TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Desiree Himuškin 193360IADB

Laevapileti hinna arvutamise veebirakendus suuremõõtmelisi vedusid korraldavale ettevõttele

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2022

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: Desiree Himuškin

15.05.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua laevapileti hinna arvutamise veebirakendus suuremõõtmelisi vedusid korraldavale ettevõttele. Probleemi olulisuse paremini mõistmiseks on sellest ettevõttest ja tema konkurentidest on tehtud ka lühike ülevaade.

Analüüsi osas on välja toodud erinevate kasutajagruppide kasutajanõuded, mida on võimalik visuaalselt näha ka plokkskeemide kujul. Analüüsitakse eksisteerivaid lahendusi ja nende puudusi, pannakse paike uue lahenduse skoop ja veebirakenduse disain ning analüüsitakse tehnoloogilisi valikuid.

Arenduse käigus luuakse veebirakendus, kus kasutaja sisestatud veose parameetrite põhjal arvutatakse ja kuvatakse kõikidele sobivatele väljumistele hinnad. Iga arvutatud hinna kohta on võimalik näha ka detailvaadet, kus on välja toodud kõik lõpphinnas sisalduvad teenusehinnad. Lisaks sellele saab kasutaja tutvuda nii väljumisgraafikute kui ka hinnakirjadega. Arvutamiseks vajalikke andmeid saab hallata nii ettevõtte administraator või administraator õigustega kasutaja. Töö rõhk on luua ammendav arvutamisloogika ja töötav veebirakendus, mis automatiseeriks tegevuse, mida on seni tehtud käsitsi koos tabeltöötlusega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 7 peatükki, 12 joonist, 2 tabelit.

Abstract

Ship Ticket Calculating Web Application for a Company Organizing Special Transportation

The aim of the current thesis is to create a web application for calculating the price of a ship ticket for a company organizing special transportation. Currently companies use their own solutions, which may not be very effective.

This thesis gives a brief overview of the company who would benefit from the web application and their competition. The analysis contains an explanation of the problem and analyses existing solutions and their disadvantages. It also includes user stories which are divided into three groups based on user groups and each user story can be visually seen on block diagrams. It also contains an analysis of technological choices for database, frontend and backend and gives and overview of the application structure and design.

During the development a web application is created, which calculates and returns transportation costs for each suitable departure based on user inserted shipment parameters. The user can see a detailed view of every calculated price. They can also see the pricelists and departure schedules. The inserted data used for computing is managed by company administrator or administrator. The emphasis is to create a functioning web application with exhaustive computational logic to fully automate calculating.

The development process is divided into four subchapters: database development, backend development, frontend development and testing. The last chapter gives an overview of possible future development ideas, for an example adding a booking functionality.

The thesis is in Estonian and contains 35 pages of text, 7 chapters, 12 figures, 2 tables.

Lühendite ja mõistete sõnastik

AJAX	*Asynchronous JavaScript and XML* – tehnoloogiate kogum, mille abil saab teha asünkroonseid päringuid
API	*Application Programming Interface* – ühendus erinevate rakenduste vahel
ASP.NET	*Active Server Pages Network Enabled Technologies* – raamistik dünaamiliste veebilehtede loomiseks
BLL	*Business Logic Layer* - äriloogikakiht
CRUD	*Create, Read, Update, Delete* – põhilised funktsioonid andmebaasi kasutamiseks
DAL	*Data Access Layer* - andmepöörduskiht
DOM	*Document Object Model* – dokumendi objektmudel
EF	*Entity Framework* – andmebaasiga suhtlust lihtsustav tarkvararaamistik
GNU	*General Public License* – vabavaralise tarkvara litsents
IMO	*International Maritime Organisation* – ohtliku veose tähistus meretranspordil
JSON	*JavaScript Object Notation* – JavaScriptil põhinev andmevahetus vorming
MP3	*MPEG Audio Layer-3* – helifaili vorming
MVC	*Model-View-Controller* – tarkvara arhitektuurimuster
.NET	Tarkvararaamistik
Object-based skriptimiskeel	Keel, kuhu on sisse ehitatud objektid
SQL	*Structured Query Language* – andmebaasipäringu keel
UI	*User Interface* – ühenduslüli kasutaja ja programmi vahel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Ülegabariidiline vedu ehk erivedu on veosega või veoseta sõiduki või sõidukite kombinatsiooni liiklemine teel, mille vähemalt üks mõõde või kaal ületab Eestis lubatud suurimad mõõtmeid, lubatud massi või teljekoormust [1].

Selliseid vedusid korraldavatele ettevõttetele on oluline leida kõige kuluefektiivseim teekond ning tihti sisaldab see teekond ka laevatransporti. Sobiva teekonna leidmiseks üle lahe, ei saa laevagraafikust lihtsalt väljasõitu valida. Iga veose transpordi planeerimisel tuleb arvesse võtta selle eripärasid, sest võib-olla ei ole seda võimalik transportida ükskõik millise laevaga.

Käesoleva lõputööga analüüsitakse eksisteerivaid lahendusi ja autori enda pakutud lahendust. Autori pakutud lahenduseks on veebirakendus, mis leiab veose andmete põhjal sobivad väljumised ja arvutab neile hinnad. Autori loodud rakendus lihtsustaks ja kiirendaks nendel ettevõtetel veose andmete põhjal sobivate väljumiste ja parimate hindade leidmist.

Probleemi olemuse selgitamiseks teostatakse intervjuu ja analüüsitakse eksisteerivate lahenduste puudusi. Selle alusel määratakse uue rakenduse funktsionaalsed nõuded.

Lahenduse analüüsi käigus tuuakse välja erinevad kasutajagrupid ja neid puudutavad nõuded, mis on jaotatud funktsionaalseteks ja mittefunktsionaalseteks nõueteks. Samuti analüüsitakse erinevaid tehnilisi lahendusi ning põhjendatakse teenusepoolsete ja kliendipoolsete tehnoloogiate valikut.

Lahenduse valmimist on kirjeldatud osadena, mis käsitlevad nii rakenduse andmebaasi ehitust kui ka serveripoolse ja kliendipoolse lahenduse valmimist. Lisaks sellele on lõpuosas kirjeldatud ka lahenduse testimist ja edasisi samme väljaspool lõputöö skoopi.

2 Ülevaade ettevõttest Kaarlaid ja nende konkurentidest

Kaarlaid OÜ on 2000. aastal asutatud ettevõtte, mis korraldab erivedusid, alates marsruudi planeerimisest ja dokumentide vormistamisest kuni veokorraldamiseni. Neile on oluline pakkuda oma klientidele kvaliteetset ja taskukohast veoteenust. Neil on selleks Eesti suurim masinapark eriveohaagiseid, umbes 60 erimõõdus haagist.

Nad on transportinud näiteks moodulmaju, suuri mahuteid, sillaosasid ja pelletite ladustamise konteinereid. Nendel kõikidel veostel on ühine see, et nad on mõõtudelt suured, mistõttu ei ole nende transportimine lihtne ning eeldab pikka ja põhjalikku planeerimistööd. [2]

Enne marsruudi paika panemist peab tegema põhjaliku marsruudiuuringu, et leida veose transpordiks sobiv teekond. Selleks on nad isegi ehitanud teid ja sildu, et veost sihtpunkti toimetada. Nende igapäevane töö hõlmab ka teekatte arvutusi, sildade ja viaduktide tugevusarvutusi. Sellises töös vigadele ruumi ei ole, sest vale marsruudi valik põhjustab tööseisakuid ja võib kaasa tuua ka trahve. [3]

Nad pakuvad transpordi teenuseid erinevates valdkondades: põllu- ja metsamajandus, keemia-, õli- ja gaasitööstus, energiatööstus, metallitööstus, ehitus ja lammutus, karjäärid ja kaevandus, laevatööstus ning jäätmekäitlus ja taaskasutus. [4]

Eestis tegutseb umbes paarkümmend ettevõtet, kes korraldavad vedusid. Mõned neist on suuremad ettevõtted, mis pakuvad ka muid veoteenuseid, peale ülegabariidiliste vedude. Kaarlaidile pakub Eestis ülegabariidilistes vedudes konkurentsi vaid paar ettevõtet.

Nende kõige suuremaks konkurendiks on rahvusvaheline logistika ettevõtte DSV, mis loodi juba aastal 1976 [5]. Nende ettevõtte pakub erinevaid transpordi võimalusi: õhu-, mere-, maantee-, raudteetransporti, kullerteenust ja projektvedusid [6]. Just projektvedude all mõeldaksegi saadetisi, mis on rasked, ülegabariidilised või muidu keerukad [7]. Nende eelis on see, et nemad pakuvad teenust ülemaailmselt, tänu millele, saavad korraldada transpordi ühest maailmanurgast teise.

Nende teine suurim konkurent on vanim Eestis tegutsev raskevedude ja erivedude ettevõte JTH Eesti OÜ, mis pakub enda teenuseid juba aastast 1992. Nad korraldavad raske- ja ülegabariidilisi vedusid Balti riikides, Lääne- ja Põhja-Euroopas ning Venemaal. Nad on muuhulgas transportinud moodulmaju, tuulegeneraatorite osi ja suuri tsisterne.[8]

3 Ülevaade probleemist

Laevatranspordi planeerimine on hetkel aeganõudev ja keeruline, sest kõige soodsamate väljumiste leidmiseks tuleb ette võtta mitmete erinevate laevafirmade väljumisgraafikud.

Kõigepealt tuleb välja filtreerida laevad, millele veos peale mahuks ja siis hakata arvutama iga sobiva laevaväljumise maksumust, millele lisanduvad muud hinnad sõltuvalt sadamatest, laevast ja koormast. Kõige täpsemini peab silmas pidama koorma mõõtmeid, sest erinevatel pikkuse ja laiuse vahemikel on erinevad meetrihinnad. Ka erinevatel sadamatel võivad laevast ja koormast sõltuvalt olla erinevad maksud ja väljumise kellajad.

Tüüpilise väljumise hind koosneb veose pikkuse maksumusest, ülelaiuse maksumusest, elektriühenduste kogusest, sadama tasust, mis sõltub laevast ja koorma kaalust, terminali tasust ja kütuse lisatasust.

Selline arvutamine tabeltöötlusega on aeganõudev ja veaohtlik.

3.1 Eksisteeriv lahendus

Eestis ühtset üle kõigi ettevõtete eksisteerivat lahendust ei leidu. Küll aga leidub palju firmasid, kes pakuvad teenusena kauba meretransporti. Sellest võib järeldada, et igal ülegabariidilisi vedusid korraldaval ettevõttel on loodud enda firmasisene lahendus laevatranspordi planeerimiseks.

Kaarlaidi ettevõttes on kasutusel tabeltöötlus. Tabeltöötluse rakendusse on loodud kahe erineva laevafirma kohta kaks erinevat arvutamisloogikat.

Kasutaja sisestab etteantud lahtritesse veose pikkuse, kaalu, juhtide arvu, lisajuhtide arvu, elektriühenduste koguse, IMO (*International Maritime Organisation*) koguse ja kirjutab number ühe lahtrisse, mille vahemikku jääb antud veose ülelaius, teistesse ülelaiuste lahtritesse tuleb kirjutada number null. Seda selleks, et hinna arvutamisel teiste ülelaiuste hinnad ei mõjutaks lõpptulemust. Iga väljumise kohta arvutatakse maksumus korrutades ja kokku liites kasutaja sisestatud muutujaid ja konstandid.

Selle lahenduse puuduseks on see, et tulemus arvutatakse kõikidele väljumistele, arvesse võtmata, kas selliste parameetritega veos ka tegelikult nendele väljumistele lubatakse. Kasutaja peab hakkama ise väljumiste graafikust leidma sobivaid väljumisi ja mis on antud laevaga lisanduvad kulud. See omakorda raskendab soodsaimate väljumiste leidmist. Lisaks sellele, on selle lahenduse puuduseks ka see, et iga laevafirma jaoks tuleb luua eraldi arvutamisloogika.

Lisaks sellele tuleb silmas pidada ka terminale, mille kaudu laevale minnakse või maha tullakse, sest kui see osutub liiga väikseks, siis peab ettevõte küsima mõne muu terminali lahti tegemist, millest veos läbi mahuks.

