seejärel loob rakendus kõigi korterite jaoks arved PDF formaadis.

Visuaalse arusaadavuse osas ei olnud vana tabeltöötlusprogramm väga hea. Kogu informatsioon oli suhteliselt sarnane, mistõttu oli kõrvaltvaatajate jaoks keeruline mõista, millist informatsiooni tabelid sisaldasid. Uuemas lahenduses on see probleem vähendatud, jagades informatsiooni erinevatesse plokkidesse ja kasutades visuaalseid elemente, mis aitavad mõista arve ülesehitust.

Võrdluse järeldusena saab öelda, et loodud lahendus on efektiivsem kui vana tabeltöötluse lahendus, pakkudes kiiremat ja lihtsamat kasutajakogemust. Täiustused võimaldavad arvete koostajatel süsteemi kasutada tõhusamalt ja saavutada soovitud eesmärke kiiremini.

5.2 Kliendi tagasiside

Veebirakenduse loomisel lähtuti kasutajakesksest disainist, mistõttu küsiti arendamise ajal kliendilt pidevalt tagasisidet. Tänu tagasisidele lahendati arendamise käigus mitmeid probleeme, et tagada parem vastavus kliendi soovidele.

Algselt oli plaanis lukustada mõned automaatselt täidetud väljad, sealhulgas vee algnäit, et vähendada inimlike vigade tekkimise võimalust. Klient tõi aga välja, et kuigi selline funktsionaalsus võib enamikul juhtudel töötada, võib see põhjustada probleeme, kui omanikud annavad valesti edasi oma vee näite või olukordades, kus toimub veearvesti vahetamine. Seega eemaldati veebirakendusest väljade lukustus täielikult.

Teine probleem, mis lahendati kliendi tagasiside põhjal, puudutas kommunaalide arvutamist. Algselt jagati kommunaalid lihtsalt korterite vahel, kuid tagasiside põhjal lisati ka spetsiifilised arvutused. Mõne kommunaali puhul oli vaja arvestada inimeste arvu korterites või korterite suurust.

Kolmas probleem, millele leiti kompromiss, oli aruannete koostamine. Aruande struktuurse keerukuse tõttu ei olnud võimalik lahendada probleemi ettenähtud ajavahemikus. Seetõttu jõuti kokkuleppele, et aruannete asemel koostatakse korterite kohta kokkuvõtted, mida arvete koostaja saab kasutada aruannete loomisel.

Lõppkokkuvõttes oli klient arendatud veebirakendusega rahul. Peamisteks eelisteks peeti automatiseerimist, mis oluliselt vähendas korteriühistu arvete koostamise aega.

Samuti hinnati uue lahenduse lihtsust, mis võimaldab tulevikus arvete loomise ülesannet sujuvamalt üle anda.

Ainus lahendamata probleem oli kommunaalide arvutamine, kus kommunaal tuleks korrutada iga korteri kohta eraldi, mitte jagada korterite vahel. Kuid hoolimata sellest probleemist oli kliendi jaoks lõpptulemus enam kui rahuldav.

5.3 Edasiarendamise võimalused

Loodud veebirakenduse peamiseks edasiarendamise prioriteediks on aruannete koostamise funktsiooni lõpuleviimine, mis aitaks ületada töö käigus tehtud kompromissi. Samuti on prioriteetne lisada kordamise baasil kommunaalide arvutus.

Arvete koostamise protsessi iseenesest saab veelgi optimeerida, lisades täiendavaid mugavusi. Loodud lahenduses puudub andmete automaatne uuendamine. Selle jaoks peab kasutaja vajutama nuppu, mis salvestab andmed andmebaasi ja teeb vajalikud arvutused. Samuti võiks arvete loomise visuaalsele protsessile lisada genereeritavate arvete eelvaate funktsiooni, mis võimaldaks kasutajatel saada parema ettekujutuse arvete lõplikust välimusest.

Veebirakendusele on veel lisamata korteriomanikele mõeldud funktsioonid. Need võimaldaksid korteriomanikel saada ülevaadet oma korteri kohta, teatada veenäitudest ja elanike muutustest. Korteriomanike olemasolu rakenduses avaks ka võimaluse automaatseks arvete väljastamiseks, kasutades selleks nende sisestatud e-posti aadresse, mis muudaks arvete saatmise ja haldamise protsessi tõhusamaks.

Pikemas perspektiivis võiks rakendusele lisada maksefunktsiooni, mis võimaldaks korteriomanikel tasuda arveid otse veebirakenduse kaudu. See lihtsustaks veelgi koostamise protsessi, kuna arvete koostajal ei oleks enam vaja manuaalselt täita omaniku poolt tehtud makseid.

Kokkuvõte

Bakalaureusetöö käigus loodi Kase Tee 5 korteriühistule arvete ja aruannete koostamise veebirakendus, mis oli kasutajasõbralik ja lihtsasti arusaadav. Veebirakendus võimaldas sisestada korterite andmeid, pakkudes seeläbi korteriühistule selget ülevaadet ning võimaldama arvete ja aruannete loomist nende põhjal.

Töö käigus uuriti korteriühistu probleemi põhjalikumalt, võrreldi olemasolevaid lahendusi ja koostati rakenduse nõuded. Analüüsiti probleemide lahendamiseks parimaid tehnoloogilisi valikuid, määrati veebirakenduse arhitektuur ning kirjeldati lähenemist disainile. Arendusfaasis selgitati tagarakenduse ja eesrakenduse ülesehitust ning kasutatud metoodikaid. Lõpuks hindas töö autor loodud lahendust ning mindi üle kliendi tagasisidest.

Arendatud rakendus oli edukas, kuna projekti lõpuks lahendati suurem osa töö alguses püstitatud probleemidest ning kliendi ootused täideti. Projekti tulemusel loodi korteriühistu veebirakenduse algversioon, mis põhineb Angular'i eesrakendusel ja Spring Boot'i tagarakendusel. Loodud lahendus pakub korteriühistule ülevaadet ning võimaldab koostada arveid ja kokkuvõtteid.

Tulevikus on kavas projekti edasi arendada, et kõrvaldada arenduse käigus tehtud kompromisse ning lisada planeeritud täiendusi.

Kasutatud kirjandus

[1] Atlassian, “User stories with examples and a template“, https://www.atlassian.com/agile/project-management/user-stories, (vaadatud 02.03.2023)

[2] Korto, Korto koduleheküg, https://korto.ee/public, (vaadatud 03.03.2023)

[3] Digimaja, Digimaja koduleheküg, https://digimaja.ee/, (vaadatud 03.03.2023)

[4] Ellerex, Korteriühistu raamatupidamine ja iseteenindus, https://ellrex.ee/korteriühistu, (vaadatud 03.03.2023)

[5] Anton Šiškov, “Korteriühistu raamatupidamis veebirakendus”, bakalaureusetöö, Tallina tehnikaülikool, 2018

[6] Lokesh Gupta, “What is REST”, 2022, https://restfulapi.net/, (vaadatud 06.03.2023)

[7] Singlepageappbook, “Single page apps in depth”, http://singlepageappbook.com/single-page.html, (vaadatud 06.03.2023)

[8] Microsoft, “A tour of the C# language”, 2023, https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/, (vaadatud 07.03.2023)

[9] IBM, “What is Java?”, https://www.ibm.com/topics/java, (vaadatud 07.03.2023)

[10] AWS, “What is .Net?”, https://aws.amazon.com/what-is/net/, (vaadatud 07.03.2023)

[11] IBM, “What is Java Spring Boot?“, https://www.ibm.com/topics/java-spring-boot, (vaadatud 07.03.2023)

[12] “2022 Developer Survey”, veebilehe StarckOverflow poolt läviidud uurimust arendajate tehnoloogiste valikut osas aastal 2022, https://survey.stackoverflow.co/2022/#most-popular-technologies-language, (vaadatud 08.03.2023)

[13] Nihar Raval, „TypeScript vs JavaScript: The Difference You Should Know”, 2023, https://survey.stackoverflow.co/2022/#most-popular-technologies-language, (vaadatud 09.03.2023)

[14] Vue ametlik dokumentatsioon, https://vuejs.org/guide/introduction.html, (vaadatud 09.03.2023)