3.2 Uue lahenduse skoop

Eksisteerivate lahenduste puuduseks on see, et puudub ühtne lahendus, mida ettevõtted saaksid kasutada, seetõttu tulebki ettevõtetel ise luua enda firmasisesed lahendused, mis ei pruugi olla kõige efektiivsemad. Ka väljatoodud lahenduse puuduseks on see, et lõpphinna arvutus sõltub siiski suuresti kasutajast ja tema tähelepanelikkusest. Kasutaja peab ise näpuga järge ajama, mis väljumised on üldse sobivad ja milline neist tuleb soodsaim.

Selle tõttu pakub lõputöö autor välja lahenduse luua veebirakendus Eestis ülegabariidilisi vedusid planeerivatele ettevõtetele, mis arvutaks väljumiste hinnad, võttes arvesse ka laevade mõõtmeid ja arvutaks hinna ainult nendele väljumistele, millega antud veost saab transportida. Lisaks sellele on võimalik näha ka iga väljumise kohta täpsemalt, missugustest maksudest ja tasudest arvutatud lõpphind koosneb. See tuleb abiks ka hiljem, kui on vaja veenduda esitatud arve õigsuses. Kasutaja näeb veebirakenduses ka hinnakirjasid, sõidugraafikuid ja muid andmeid, mis on arvutamiseks vajalikud.

Lahendus piiritleb vaid arvutamise alla käivate tegevustega, kuid jätab võimaluse tulevikus rakendusse lisada ka laevapileti broneerimise funktsionaalsuse. Seda selleks, et keskenduda suuremale murekohale, milleks on luua ammendav arvutamisloogika. Broneerimiseloogika peab igal sellisel vedude planeerimisteenust pakkuval firmal juba olemas olema, kasvõi iseseisva rakendusena, sest muidu pole võimalik seda teenust pakkuda.

Lõputöö kontekstis piirdutakse vaid veebirakendusega, eelkõige seetõttu, et see on kõige mugavam viis, kuidas kasutaja saab sellele ligi. Veebirakendust ei pea installeerima ja see ei sõltu platvormist [9].

Lõputöö eesmärk on esialgu luua lahendus ühele ettevõttele, mida saab hiljem pakkuda ka teistele samas valdkonnas tegutsevatele ettevõtetele. Seetõttu on tehnoloogilised valikud mõjutatud selle ühe ettevõtte eelistustest.

4 Loodava veebirakenduse analüüs

Selles peatükis analüüsitakse erinevaid tehnoloogiaid erinevate rakenduse osade jaoks ja põhjendatakse valikuid. Lisaks sellele on funktsionaalsete ja mittefunktsionaalsete nõuete alusel loodud kasutajalood ja funktsionaalsete nõuete põhjal loodud kasutajalugude juurde on lisatud ka protsessi illustreerivad plokkskeemid.

4.1 Nõuete määramine

Nõuete määramiseks on viidud läbi intervjuu. Nõuete määramisel on arvestatud, et veebirakendust hakatakse kasutama Eesti piires.

Nõuded põhinevad kasutajalugudel, mis jagunevad kolme kasutajagruppi:

- Tavakasutaja
- Ettevõtte administraator
- Veebirakenduse administraator

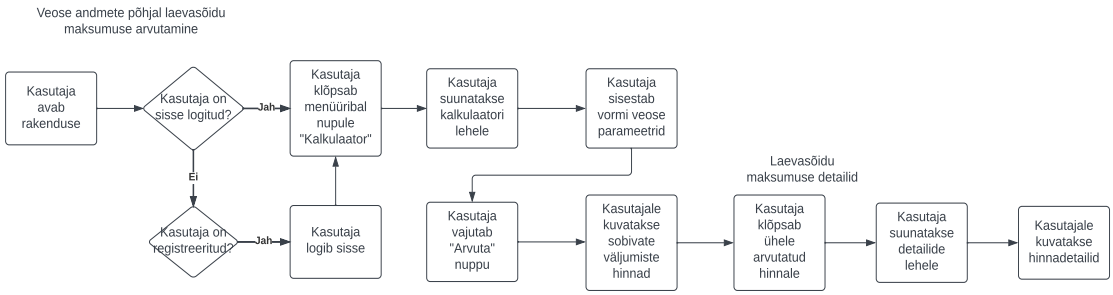
Tavakasutaja all mõeldakse kasutajat, kes kasutab oma igapäevases töös rakendust arvutamiseks, hinnakirjade ja sõidugraafikute vaatamiseks. Ettevõtte administraator on kasutaja, kellel lisaks eelnevale on õigus luua ja muuta sisestatud andmeid ning hallata enda ettevõtte kasutajaid ja nende õiguseid. Veebirakenduse administraator on kasutaja, kellel on õigus luua ja kustutada kasutajaid ning nende õiguseid.

4.1.1 Kasutajalood funktsionaalsete nõuete põhjal

Tavakasutaja-põhised kasutajalood funktsionaalsete nõuete põhjal:

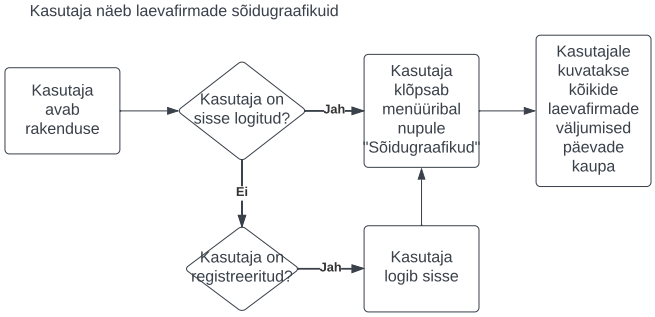
- „Tavakasutajana tahan arvutada veose andmete põhjal laevasõidu maksumuse“ (Joonis 1). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto, siis pärast sisse logimist klõpsab ta menüüribal nupule „Kalkulaator“. Kasutajale kuvatakse vorm, kuhu peab sisestama arvutamiseks vajalikud andmed. Pärast nupu „Arvuta“ vajutamist, kuvatakse kasutajale kõik sobivatele väljumistele arvutatud hinnad väljumisgraafikute kujul.

- „Tahan näha arvutatud laevasõidu maksumuse detaile“ (Joonis 1). Pärast hindade arvutamist, saab kasutaja näha iga arvutatud hinna detaile, klõpsates ühele soovitud hindadest. Kasutaja suunatakse detailide lehele, kus kasutajale kuvatakse hinnadetailid tabeli kujul.



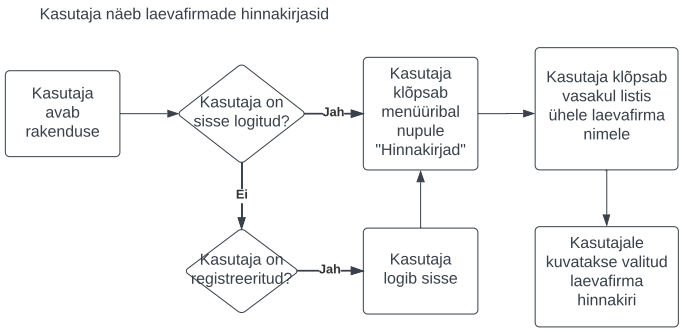
Joonis 1. Veose Andmete põhjal laevasõidu maksumuse arvutamise ja maksumuse detailide nägemise plokkskeem.

- „Tahan näha laevafirmade sõidugraafikuid“ (Joonis 2). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto, siis pärast sisse logimist klõpsab ta menüüribal nupule „Sõidugraafikud“. Kasutajale kuvatakse kõikide laevafirmade väljumised päevade kaupa.



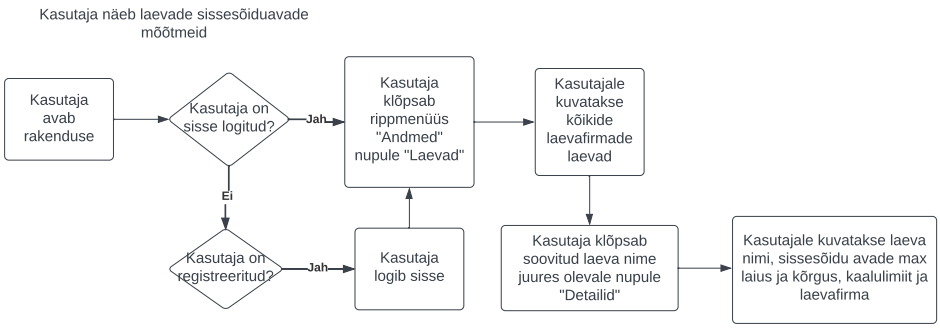
Joonis 2. Sõidugraafikute nägemise plokkskeem.

- „Tahan näha laevafirmade hinnakirjasid“ (Joonis 3). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto, siis pärast sisse logimist klõpsab ta menüüribal nupule „Hinnakirjad“. Kasutajale kuvatakse vasakul pool lehte loend laevafirmasid, mille hinnakirjadega on tal võimalik tutvuda. Kasutaja klõpsab ühele laevafirma nimele ja talle kuvatakse paremale lehepoolele valitud laevafirma hinnakiri.



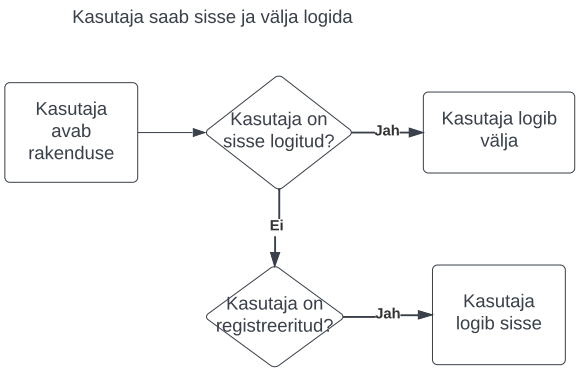
Joonis 3. Hinnakirjade nägemise plokkskeem.

- „Tahan näha laevade sissesõiduavade mõõtmeid“ (Joonis 4). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto, siis pärast sisse logimist klõpsab ta rippmenüüs „Andmed“ nupule „Laevad“. Kasutajale kuvatakse loend kõikide laevafirmade laevadest. Kasutaja klõpsab soovitud laeva nime juures olevale nupule „Detailid“ ja kasutajale kuvatakse leht, kus on võimalik näha laeva nime, sissesõidu avade maksimaalne laius ja kõrgus, kaalulimiit ja laevafirma, kellele laev hetkel kuulub.



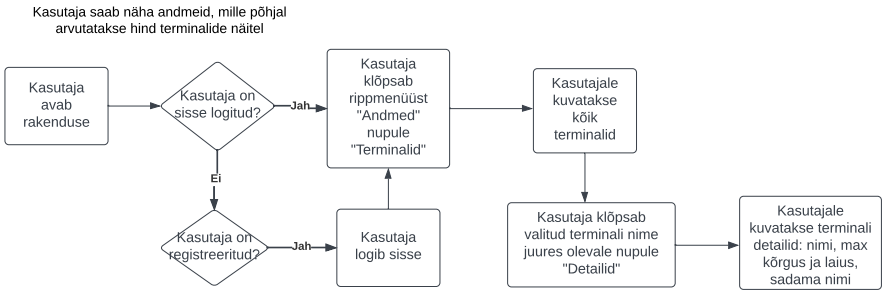
Joonis 4. Laevade sissesõiduavade mõõtmete nägemise plokkskeem.

- „Tahan veebirakendusse sisse ja välja logida“ (Joonis 5). Kasutaja avab rakenduse ja kui ta on juba sisse logitud, siis ta saab logida välja või kui tal eksisteerib konto, siis saab ta sisse logida.



Joonis 5. Rakendusse sisse- ja väljalogimise plokkskeem.

- „Tahan näha andmeid, mille põhjal arvutatakse hind“ (Joonis 6). Plokkskeem on tehtud terminalide näitel. Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto, siis pärast sisse logimist klõpsab ta rippmenüüs „Andmed“ nupule „Terminalid“. Kasutajale kuvatakse kõik terminalid ja kasutaja klõpsab soovitud terminali nime juures olevale nupule „Detailid“. Kasutajale kuvatakse leht, kus on näha terminali nimi, maksimaalne lubatud veose kõrgus ja laius ja sadama nimi, kus valitud terminal asub.



Joonis 6. Arvutamiseks vajalikke andmete nägemise plokkskeem terminalide näitel.

Ettevõtte administraatori-põhised funktsionaalsed nõuded:

- „Tahan lisada/muuta/kustutada andmeid, mille põhjal arvutatakse hind“ (Lisa 1). Plokkskeem on tehtud sadama näitel Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto, siis pärast sisse logimist klõpsab ta rippmenüüs „Andmed“ nupule „Sadamad“. Kasutajale kuvatakse kõik sadamad loendina. Kui kasutajal on veebirakenduse administraatori või

ettevõtte administraatori õigused, siis on tal iga sadama juures valikud lisaks valikule „Detailid“ valikud „Muuda“ ja „Kustuta“. Kui kasutaja soovib lisada uue sadama, siis vajutab ta nupule „Lisa“ ja kasutajale kuvatakse vorm uue kirje loomiseks. Kui kasutaja soovib seda salvestada, siis ta vajutab nupule „Salvesta“ ja ta suunatakse tagasi sadamate loendi lehele, kui ta ei soovi uut kirjet lisada vajutab ta „Tühista“ ja kasutaja suunatakse tagasi loendi lehele. Kui kasutaja soovib kirjet kustutada, siis ta klõpsab nupule kustuta ja kirje kustutatakse. Kui kasutaja soovib mingit kirjet muuta, siis klõpsab ta soovitud sadama nime juures olevale nupule „Muuda“ ja kasutajale kuvatakse vorm sadama andmetega. Pärast kirje muutmist saab kasutaja muudatused salvestada klõpsates nupule „Salvesta“ või muudatuste mitte salvestamiseks nupule „Tühista“ ja kasutaja suunatakse tagasi sadama loendi lehele. Kui kasutaja soovib näha mingi kirje detaile, klõpsab ta valitud sadama nime juures olevale nupule „Detailid“ ja kasutajale kuvatakse valitud sadama detailid. Kui ta soovib antud kirjet muuta, siis klõpsab ta nupule „Muuda“ ja kasutajale suunatakse kirje andmete muutmise lehele.

- „Tahan hallata enda ettevõtte kasutajaid ja nende õiguseid“ (Lisa 2). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto ja tal on ettevõtte administraatori õigused, siis pärast sisse logimist klõpsab ta enda ettevõtte kasutajate õiguste haldamiseks rippmenüüribal „Admin“ nupule „Kasutajad“. Antud kasutajal on ettevõtte administraatori õigused ehk kuvatakse ainult tema ettevõtte kasutajad. Iga kasutaja nime juures on valikud „Detailid“, „Muuda“ ja „Kustuta“. Kui kasutaja soovib mingi kasutaja andmeid muuta, siis klõpsab ta soovitud kasutaja nime juures olevale nupule „Muuda“ ja kasutajale kuvatakse vorm kasutaja andmetega. Pärast kirje muutmist saab kasutaja muudatused salvestada klõpsates nupule „Salvesta“ või muudatuste mitte salvestamiseks nupule „Tühista“ ja kasutaja suunatakse tagasi kasutajate loendi lehele. Kui kasutaja soovib näha mingi kasutaja andmeid, klõpsab ta valitud kasutaja nime juures olevale nupule „Detailid“ ja kasutajale kuvatakse valitud kasutaja detailid. Kui ta soovib antud kirjet muuta, siis klõpsab ta nupule „Muuda“ ja kasutajale suunatakse kasutaja andmete muutmise lehele.

- „Tahan luua ja kustutada enda ettevõtte kasutajaid“ (Lisa 2). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto ja tal on ettevõtte administraatori õigused, siis pärast sisse logimist klõpsab ta enda ettevõtte kasutajate õiguste haldamiseks rippmenüüribal „Admin“ nupule „Kasutajad“. Antud kasutajal on ettevõtte administraatori õigused ehk kuvatakse ainult tema ettevõtte kasutajad. Kui kasutaja soovib luua uue kasutaja, siis ta klõpsab nupule „Lisa“ ja kasutajale kuvatakse vorm uue kasutaja loomiseks. Antud kasutajal on ettevõtte administraatori õigused ehk uuele kasutajale määratakse sama ettevõte nagu on antud kasutajal. Kui kasutaja soovib loodud kasutajat lisada, siis vajutab ta nupule „Salvesta“ ja uue kasutaja loomisest loobumiseks nupule „Tühista“ ja kasutaja suunatakse tagasi kasutajate loendi lehele. Kui kasutaja soovib mingit kasutajat kustutada, siis ta klõpsab nupule kustuta ja kasutaja kustutatakse.

Veebirakenduse administraatori-põhised funktsionaalsed nõuded:

- „Tahan hallata kasutajaid ja nende õiguseid“ (Lisa 2). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto ja tal on veebirakenduse administraatori õigused, siis pärast sisse logimist kasutajate ja nende õiguste haldamiseks klõpsab ta enda ettevõtte kasutajate õiguste haldamiseks menüüribal „Admin“ nupule „Kasutajad“. Antud kasutajal on veebirakenduse administraatori õigused ehk kõik kasutajad. Iga kasutaja nime juures on valikud „Detailid“, „Muuda“ ja „Kustuta“. Kui kasutaja soovib mingi kasutaja andmeid muuta, siis klõpsab ta soovitud kasutaja nime juures olevale nupule „Muuda“ ja kasutajale kuvatakse vorm kasutaja andmetega. Pärast kirje muutmist saab kasutaja muudatused salvestada klõpsates nupule „Salvesta“ või muudatuste mitte salvestamiseks nupule „Tühista“ ja kasutaja suunatakse tagasi kasutajate loendi lehele. Kui kasutaja soovib näha mingi kasutaja andmeid, klõpsab ta valitud kasutaja nime juures olevale nupule „Detailid“ ja kasutajale kuvatakse valitud kasutaja detailid. Kui ta soovib antud kirjet muuta, siis klõpsab ta nupule „Muuda“ ja kasutajale suunatakse kasutaja andmete muutmise lehele.

- „Tahan luua ja kustutada kasutajaid ja õiguseid“ (Lisa 2). Kasutaja avab rakenduse ja kui ta on juba sisse logitud või tal eksisteerib konto ja tal on

ettevõtte administraatori õigused, siis pärast sisse logimist klõpsab ta enda ettevõtte kasutajate õiguste haldamiseks menüüribal „Admin“ nupule „Kasutajad“. Antud kasutajal on ettevõtte administraatori õigused ehk kuvatakse ainult tema ettevõtte kasutajad. Kui kasutaja soovib luua uue kasutaja, siis ta klõpsab nupule „Lisa“ ja kasutajale kuvatakse vorm uue kasutaja loomiseks. Antud kasutajal on ettevõtte administraatori õigused ehk ta saab uuele kasutajale ise määrata ettevõtte. Kui kasutaja soovib loodud kasutajat lisada, siis vajutab ta nupule „Salvesta“ ja uue kasutaja loomisest loobumiseks nupule „Tühista“ ja kasutaja suunatakse tagasi kasutajate loendi lehele. Kui kasutaja soovib mingit kasutajat kustutada, siis ta klõpsab nupule kustuta ja kasutaja kustutatakse. Kui kasutaja soovib hallata õiguseid siis klõpsab ta rippmenüüribal „Admin“ nupule „Õigused“. Kui kasutaja soovib luua uue õiguse, siis ta klõpsab nupule „Lisa“ ja kasutajale kuvatakse vorm uue õiguse loomiseks. Kui kasutaja soovib loodud õigust salvestada, siis vajutab ta nupule „Salvesta“ ja uue õiguse loomisest loobumiseks nupule „Tühista“ ja kasutaja suunatakse tagasi õiguste loendi lehele. Kui kasutaja soovib mingit õigust kustutada, siis ta klõpsab nupule kustuta ja kui ühelgi kasutajal valitud õigust pole, siis see õigus kustutatakse, muul juhul kuvatakse kasutajale veateada, et kirjet ei saa kustutada.

4.1.2 Kasutajalood mittefunktsionaalsete nõuete põhjal

Tavakasutaja-põhised kasutajalood mittefunktsionaalsete nõuete põhjal:

- Tavakasutajana tahan, et kalkulaatori kasutamine oleks lihtne ja loogiline.
- Tavakasutajana tahan, et lehe kasutamine oleks intuitiivne.
- Tavakasutajana tahan, et lehe välimus oleks lihtne.
- Tavakasutajana tahan, et lehte oleks võimalik kasutada ka eestikeelsena.

Ettevõtte administraatori-põhised mittefunktsionaalsed nõuded:

- Ettevõtte administraatorina tahan, et minu ettevõtte andmed ja laevafirmadega sõlmitud hinnakirjad oleksid turvalised ja kättesaadavad ainult minu enda firma kasutajatele.

- Ettevõtte administraatorina tahan, et lehte oleks võimalik kasutada ka eestikeelsena.

4.2 Tehnoloogiate valik

Veebirakenduse arendamisel on tänapäeva arendajal suur tehnoloogiate valik, mille vahel valida. Enne arenduse algust on vaja selgeks teha, kui suur saab olema rakenduse kasutajaskond ja kui keeruline on veebirakenduse ehitus. Mida suurem ja keerukam on veebirakendus, seda rohkem erinevaid raamistike ja programmeerimiskeeli on vaja kasutada erinevate rakenduse osade arendamise jaoks [10].

Tehnoloogiate valimisel on oluline arvestada järgmiste kriteeriumitega [11]:

- Populaarsus ja kogukonna suurus – mida suurem on populaarsus ja kogukonna suurus, seda tõenäosem on, et antud tehnoloogia on pidevas arengus ja probleemide korral on lihtsam leida lahendusi.

- Jätkusuutlikus – tuleb silmas pidada, et suudetakse kaasas käia tehnoloogia pideva arenguga, mis muudab rakenduse haldamise ja uuendamise lihtsamaks.

- Turvalisus – ükski tehnoloogia pole täielikult turvaline ja seetõttu tuleb valida tehnoloogiad, mida täiustatakse pidevalt ja raamistikke, mis pakuvad enda loodud lahendusi, et kasutaja ei peaks hakkama neid ise välja mõtlema. Kasutaja välja mõeldud lahendused ei ole tõenäoliselt nii turvalised kui raamistikke poolt pakutud.

- Dokumentatsioon – hästi dokumenteeritud tehnoloogiat on lihtsam kasutada ja annab eeltöö tegemisel selgema arusaama, mida see tehnoloogia võimaldab ja mida mitte.

- Litsents – veebirakenduse kasutamine võib olla piiratud, kui kasutatud tehnoloogiatel on piiravad litsentsid. Probleemide ennetamiseks tuleb veenduda, et antud tehnoloogia litsents lubab veebirakendust kasutada soovitud viisil.

- Kogemus – kasutaja enda vähene kogemus tehnoloogiaga on tihti piiravam kui kõik eelnevalt loetletu. Tehnoloogia valimisel tuleb otsustada, kas valida

tehnoloogia, mida osatakse hästi, aga ei ole võib olla kõige parem valik või valida tehnoloogia, mis on tundmatu, aga sobiks paremini probleemi lahendamiseks. Tundmatu tehnoloogia omandamisele kuluv aeg omakorda pikendab arendusprotsessi.

4.2.1 Relatsiooniline vs mitterelatsiooniline andmebaas

Hea rakenduse vundamendiks on kiire andmete päring ja töötlus [12]. Seetõttu on oluline leida enda rakendusele sobiv andmebaasisüsteem ja otsustada, kas rakendusele sobib paremini relatsiooniline või mitterelatsiooniline andmebaas.

Enne andmebaasisüsteemi valikut tuleb valida, kas luua relatsiooniline ehk SQL (*Structured Query Language*) või mitterelatsiooniline ehk NoSQL andmebaas.

Relatsioonilise andmebaasi mudel koosneb relatsioonide kogumist, mille käsitluskeel on SQL, mida kasutatakse andmestruktuuride loomiseks, kirjeldamiseks, pärimiseks ja haldamiseks [13]. Relatsiooniline andmebaas on laialdaselt kasutusel, hästi dokumenteeritud ja on kasutatav paljude tänapäevaste raamistikega. Kolm kõige populaarsemat SQL andmebaasi on Oracle, MySql ja Microsoft SQL Server ehk MSSQL [14].