[15] React’i ametlik dokumentatsioon, https://react.dev/learn, (vaadatud 09.03.2023)

[16] “What is Angular?“, Angulari ametlik dokumentatsioon, https://angular.io/guide/what-is-angular märts, (vaadatud 09.03.2023)

[17] Oracle, “What Is a Database?“, https://www.oracle.com/database/what-is-database/, (vaadatud 09.03.2023)

[18] PostgreSQL-i ametlik dokumentatsioon, https://www.postgresql.org/about/, (vaadatud 11.03.2023)

[19] Liquibase’i ametlik dokumentatsioon,

https://docs.liquibase.com/concepts/introduction-to-liquibase.html, (vaadatud 11.03.2023)
[20] Dockeri ametlik dokumentatsioon https://docs.docker.com/get-started/overview/, (vaadatud 15.04.2023)
[21] Jenny Preece, Yvonne Rogers, Helen Sharp, “Interaction Design: Beyond Human - Computer Interaction”, kolmas köide, Wiley, 2011
[22] Gradle ametlik dokumentatsioon, https://docs.gradle.org/current/userguide/what_is_gradle.html, (vaadatud 18.04.2023)
[23] “Structuring Your Code”, Spring Booti ametlik dokumentatsioon, https://docs.spring.io/spring-boot/docs/current/reference/html/using.html#using.structuring-your-code, (vaadatud 19.04.2023)
[24] Red Hat. “What is YAML?”, 2023, https://www.redhat.com/en/topics/automation/what-is-yaml, (vaadatud 19.04.2023)
[25] Javatpoint, “Lombok java”, https://www.javatpoint.com/lombok-java, (vaadatud 19.04.2023)
[26] Spring, “Spring Security Architecture”, https://spring.io/guides/topicals/spring-security-architecture/, (vaadatud 20.04.2023)
[27] auth0,“ Introduction to JSON Web Tokens“, https://jwt.io/introduction, (vaadatud 20.04.2023)
[28] “CLI Overview and Command Reference”, Angulari ametlik dokumentatsioon, https://angular.io/cli, (vaadatud 21.04.2023)
[29] Bootstrap, Bootstrapi koduleheküg, https://getbootstrap.com/, (vaadatud 21.04.2023)
[30] “Component“, Angulari ametlik dokumentatsioon, https://angular.io/api/core/Component, (vaadatud 21.04.2023)
[31] Jamie Juviler, “What Is a Modal and When Should I Use One?“, 2022, https://blog.hubspot.com/website/modal-web-design vaadatud (vaadatud 22.04.2023)
[32] jsPDF-i dokumentatsioon, https://www.npmjs.com/package/jspdf (vaadatud 22.04.2023)