Relatsiooniliste andmebaaside plussiks on andmekaitse. Andmebaasis olevatele andmetele ei saa ligi ilma andmebaasi kontota ja vastavate õigusteta, nii on kindel, et keegi väljastpoolt niisama lihtsalt andmeid varastada ei saa. Andmebaasi administraator saab erinevatele kasutajatele anda erinevaid õigusi, näiteks on ühel kasutajal ainult SELECT õigus ehk õigus näha andmeid, aga teisel õigused SELECT, UPDATE ja CREATE ehk õigus pärida, uuendada ja luua andmeid.

Relatsioonilised andmebaasimootorid järgivad *Atomicity, Consistency, Isolation, Durability* ehk ACID standardit, mis tagavad andmebaasi transaktsioonide töökindluse ja aitavad vältida andmekadu ning andmete rikkumist [15].

Relatsioonilist andmebaasi ei saa kasutada efektiivselt andmetega, mis on pooleldi struktureeritud või struktureerimata. Koos andmestruktuuri keerukuse kasvuga, muutub keerukamas ka andmevahetus teiste *data-driven* rakendustega. Lisaks sellele tuleb suurte andmemahtude korral investeerida füüsilistesse seadmetesse, sest relatsioonilist andmebaasi hoitakse ühes serveris. [12]

Mitterelatsioonilises andmebaasisis saab hoida struktureerimata andmeid nagu näiteks pilte ja MP3 (*MPEG Audio Layer-3*) helifaile. Seda tüüpi andmebaasi on ka lihtsam skaleerida, sest sellist andmebaasi saab hoida mitmes erinevas serveris ja tänu sellele suudab see püsida töös ka siis, kui näiteks üks server on maas.

Mitterelatsiooniline andmebaas pole nii laialt levinud kui relatsiooniline andmebaas ja selle taga ei ole ka väga suurt kogukonda [16]. Paindlikkuse tõttu tihti loobutakse järgimast ka ACID standardit.

Lõputöö autor on valinud enda lahenduse jaoks relatsioonilise andmebaasi, kuna loodava lahenduse jaoks on oluline, et andmed oleksid struktureeritud, sest see võimaldab arvutamisel pärida vajalikke andmeid SQL lausetega. Lisaks sellele on oluline, et oleks tagatud andmekaitse, sest ühe ettevõtte andmeid peaks nägema ainult selle ettevõtte töötajad.

4.2.2 Andmebaasisüsteem

Andmebaasisüsteem on tarkvara, millega saab luua ja hallata andmebaase ja on vahendajaks andmebaasi ja klientrakenduse vahel [17]. Stack Overflow läbiviidud küsitluses [18] selgus, et professionaalsete arendajate seas olid 2021. aastal kõige populaarsemad relatsioonilised andmebaasisüsteemid MySQL, PostgreSQL, SQLite ja Microsoft SQL Server.

MySQL andmebaasisüsteem, mida toetab Oracle, on avatud lähtekoodiga ja kasutajatele tasuta kasutamiseks GNU (*General Public License*) litsentsiga. Võrreldes Microsoft SQL Serveriga on MySQL andmebaasisüsteem kiire, skaleeritav ja lihtsasti kasutatav. Tihti kasutatakse seda teenusepoolses rakenduses koos PHPga. See on populaarne, kuna see on kiirem kui teised andmebaasisüsteemid ja tagab hea jõudluse ka suurte andmemahtude korral ning seda saab kasutada koos erinevate teenuspoolsete tehnoloogiatega. [19]

PostgreSQL on populaarne avatud lähtekoodiga variant suurtematele ettevõttetele, mis toetab nii SQL kui ka JSON (*JavaScript Object Notation*) päringuid [20]. Rohkem kui 20 aastat arendamist, on teinud sellest ühe kõige täiustatuma andmebaasisüsteemi. Seda kasutavad paljud veebi- ja mobiilirakendused ja analüütilised rakendused oma andmete hoidmiseks. Võrreldes MySQLiga toetab PostgreSQL rohkem erinevaid andmetüüpe ja tagab parema jõudluse rakendustele, kus lisaks paljudele SELECT päringutele tehakse ka

CREATE ja UPDATE päringuid [21]. PostgreSQL on populaarne suuremate ettevõtete seas just seetõttu, et mitu klienti saavad samaaegselt andmebaasi poole pöörduda, sest igale uuele kliendile, kes andmebaasiga ühendub, eraldatakse mälu ja algatatakse uus protsess [22].

SQLite on avatud lähtekoodiga andmebaasisüsteem, mis on disainitud töötama ilma andmebaasi administraatorita [23]. See on loodud toimima rakendusesiseselt ehk andmebaasi ei hoita serveris vaid kettal ja selle üles seadmine pole keeruline, rakendustele on lihtsalt vaja anda juurdepääsuõigus kettale. SQLite puuduseks on see, et see toetab väheseid andmetüüpe ning sellel puudub andmebaasisisene autentimine, mistõttu on kõigil andmetele ligipääs võrreldes MySQLi ja PostgreSQLiga. Lisaks sellele peab SQLite'i andmebaasi kasutades kõik andmed mahtuma ühte faili, mistõttu on see sobilik ainult väikeste andmebaasidega töötamiseks ja kuna sellel puudub ka kasutajahaldus süsteem, siis saab see teenindada ühte klienti korraga [24].

Microsoft SQL Server ehk MS SQL on Microsofti poolt pakutud andmebaasisüsteem. Võrreldes eelnevalt väljatoodud andmebaasisüsteemidega on see andmebaasisüsteem tasuline ja mille hind sõltub andmebaasi suurusest [25]. Litsentside hinnad hakkavad 3241,56 eurost ja suurtele ettevõtetele võib see hind tõusta kuni 12281,06 euroni. Lisaks sellele peab igale andmebaasi kasutavale kliendile lisaks ostma *Client Access License* ehk CAL litsents, mis maksab 188,93 eurot. Tasuta versioonid on olemas tudengitele ja arendajatele arendamiseks. MS SQL on hea skaleeritavusega, kasutajatele mugav kasutada ja töökindel ning ühildub hästi .NETiga (tarkvararaamistik). [26]

Andmebaasisüsteemi valimisel oli autorile kui tudengile eelkõige oluline selle maksumus ja seetõttu langes valikust kohe välja MS SQL. Andmebaasisüsteemiks on autor valinud MySQLi, sest see on hästi dokumenteeritud ja tänu sellele on tõenäosem probleemide korral kiiresti lahendus leida. Lisaks sellele sobib see kasutamiseks ka suurte andmemahtude korral, kuna rakenduse toimimiseks on vaja andmebaasis hoida palju andmeid.

4.2.3 Teenusepoolne programmeerimiskeele valik

Kõige levinumad teenusepoolsed programeerimiskeeled on JavaScript, Python, TypeScript, Java ja C# [18]. Programmeerimiskeele valikust sõltub ka raamistike valik

ehk enne keele valimist tuleb kindlasti tutvuda ka potentsiaalsete raamistikega. Teenusepoolseid raamistike kirjeldatakse järgmises peatükis.

JavaScript on *object-based* skriptimiskeel (sisseehitatud objektidega keel), mida saab kasutada nii kliendipoolse rakenduse loomiseks kui ka teenuspoolse rakenduse arenduseks [27]. JavaScriptis saab programmeerida objekt-orienteeritult. JavaScriptiga teenusepoolse rakenduse arenduseks kasutatakse enamasti raamistiku Node.js [28]. Täielikult JavaScriptis loodud rakendused on kiired ja ei vaja kompileerimist enne selle käitamist. JavaScripti süntaks on sarnane Java omale ja seda keelt on suhteliselt lihtne õppida võrreldes näiteks C++'iga. Lihtsamad JavaScripti rakendused ei pruugi vajadagi serverit, sest JavaScript on kliendipoolne ehk skriptid käitavad brauseris, kuid seda saab jällegi pahatahtlikult ära kasutada [29].

Python on dünaamiline objekt-orienteeritud keel, millel on hästi loetav süntaks ja tänu sellele on seda kerge õppida [30]. Võrreldes Javaga saab Pythonis vähesemate koodiridadega luua sama funktsionaalsuse. Lisaks sellele on Pythoniga võimalik kasutada palju erinevaid teeke ja raamistike [31].

TypeScript on objekt-orienteeritud programmeerimiskeel, mis kompileerub JavaScriptiks [27]. Seda keelt saab kasutada nii teenus- kui ka kliendipoolse rakenduse loomiseks. TypeScript võimaldab lisada staatilist tüübikirjeldust ja kuna seda kompileeritakse, siis, erinevalt JavaScriptist, saab juba kompileerimise ajal teada vigadest [32].

Java on samuti objekt-orienteeritud keel, mille süntaks on inspireeritud C++'ist [33]. Selle keele plussiks on see, et Java on mitmekülgse kasutusega ja Javas loodud koodi saab käitada ükskõik millisel Java toetusega platvormil. Tänu sellele on Javas võimalik arendada veebirakendusi nii suurtele ettevõtetele kui ka väiksemaid Androidi mobiilirakendusi [31]. Java rakenduste nõrgaks kohaks on jõudlus ja liigne mälu tarbimine, sest rakendust interpreteeritakse käitamise ajal abil *Java Virtual Machine*'i ehk JVMiga [33].

C# on objekt-orienteeritud keel, mille on loonud Microsoft. Enamik Windowsi desktop-rakendusi on kirjutatud just C#'iga. C#'iga sobib hästi kokku .NET raamistik, mis pakub laia valiku teeke, tänu millele saab kirjutada platvormist sõltumatuid rakendusi. C#'i süntaks on sarnane Java omale, kuid selle keele õppimine on Javast veidi keerulisem. C# on hästi dokumenteeritud suure kasutajaskonnaga keel. Keele puuduseks on selle sõltuvus

.NET *Core* raamistikust, sest ainult C#'iga saab luua rakendusi Windowsi operatsioonisüsteemiga seadmetele. [34]

Järgnevalt on tabelis välja toodud autori kogemus eelmainitud keeltega ja nende õppimiskeerukus.

Tabel 1. Teenuspoolsete programmeerimiskeelte võrdlus.

Keel	Kogemus	Õppimiskeerukus
JavaScript	Rahuldav	Keskmine [35]
Python	Madal	Madal [36]
TypeScript	Hea	Keskmine [35]
Java	Väga hea	Keskmine [36]
C#	Väga hea	Keskmine [35]

Tabelis olevate keeltega on autoril juba mingil määral kogemus olemas, kuid siiski oleks mõistlik kaaluda ainult neid keeli, millega on kõige rohkem kogemust. Arvestades sellega, et vähem tuntud keele valimisel pikeneb ka arendusprotsess, sest läheb aega keele juurde õppimisele. Lõputöö raames pole see mõistlik aja investeering, sest lõputöö valmimisel on kindel tähtaeg, millest üle ei tohi minna. Seetõttu jäävad valikusse Java ja C#, millega mõlemal on autoril väga hea kogemus.

Nii Javal kui C#'il mõlemal on lai teekide valik ja suur kasutajaskond, kuid lisaks sellele on C#'is suur valik erinevaid valmislahendusi, mida saab enda lahendustes kasutada [37]. C#'i ja .NETi eeliseks on autori parem kogemus ASP.NET (*Active Server Pages Network Enabled Technologies*) *Core*'iga klientrakenduse loomisel. Lisaks sellele on on C#'is loodud rakendus paremini ühilduv Kaarlaidis juba olemasolevate rakendustega.

4.2.4 Teenusepoolne raamistiku valik

Raamistikud annavad rakendustele põhja ja teevad arenduse lihtsamaks, pakkudes geneerilisi valmislahendusi, mida arendaja saab rakendada ja muuta oma äranägemise järgi [38]. Teenusepoolse raamistiku valik sõltub tugevalt programmeerimiskeele

valikust ehk üldjuhul määrab programmeerimiskeel ära juba raamistikud, mida saab arenduseks kasutada.

Node.js on avatud lähtekoodiga JavaScriptil põhinev populaarne *full-stack* raamistik ehk sellega saab luua nii kliendipoolse kui ka teenusepoolse rakenduse. Seda raamistiku saab kasutada nii JavaScripti kui ka TypeScriptiga. Node.js'i eeliseks on see, et ta ei nõua palju mälu ja ressursse ning selle taga on suur kogukond, kes panustavad raamistiku pidevasse arengusse [39].