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

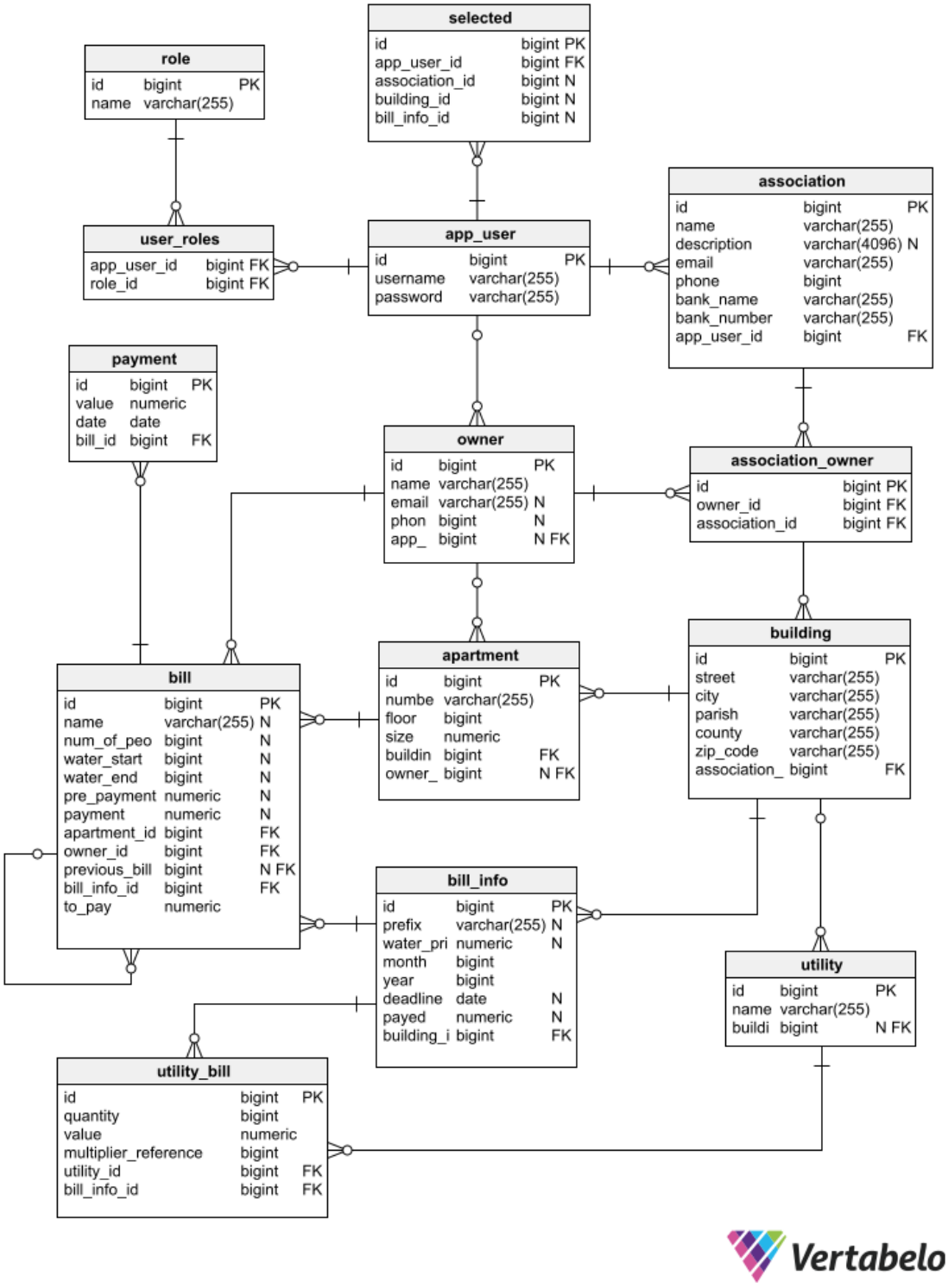
Mina, Kristjan Põldroos

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakendus korteriühistute arvete ja aastaaruannete loomiseks“, 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.

24.04.2023

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.

Lisa 2 – Andmebaasi diagramm



Lisa 3 – jsPDF-iga koostatud arve

Arve nr 1
Kuupäev: 16.01.2023

Arve saaja: **Test nimi 1**
Aadress: Test tänav 1 - 1
Test Küla
Test Vald
12345 Test Maakond

Arve esitaja: **Test Ühistu**
Aadress: Test tänav 1
Test Küla
Test Vald
12345 Test Maakond

AS Test Pank
EE00000000000000000

Kululiik	Periood	Kogus	Ühik	Ühiku hind	summa
Vesi/Kanalisatsioon	Jaanuar	1.5	m3	4.512	6.77
Üldmakse	Jaanuar	82.7	m2	0.17	14.06
Prügi	Jaanuar	1	in	1.46	1.46
Koridori elekter	Jaanuar	1	krt	1.78	1.78
		Kokku			**24.07**
		Võlg			0
		Ettemaks			0
		Tasumisele kuulub			**24.07**

Maksetähtaeg: 28.02.2023

Algnäit: 124.5
Lõppnäit: 126

Koostaja: Kristjan Põldroos
Telefon: 55555555

Siia käib lisainfo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum quis fringilla purus. Aliquam erat volutpat.
Phasellus maximus rutrum felis ut fringilla. Vivamus felis purus, porttitor non turpis in, maximus fringilla dolor. Phasellus et est ex.
Suspendisse potenti. Maecenas aliquet eros sit amet magna aliquet, eu sagittis lacus dignissim. Nulla quis volutpat justo.