Kõige populaarsemad Pythoni raamistikud on Flask ja Django [18]. Flask on Pythonil-põhinev mikroraamistik, mida kasutatakse veebirakenduste arenduseks. Mikroraamistik on raamistik, mis ei sõltu raamistiku välistest teekidest ja pakub kogu funktsionaalsuse ise. Paljud arendajat eelistavadki alustada Flaskiga, kuna see on iseseisev ja paindlik. Django on Pythonil-põhinev avatud lähtekoodiga *full-stack* raamistik, mida samuti kasutatakse veebirakenduste arenduseks. Django võimaldab lahenduse kiiret arendust ja soosib pragmaatilist disaini. Django raamistik tagab hea rakenduse turvalisuse raamistiku pakutavate autentimislahendustega. [40]

Spring on kõige tuntum Javal põhinev raamistik. Springi raamistik ei nõua suurt mälumahtu ja võimaldab luua hea jõudlusega ning lihtsasti testitavat koodi [41]. Springi raamistik võimaldab *Dependency Injection*'it ehk sõltuvuste süstimist, mis võimaldab hoida erinevaid koodi osasid eraldi, vähendades koodi hulka ja lisades paindlikust. Springi rakendusse saab lisada väliseid tööriistu Gradle või Maveniga.

.NET *Core* on uuem versioon .NET *Framework*'ist, mis on Microsofti loodud avatud lähtekoodiga raamistik. Raamistik sisaldab põhilisi funktsionaalsusi rakenduse töötamiseks, kuid võimaldab saada lisafunktsionaalsusi NuGet paketihalduriga. Erinevalt .NET *Framework'*ist, millega saab luua rakendusi ainult Windowsi operatsioonisüsteemiga seadmetele, ei sõltu .NET *Core* platvormist. .NET *Core*'iga on võimalik luua nii mobiili- kui ka veebirakendusi. [42]

Järgnevas tabelis on välja toodud autori kogemus raamistikega ja raamistike turvalisuse hinnang erinevate allikate põhjal. [43] – [49]

Tabel 2. Teenuspoolsete raamistike võrdlus.

Raamistik	Kogemus	Turvalisus
Node.js	Puudub	Rahuldav
Flask	Puudub	Kasin
Django	Puudub	Väga hea
Spring	Hea	Hea
.NET *Core*	Hea	Väga hea

Võttes aluseks nii eelnevat tabelit kui ka eelmises peatükis tehtud järeldusi, on mõistlik valida .NET *Core* raamistik. Selle raamistikuga on autoril eelnevalt juba kogemust ja see ühildub hästi Kaarlaidis juba olemasolevate lahendustega, mis tuleb kasuks edasiarendusel või programmide kokkuviimisel.

4.2.5 Kliendipoolse tehnoloogia valik

Kliendipoolse tehnoloogia valik on eriti oluline, kuna see on üks esimesi asju, mida kasutaja näeb. Seetõttu on arendajatele oluline luua funktsionaalne ja kasutajatele meeldiv UI (*User Interface*).

React on vabavaraline JavaScripti teek UI arendamiseks. React võimaldab luua komponente, mis teeb arenduse lihtsamaks ja aitab vältida koodikordust. Seda on lihtne õppida, kui omatakse varasemat JavaScripti kogemust. React on kiiresti muutuv ja arenev, mistõttu ei pruugi dokumentatsioon olla asjakohane [50]. Paljud suurfirmad kasutavad seda enda rakendustes nagu näiteks PayPal, Netflix ja Tesla [51].

JQuery on samuti vabavaraline JavaScripti teek, mille motoks on „*write less, do more*“ ehk võimaldab luua sama funktsionaalsuse vähesemate koodiridadega kui näiteks JavaScriptis. Samuti teeb see lihtsamaks sündmuste halduse, DOM (*Document Object Model*) elementide manipulatsiooni ja AJAX’i (*Asynchronous JavaScript and XML*). JQuery’i pole küll eriti hästi dokumenteeritud, kuid sellel on palju pistikprogramme kasutamiseks. Seda on sobilik kasutada lihtsamate ja väiksemate rakenduste jaoks [51]. [52]

Angular on Google poolt loodud vabavaraline JavaScripti ökosüsteem. Angular võimaldab kahepoolset andmesidumist (*two-way data binding*), tänu millele on arendus kiirem, sest pole vaja luua eraldi loogikat, mis kasutajaliidest reaalajas uuendaks. See raamistik toetab ka sõltuvuste süstimist, mis võimaldab komponente taaskasutada ja lihtsustab nende testimist. Angulari puuduseks on madal jõudlus suurte dünaamiliste rakendustega ja järsk õppimiskõver. [53]

ASP.NET *Core* Microsofti poolt loodud vabavaraline raamistik, mis töötab .NET *Core*'i peal. ASP.NET *Core* on kombinatsioon MVCst (*Model-View-Controller*) ja Web APIst (*Application Programming Interface*). See raamistik on kiire ja võimaldab rakendusse sisse integreerida ka muid kasutajaliidese raamistike nagu näiteks Angular ja React. ASP.NET *Core* lihtsustab ka versioneerimist ja võimaldab paralleelselt käitada erinevaid rakenduse versioone. ASP.NET *Core* on võrdlemisi uus raamistik ja seetõttu ei pruugi olla toetust mõnedel kolmandaosapoole teekidel [54]. [55]

Antud lõputöö autor omab kogemust nii Reacti kui ASP.NET *Core*'iga, kuid kuna ASP.NET *Core*'iga saab rohkem funktsionaalsust kui ainult UI loomine ja sobib hästi .NET *Core*'iga kasutamiseks, siis on ASP.NET *Core* eelistatud valik. ASP.NET *Core* lihtsustab MVC'de loomist, mis lihtsustab ja kiirendab arendust.

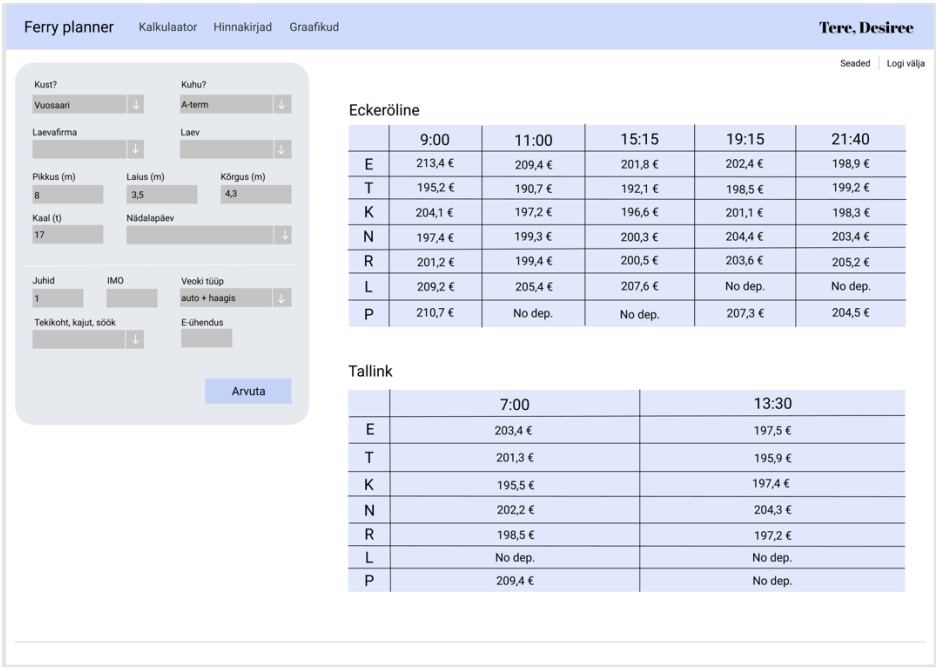
4.3 Veebirakenduse disain

Antud veebirakenduse valmimisel on võetud strateegiaks alustada arvutamise loogikast, kuna see on loodava rakenduse juures kõige olulisem ja see aitab paika panna ka andmebaasi disaini, mis arvutamist võimaldaks.

4.3.1 Arvutamise loogika

Enne arvutamise loogika loomist, tuli tutvuda hinnakirjadega, arvetega ja ettevõtte senise arvutamisloogikaga, et mõista, mis parameetrid arvutamisel on olulised. Arvetel oli selgelt näha, millest teenuse hind koosnes ja milline veose parameeter seda hinda mõjutas. Arvete kujundusest inspireeritult on loodud hinna detailvaate kujundus (Joonis 7).

Ettevõtte senine arvutamisloogika tabeltöötlusega andis põhja autori arvutamisloogikale ja selle põhjal pandi paika kalkulaatori vormi väljad. Hiljem sai senise arvutamisloogika abil veenduda ka uue arvutamisloogika toimimises.



Joonis 7. Kalkulaatori ja arvutustulemuste vaade.

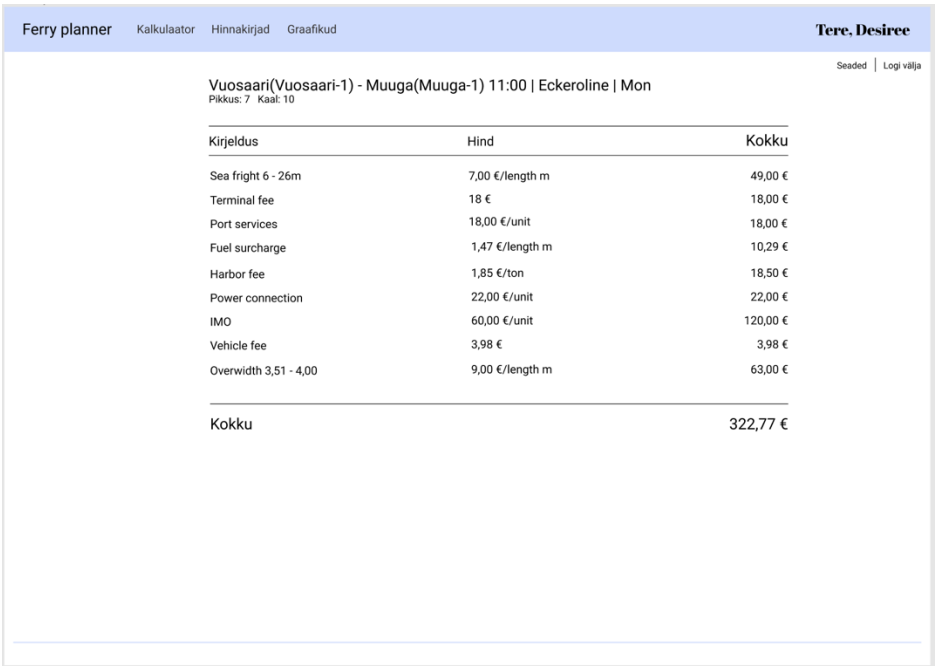
Hinnakirjasid analüüsides selgus, et ühel teenusel võib ühiku hind olla erinev sõltuvalt päevast ja kellajast lisaks laeva ja veose enda parameetritele ehk igal väljumisel on palju erinevaid võimalikke baashindasid. Kõige loogilisem lahendus sellele oli andmebaasis iga väljumine siduda vahetabeli abil ära kõikide hindadega, mis talle kehtiksid ja arvutamise ajal filtreerimise abil leida need hinnad, mis antud kriteeriume rahuldavad. See aga omakorda tähendas seda, et tabelil „Hind“ on palju veerge, mis aitavad filtreerimisel.

Kalkulaatori kohustuslikud väljad on suunad ja veose parameetrid. Kui kasutaja täidab ainult kohustuslikud väljad, siis otsitakse andmebaasist välja kõik ühendused, mis kahe linna vahel on. Näiteks Eckeröline’iga saab Tallinnast Helsinki sõita nii A-terminalist kui ka Muuga sadamast ja Helsinkisse võib randuda kas Länsisatama või Vuosaari sadamasse. Järgmise sammuna on vaja välja filtreerida kõik laevad, millele antud veos mahuks. Kui kasutaja on täpsustanud laeva või laevafirma, siis otsitakse üles vastavalt kas üks laev või kõik ühe laevafirma laevad ja vaadatakse, kas veose kaal ja mõõtmed jäävad alla laevale lubatava maksimumi. Eelnevate sammude tulemuste abil filtreeritakse välja kõik väljumised, mis sõidavad sobivatel suundadel, sobivate laevadega ja kui on täpsustatud ka nädalapäev, siis ka sobival nädalapäeval. Nüüd kui on üles leitud kõik sobivad väljumised, siis arvutatakse igale sobivale väljumisele hind.

Hindade arvutamiseks, päritakse andmebaasist kõik ühe väljumisega seotud hinnad, mida antud veose parameetrid rahuldavad. Näiteks ülelaiustel on mitu erinevat baashinda olenevalt veose laiusest ja arvutamisel võetakse just see hind, mille miinimum ja maksimum laiuse vahele jääb veose laius. Lihtsustamiseks on mõõtmetega hindade ühikud kas „*length m*“, „*height m*“ või „*width m*“, et oleks arusaadav, millise parameetriga antud hinda korrutada tuleb. Mõned teenused on universaalse nimetusega ehk igas hinnakirjas on olemas teenused nagu „Laevateenus“ ja „Elektriühendus“, mis liidetakse lihtsalt kogusummale juurde. Arvutustulemused kuvatakse kasutajale laevafirmade väljumisgraafikute kaupa.

4.3.2 Hinna detailvaade

Lähtuvalt kasutajanõuetest peab iga väljumise kohta olema võimalik näha ka hinna detailvaadet (Joonis 8). Kui kasutaja klõpsab ühele hinnale, siis kuvatakse talle uus leht, kus on välja toodud iga teenuse hind, kogus ja lõplik hind. See võimaldab kasutajal tutvuda, mis teenuste eest raha küsitakse ja hiljem saab selle abil veenduda ka esitatud arvete õigsuses.

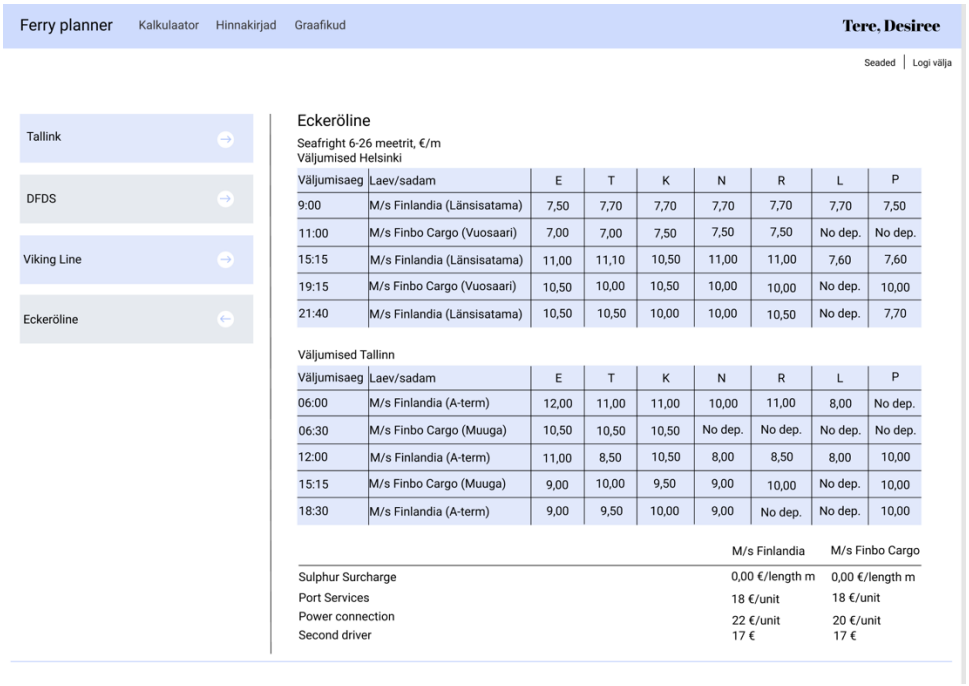


Joonis 8. Arvutatud hinna detailvaade.

4.3.3 Hinnakirjad

Kasutajalugudele tuginedes, peab kasutajatel olema võimalik näha iga sisestatud laevafirma hinnakirjasid. Hinnakirjade vaatele liikudes, peab kasutaja valima laevafirma, kelle hinnakirja ta näha soovib. Hinnakirja vaade on sarnane paberkujul hinnakirja kujundusele.

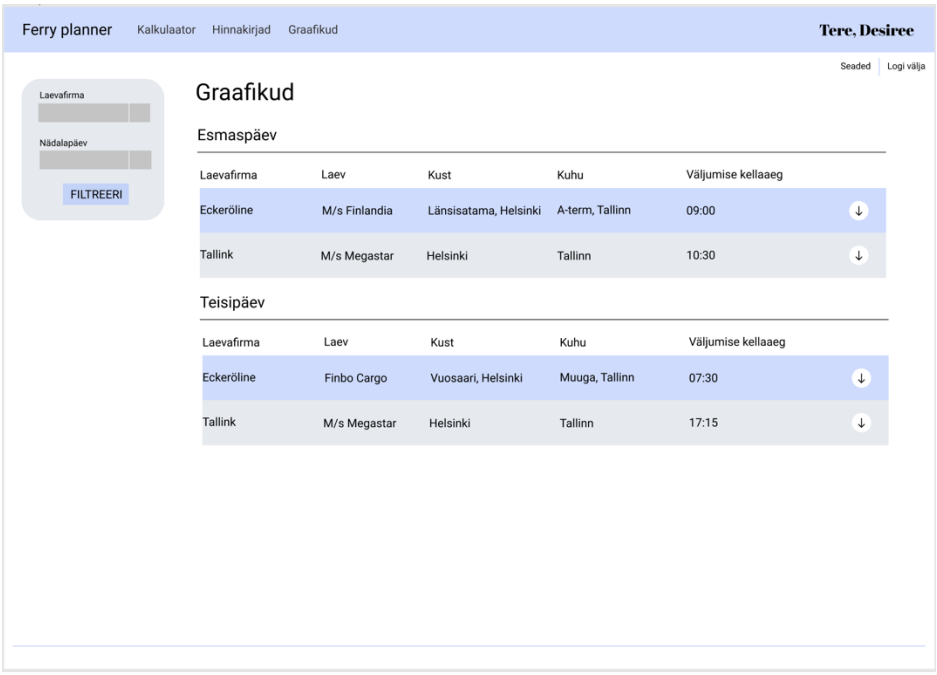
Hinnad on jaotatud kolme osasse: ülelaiused, laevadega soetud hinnad ja muud hinnad. Ülelaiuste hinnad on esitatud suundade kaupa väljumisgraafiku kujul. Laevadega seotud hinnad on esitatud tabeli kujul, et kasutajal oleks hea võrrelda teenuste hindasid erinevatel laevadel. Muud hinnad ehk hinnad, mis pole ülelaiuste hinnad ega laevadega seotud on esitatud ka tabeli kujul. Kui kasutajal on administraatori või ettevõtte administraatori õigused, siis tohib ta sisestada uue hinna või hinnale klõpsatas seda muuta. Hinnakirja vaate kujunduse näide on Joonisel 9.



Joonis 9. Hinnakirja vaade.

4.3.4 Väljumisgraafikud

Väljumisgraafikute vaatele minnes, kuvatakse päevade kaupa kõik väljumised. Tulemuste filtreerimiseks on kasutajal võimalik valida soovitud nädalapäev ja laevafirma. Iga väljumise kohta on näha väljumisaeg, suund, laeva nimi ja laevafirma. Väljumisgraafikute vaate kujundus on näha Joonisel 10



Joonis 10. Väljumisgraafikute vaade

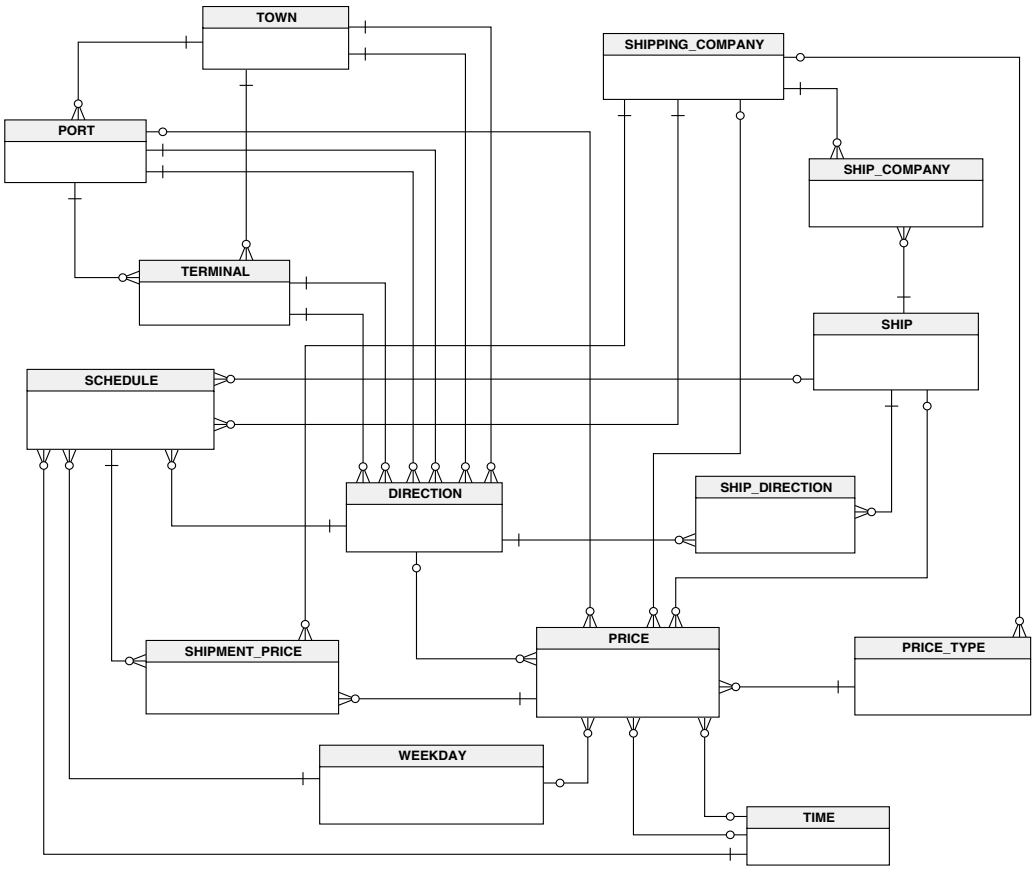
4.3.5 Andmebaasi disain

Andmemudelis tehtud vead on kõige kallimad vead, sest need kanduvad üle loodavasse rakendusse ja kui need vead tulevad hiljem välja, siis tuleb muuta ka palju programmikoodi. Selle vältimiseks tuleb andmemudel koostada läbimõeldult.

Arvutamisloogika paika panemisel tekkis selgem arusaam, milline võiks olla andmebaasi struktuur ja olemite veerud, mis toetaksid arvutamisloogika realiseerimist. Olemi-suhte diagramm on loodud tööriistaga Vertabelo. Täielik olemi-suhte diagramm on nähtav peatükis Lisa 3, Joonisel 11 on kujutatud lihtsustatud olemi-suhte diagrammi, kust on andmebaasi skeemi lugemise lihtsustamiseks eemaldatud „Company“ olem, mis tagab, et iga ettevõtte näeb ainult enda ettevõttega seotud kirjeid.

Tabelisse „Town“ salvestatakse kõik linnad või suuremad asulad, kus asuvad sadamad. Tabelis „Port“ hoitakse kõiki sadamad ja tabelis „Terminal“ on sadamate terminalid. Tabelis „Direction“ on kõik suunad, kus laevad liiguvad ja tabelis „Schedule“ hoitakse väljumisgraafikuid. Tabelis „Price“ hoitakse hindasid ja tabel „Shipment price“ seob need hinnad väljumistega. Tabelis „Weekday“ on olemas nädalapäevad ja tabelis „Time“ hoitakse kellaaegasid, mis on seotud hindade või väljumistega. „Price type“ tabelis hoitakse erinevaid hinnatüüpe, näiteks „Sadama teenused“ või „Ülelaius“. „Shipping company“ tabelis hoitakse laevafirmasid ja tabel „Ship company“ aitab järge hoida

laevade omanikke vahetusel. Tabelis „Ship“ on näha kõik laevad ja tabel „Ship direction“ seob omavahel suundasid ja laevasid.



Joonis 11. Lihtsustatud andmebaasiskeem.

4.4 Analüüsi kokkuvõte

Analüüsis analüüsiti erinevaid kasutajagruppe ja nende kohta käivaid funktsionaalseid ja mittefunktsionaalseid kasutajanõudeid. Kasutajanõudeid analüüsides selgus, et rakendus peaks eksisteerima veebirakenduse kujul.

Rakenduse andmeid hoitakse relatsioonilise mudeliga andmebaasis. Andmebaasisüsteemiks on autor valinud MySQLi, kuna see on vabavaraline ja sobib kasutamiseks erinevate teenuspoolsete tehnoloogiatega. Sellel on hea jõudlus ja sobib töötamiseks ka suurte andmemahtudega.

Teenuspoolse rakenduse keeleks valiti C# koos .NET *Core* raamistikuga, kuna sellega on lõputöö autoril hea kogemus ja enamik Kaarlaidi eksisteerivaid lahendusi on loodud ka

just selles keeles. Lisaks selle on see keel hästi dokumenteeritud ja .NET *Core* pakub palju häid valmislahendusi, mida saab integreerida enda rakendusse.

Kliendipoolse rakenduse arendamise raamistikuks valiti ASP.NET *Core*'i, mis sobib hästi kokku .NET *Core*'iga. ASP.NET *Core* ei ole ainult UI disaini loomiseks, vaid tal on ka lisafunktsionaalsusi nagu näiteks *Scaffolding*, mis lihtsustab Razor'i lehtede ja MVC'de loomist.

5 Veebirakenduse arendus

Veebirakendus on jaotatud kolmeks suuremaks loogiliseks kihiks: andmepöörduskiht ehk DAL (*Data Access Layer*), äraloogikakiht ehk BLL (*Business Logic Layer*) ja esituskiht. Veebirakenduse arendus on jaotatud nelja suuremasse peatükki: andmebaasi arendus, teenuspoolse rakenduse arendus, kliendipoolse rakenduse arendus ja testimine.

5.1 Andmebaasi arendus

Arendus hakkas andmebaasiskeemi loomisega. Rakenduse andmebaas hakkab hoidma kõiki arvutuseks vajalikke andmeid, mistõttu on oluline, et selle struktuur võimaldaks päringuid teha võimalikult optimaalselt ja mugavalt.

Esialgne andmebaasiskeem loodi pärast esmast tutvust ülesandega. Teise intervjuu järgselt sai tehtud sinna mõned muudatused. Näiteks oli esialgses mudelis olemi „laeva ettevõtte“ ja „laev“ vahel üks-mitmele suhe, kuid kui selgus, et laevad vahetavad tihti omanikke, siis oli mõttekam luua kahe olemi vahele vahetabel, millega saab sellel järge pidada.

Arenduse ajaks pandi Dockerisse üles MySQL andmebaas. Andmebaasiskeemi alusel sai loodud andmebaasimudelid, mille põhjal sai EF (*Entity Framework*) *Core*'iga luua andmebaas [56]. Selleks loodi AppDbContext klass tänu millele leiab EF *Core* andmebaasi mudelid ja nende omavahelised seosed. Lisaks selle lihtsustab see ka andmebaasi päringute tegemist, sest EF *Core* kasutab *Language-Integrated Query*'t ehk LINQi. See teeb lihtsaks suhtluse erinevate andmebaasisüsteemidega, sest näiliselt käivad kõik päringud ühtemoodi ja alles hiljem muudetakse need valitud andmebaasi spetsiifilisse süntaksi [57].

5.2 Teenuspoolse rakenduse arendus

Järgmine samm oli luua andmehoidlad, kus tehakse andmebaasipäringuid. Selleks, et vältida koodi kordust ja kirjeldada ära põhilised meetodid, mida andmehoidlad peavad rakendama, sai loodud baasandmehoidla.

Iga andmebaasi tabeli jaoks tehti oma andmehoidla, mis liidese kaudu pärib nii baasandmehoidla meetodeid, kui ka tabeli spetsiifilisi meetodeid. Näiteks hindade ja laevade pärimisel on oluline, et oleksid olemas päringud, mille WHERE lauses saaks täpsustada filtreerivaid parameetreid.

5.2.1 Unit of Work

Andmebaasipäringute töökindluse tagamisest ja andmete rikkumise vältimiseks rakendati *Unit of Work* ehk UOW disainimustrit. See võimaldab teha mitut erinevat andmebaasioperatsiooni ühe transaktsioonina. Kui üks operatsioonidest ei lähe läbi või seda ei saa lõpuni teha, siis võetakse tagasi kõik muudatused, mis selle transaktsiooni sees tehti. Ilma selle disanimustrita tekiksid andmebaasi vigased või poolikud andmed Selle saavutamiseks loodi klass AppUnitOfWork, mis saab sisendiks andmebaasi konteksti instantsi ja loob ise instantse juurde, mida annab edasi andmehoidlatele kasutamiseks.

5.2.2 CRUD lehed

Andmebaasipäringute katsetamiseks ja hiljem andmebaasis olevate andmete haldamiseks loodi CRUD (*Create, Read, Update, Delete*) lehed. *EF Core* võimaldab need Scaffoldingu abil luua. Scaffolding loob iga andmemudeli kohta standardse kontrollerite ja vaadete lahenduse, mille kontrollerid koosnevad kümnest meetodist: viis GET meetodit ja viis POST meetodit. Nende meetodite abil saab pärida kõik kirjeid, ühe kirje, luua uue kirje, muuta olemasolevat kirjet või kustutada eksisteeriv kirje. Algselt kasutavad kontrollerid otse andmebaasi konteksti andmebaasipäringute tegemiseks, kuid need muudeti ümber pöörduma teenuskihi klasside poole.

Nendel lehtedel saab hiljem administraator või ettevõtte administraator luua, muuta või kustutada eksisteerivaid andmeid. Tavakasutajal saab olema õigus ainult näha neid andmeid.

5.2.3 Veebirakenduse turvalisus

Andmete turvalisuse tagamiseks on ASP.NETil *Identity* teek, mis seab ülesse nii autentimise kui ka autoriseerimise. Autentimisel tehakse kindlaks, kas isik on see, kes ta väidab ja autoriseerimisel veendutakse, kas kasutajal on vastavad õigused soovitud tegevuse jaoks. *Identity* teek võimaldab luua kasutajaid ja kasutajagruppe ning siis nende alusel kontrollida kasutajate ligipääsu vaadetele ja meetoditele.

Administraatoritele luuakse eraldi vaated kasutajate, kasutajaõiguste ja kasutajagruppide haldamiseks. Kasutajanõuetele tuginedes on autori loodud veebirakenduses ettevõtte administraatoritel õigus hallata enda ettevõtte töötajaid ja nende õiguseid.

Selleks, et kasutajad pääseksid ligi andmetele, peab nii tavakasutaja kui ka ettevõtte administraator olema seotud ettevõttega. Andmebaasipäringu tegemisel tagastatakse vaid need kirjed, millel on küljes kasutajaga sama ettevõtte ID.

5.2.4 Teenuskihi loomine

Teenuskihi eesmärk on vahendada päringuid esituskihist andmepöörduskihti. See muudab esituskihi sõltumatuks andmete pärimise kujust ehk esituskihti ei pea muutma kui muudetakse andmete pärimise viisi. Selleks on loodud igale andmekihile enda andmemudelid ja andmete liigutamiseks kihtide vahel on loodud andmete teisendaja. Selle abil teisendatakse andmed ühest klassist teise. Lisaks sellele on teenuskihis ka arvutamise äriloogika.

Kui veebirakenduses tehakse päring, mis sisaldab mitut andmebaasipäringut, siis need suunatakse vastavate teenuse poole ja need omakorda kutsuvad välja andmehoidlatest meetodid, mis pöörduvad andmebaasi poole. Teenuskiht vormib andmebaasist saadud andmed esituskihile saatmiseks sobivale kujule ja vastupidi.

5.3 Kliendipoolse rakenduse arendus

.NETi ja ASP.NETiga rakenduse loomisel on teenusepoolne ja kliendipoolne rakendus tihedalt seotud. Näiteks Scaffoldingu tulemusel saadud CRUD vaated olid enamikke andmete sisestamise ja kuvamise jaoks piisavad. Pidi tegema vaid mõningaid muudatusi kujunduses.

Kui andmebaasiga suhtlus oli olemas CRUD lehtede abil, siis sai hakata looma kalkulaatori vaadet. Pärast kontrolleri arendamist sai valmis andmete sisestamisvorm ja siis tulemuste kuvamise tabel, millele omakorda järgnes hinnadetaili vaate ja kontrolleri loomine.

Lisaks sellele muudeti hindade ja väljumiste graafiku vaateid, et need sarnaneksid rohkem hinnakirjale ja sõidugraafikule, mitte ei oleks lihtsalt pikk loend kirjetest.

Kui rakendusel oli vajalik funktsionaalsus olemas, siis loodi sellele kujundus.

5.3.1 Veebirakenduse keeled

Vastavalt kasutajalugudele peab olema kasutajal võimalik kasutada rakendust ka eestikeelsena. Selleks on .NET *Core*'is võimalik kasutada .resx faile (Joonis 12). Sinna saab võti-väärtus paaridena salvestada tõlkeid.

Name	Default Culture	et - Estonian
Cabin	Cabin	Kajut
Cabin+food	Cabin+food	Kajut+söök
Calculate	Calculate	Arvuta
Calculator	Calculator	Kalkulaator
Data	Data	Andmed
Deck	Deck	Tekikoht
Departure	Departure	Väljumine
Description	Description	Kirjeldus
Dest cannot be same start	Destination cannot be same as	Lõpp-punkt ei saa olla sama, mi
Drivers	Drivers	Juhid
EConnection	E-con	E-ühendus
Food	Food	Söök
FromWhere	From where	Kust
Height	Height	Kõrgus
Length	Length	Pikkus
No deps	No suitable departures with giv	Ei leia sobivaid väljumisi antud p
PlaceFood	Cabin, food, deck	Kajut, söök, tekikoht
Price	Price	Hind
Results	Results	Tulemused

Joonis 12. .resx faili näidis.

Selle abil saab rakendus toetada paljusid erinevaid keeli ja kui peaks juhtuma, et ühes keeles puudub mõnel võtmel väärtus, siis kuvatakse vaikimisi väärtus, mis autori rakenduses on inglisekeelne. UI's on võtmeväärtus kirjutatud soovitud kohale ja siis vastavalt aktiivsele kultuurile tagastatakse sellele õiges keeles vastav väärtus.

5.4 Testimine

Käesoleva diplomitöö kontekstis mõeldakse testimise all inimese poolt rakenduse testimist, just eriti arvutusloogika testimist. Arvutusloogikast saaks automaattestida arvutuskäiku ennast, kuid palju olulisem on selle eelnevad andmebaasipäringud, kus andmed välja filtreeritakse ja andmebaasi testimise ülesseadmine on keeruline ja ajakulukas.

Arvutusloogika toimimist saab pooleldi testida hetkel kasutuses oleva tabeltöötlusega ja arvetega. Tabeltöötluse abil testimine ei ole ammendav, kuid annab hea ülevaate, kas põhilised hinnad on õiged. Arve abil testimisel sisestatakse arvel olevad veose parameetrid kalkulaatorisse ja siis võrreldakse tulemusi, esiteks, kas lõpptulemus on sama, mis arvel ja teiseks, kas detailvaates on näha samad teenused ja samad hinnad.

6 Hinnang loodud veebirakendusele

Loodud veebirakendust saab objektiivselt hinnata kasutajanõuete põhjal. Kõik funktsionaalsed ja mittefunktsionaalsed nõuded said täidetud. Kõige keerulisem arenduse juures oli arvutusloogika loomine, sest oli oluline, et see oleks ammendav ja andmebaasipäringud oleksid võimalikult optimaalselt tehtud. Arendusprotsessi käigus arenes diplomitöö autoril SQL-lausete koostamise ja UI kujundamise oskus.

Tehnoloogia valimisel oli jälgitud põhimõtet, et kasutada arenduseks tehnoloogiaid, mis on laialt levinud, hästi dokumenteeritud ja poleks tasuline. See annab kindlustunde, et rakenduses kasutatavad tehnoloogiaid uuendatakse tihti ja tänu sellele saab ära hoida turvaaukude tekke, mille võivad põhjustada vananenud tehnoloogiad. Nii .NET *Core* kui ASP.NET on laialt kasutatavad tehnoloogiad veebirakenduste arendamiseks ja sobivad hästi koos kasutamiseks.

Edasiarendusena saaks veebirakendust ühildada ettevõttes olemasoleva laevapileti broneerimisrakendusega või tuua selle loogika üle enda rakendusse, mis muudaks kogu protsessi palju mugavamaks ja kiiremaks. Selleks tuleb luua broneerimisrakendusse ühendused, mille abil saaks broneerimisfunktsionaalsust pakkuda ka enda rakenduses, aga seda ainult sellele kindlale ettevõttele.

Lisaks on üks võimalus luua ka laevafirmapoolne klientrakendus, mis võimaldaks laevafirmadel endal ise sisestada ja muuta väljasõidugraafikuid ning hindasid. Kuid selle lahenduse tulusus on kaheldav, sest laevafirmadel on palju lihtsam saata hinnakiri pdf-kujul.

7 Kokkuvõte

Antud diplomitöö eesmärk oli luua laeva piletihinna arvutamise veebirakendus ülegabariidilisi vedusid korraldavatele ettevõttetele. Kasutajanõuded rakenduse jaoks koguti intervjuude käigus. Intervjuude käigus selgus, et rakenduses peab olema võimalik ka vaadata hinnakirjasid ja väljumisgraafikuid.

Töö analüüsi osa annab ülevaade kasutajanõuetele põhinevalt planeeritavatest funktsionaalsustest, võimalikkest tehnoloogiatest ning nende eelistest ja puudustest. Veebirakenduse arenduskäiku kirjeldav peatükk annab ülevaate lõpptulemuseni viivatest sammudest, mis hõlmab andmebaasi arendust, teenuspoolse rakenduse arendust, kliendipoolse rakenduse arendust ja testimist.

Loodud veebirakendus rahuldab kõiki analüüsis väljatoodud kasutajanõudeid. Rakendusel on võimalusi edasiarenguks, luues sinna laevapileti broneerimisvõimaluse või laevafirmadele klientrakenduse hinnakirjade ja väljumisgraafikute haldamiseks.

Kasutatud kirjandus

[1] T.-. j. Sideminister, „Suuremõõtmelise ja/või raskekaalulise autoveo eeskiri,“ 2012. [Võrgumaterjal]. Loetud aadressil: https://www.riigiteataja.ee/akt/124072012015. [Kasutatud 13.02.2022].

[2] K. OÜ, „Projects,“ 2015. [Võrgumaterjal]. Loetud aadressil: https://kaarlaid.ee/et/reference/. [Kasutatud 13.02.2022].

[3] K. OÜ, „Teenused,“ 2015. [Võrgumaterjal]. Loetud aadressil: https://kaarlaid.ee/et/teenused/. [Kasutatud 13.02.2022].

[4] K. OÜ, „Erialad,“ 2015. [Võrgumaterjal]. Loetud aadressil: https://kaarlaid.ee/et/erialad/. [Kasutatud 13.02.2022].

[5] DSV, „Ajalugu – teke, ühinemised & omandamised | DSV,“ [Võrgumaterjal]. Loetud aadressil: https://www.dsv.com/et-ee/dsv/ajalugu. [Kasutatud 05.03.2022].

[6] DSV, „Transpordi ja kaubaveo tüübid DSV,“ [Võrgumaterjal]. Loetud aadressil: https://www.dsv.com/et-ee/meie-lahendused/transpordiliigid. [Kasutatud 05.03.2022].

[7] DSV, „Projektiveod & logistika - kohandatud teenused | DSV,“ [Võrgumaterjal]. Loetud aadressil: https://www.dsv.com/et-ee/meie-lahendused/transpordiliigid/projektivedu. [Kasutatud 05.03.2022].

[8] J. E. OÜ, „JTH Eesti OÜ – Raske- ja Eriveod Balti, Lääne- ja Põhja-Euroopa Riikides,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://jth.ee/#veotehnika. [Kasutatud 05.03.2022].

[9] D. S. CORP, „Desktop App vs Web App: Comparative Analysis,“ 2020. [Võrgumaterjal]. Loetud aadressil: https://digitalskynet.com/blog/Desktop-App-vs-Web-App-Comparative-Analysis. [Kasutatud 06.03.2022].

[10] D. Bulatovych, „Choosing a Tech Stack for the Full-Cycle Web Application Development in 2021,“ 2021. [Võrgumaterjal]. Loetud aadressil: https://yalantis.com/blog/tech-stack-for-web-app-development/. [Kasutatud 11.03.2022].

[11] Symfony, „Symfony, High Performance PHP Framework for Web Development,“ [Võrgumaterjal]. Loetud aadressil: https://symfony.com/ten-criteria. [Kasutatud 11.03.2022].

[12] L. Harkushko, „Detailed Analysis of the Top Modern Database Solutions,“ [Võrgumaterjal]. Loetud aadressil: https://yalantis.com/blog/how-to-choose-a-database/. [Kasutatud 12.03.2022].

[13] P. Raspel, „Relatsiooniline andmebaas,“ [Võrgumaterjal]. Loetud aadressil: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1.%20Loengumaterjalid/. [Kasutatud 11.03.2022].

[14] „DB-Engines Ranking,“ aprill 2022. [Võrgumaterjal]. Loetud aadressil: https://db-engines.com/en/ranking. [Kasutatud 11.03.2022].

[15] Ian, „What does ACID mean in Database Systems?,“ 20.06.2016. [Võrgumaterjal]. Loetud aadressil: https://database.guide/what-is-acid-in-databases/. [Kasutatud 11.03.2022].

[16] ScaleGrid, „2019 Database Trends: SQL vs. NoSQL - Top Databases,“ 04.03.2019. [Võrgumaterjal]. Loetud aadressil: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/. [Kasutatud 11.03.2022].

[17] C. S. Mullins, „What is a DBMS? Database Management System Definition,“ veebruar 2021. [Võrgumaterjal]. Loetud aadressil: https://searchsqlserver.techtarget.com/definition/database-management-system. [Kasutatud 11.03.2022].

[18] S. Overflow, „Stack Overflow Developer Survey 2021,“ 2021. [Võrgumaterjal]. Loetud aadressil: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-database-prof. [Kasutatud 11.03.2022].

[19] Javatpoint, „Learn MySQL Tutorial - javatpoint,“ [Võrgumaterjal]. Loetud aadressil: https://www.javatpoint.com/mysql-tutorial. [Kasutatud 11.03.2022].

[20] AWS, „What is PostgreSQL? – Amazon Web Services,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://aws.amazon.com/rds/postgresql/what-is-postgresql/. [Kasutatud 11.03.2022].

[21] B. Chen, „PostgreSQL vs. MySQL,“ 02.09.2021. [Võrgumaterjal]. Loetud aadressil: https://www.fivetran.com/blog/postgresql-vs-mysql. [Kasutatud 11.03.2022].

[22] Oracle, „Process Structure,“ 1997. [Võrgumaterjal]. Loetud aadressil: https://docs.oracle.com/cd/A58617_01/server.804/a58227/ch_procs.htm. [Kasutatud 13.03.2022].

[23] E. S., „SQLite vs MySQL – What’s the Difference,“ 17.03.2022. [Võrgumaterjal]. Loetud aadressil: https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/. [Kasutatud 19.03.2022].

[24] A. Duggal, „SQLite vs MySQL: 5 Critical Differences,“ 01.11.2021. [Võrgumaterjal]. Loetud aadressil: https://hevodata.com/learn/sqlite-vs-mysql/. [Kasutatud 17.03.2022].

[25] Microsoft, „SQL Server 2019—Pricing | Microsoft,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.microsoft.com/en-us/sql-server/sql-server-2019-pricing#OneGDCWeb-ContentPlacementWithRichBlock-pp5ed24. [Kasutatud 17.03.2022].

[26] E. Team, „PostgreSQL vs. SQL Server (MSSQL) - Extremely Detailed Comparison,“ 30 07 2020. [Võrgumaterjal]. Loetud aadressil: https://www.enterprisedb.com/blog/microsoft-sql-server-mssql-vs-postgresql-comparison-details-what-differences. [Kasutatud 17.03.2022].

[27] Javatpoint, „JavaScript vs TypeScript - javatpoint,“ 2021. [Võrgumaterjal]. Loetud aadressil: https://www.javatpoint.com/javascript-vs-typescript. [Kasutatud 13.03.2022].

[28] T. Fowler, „Is JavaScript Front End or Back End?,“ 05.08.2020. [Võrgumaterjal]. Loetud aadressil: https://careerkarma.com/blog/javascript-front-end-or-back-end/. [Kasutatud 19.03.2022].

[29] freeCodeCamp, „The Advantages and Disadvantages of JavaScript,“ 05.12.2019. [Võrgumaterjal]. Loetud aadressil: https://www.freecodecamp.org/news/the-advantages-and-disadvantages-of-javascript/. [Kasutatud 13.03.2022].

[30] T. point, „Python Tutorial,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.tutorialspoint.com/python/index.htm. [Kasutatud 19.03.2022].

[31] M. Rana, „Top 8 Backend Languages That Will Make Wave in 2022,“ 27.12.2021. [Võrgumaterjal]. Loetud aadressil: https://www.europeanbusinessreview.com/top-8-backend-languages-that-will-make-wave-in-2022/. [Kasutatud 19.03.2022].

[32] D. Tate, „What is TypeScript? Pros and Cons,“ 08.04.2015. [Võrgumaterjal]. Loetud aadressil: https://medium.com/@BuildMySite1/what-is-typescript-pros-and-cons-8dc5cdc3e78d. [Kasutatud 19.03.2022].

[33] Javatpoint, „Advantages and disadvantages of Java - Javatpoint,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.javatpoint.com/advantages-and-disadvantages-of-java. [Kasutatud 19.03.2022].

[34] Altexsoft, „The Good and the Bad of C# Programming,“ 29.10.2021. [Võrgumaterjal]. Loetud aadressil: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/. [Kasutatud 19.03.2022].

[35] S. Veeraraghavan, „14 Best Programming Languages to Learn in 2022 | Simplilearn,“ 08 12 2019. [Võrgumaterjal]. Loetud aadressil: https://www.simplilearn.com/best-programming-languages-start-learning-today-article. [Kasutatud 20.03.2022].

[36] R. Kenneth, „How to Measure Programming Language Complexity,“ 22.02.2019. [Võrgumaterjal]. Loetud aadressil: https://richardeng.medium.com/how-to-measure-programming-language-complexity-afe4f7e75786. [Kasutatud 20.03.2022].

[37] R. Krajewski, „C# vs Java: Which Is Better For Building Your Product?,“ 05.07.2021. [Võrgumaterjal]. Loetud aadressil: https://www.ideamotive.co/blog/c-sharp-vs-java-which-is-better-for-building-your-product. [Kasutatud 20.03.2022].

[38] M. Szorad, „Why would someone use a framework?,“ 15.06.2021. [Võrgumaterjal]. Loetud aadressil: https://levelup.gitconnected.com/why-would-someone-use-a-framework-bd4706e4464f. [Kasutatud 20.03.2022].

[39] A. Danielkievich, „Exploring Node.js Pros and Cons,“ 08.09.2021. [Võrgumaterjal]. Loetud aadressil: https://forbytes.com/blog/nodejs-pros-and-cons/. [Kasutatud 20.03.2022].

[40] S. Academy, „Flask Vs Django: Which Python Framework to Choose?,“ 08 02 2022. [Võrgumaterjal]. Available: https://www.interviewbit.com/blog/flask-vs-django/. [Kasutatud 20 03 2022].

[41] Tutorialspoint, „Spring Framework - Overview,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.tutorialspoint.com/spring/spring_overview.htm. [Kasutatud 20.03.2022].

[42] TutorialsTeacher, „.NET Core Overview,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.tutorialsteacher.com/core/dotnet-core. [Kasutatud 20.03.2022].

[43] J. Boyer, „How Secure Are Popular Web Frameworks? Here Is a Comparison,“ 17.07.2018. [Võrgumaterjal]. Loetud aadressil:

https://www.veracode.com/blog/secure-development/how-secure-are-popular-web-frameworks-here-comparison. [Kasutatud 20.03.2022].

[44] ThemeSelection, „The Best Python Web Frameworks,“ 10.03.2022. [Võrgumaterjal]. Loetud aadressil: https://dev.to/theme_selection/the-best-python-web-frameworks-d2d. [Kasutatud 20.03.2022].

[45] C. Details, „Nodejs Node.js : List of security vulnerabilities,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.cvedetails.com/vulnerability-list/vendor_id-12113/product_id-30764/year-2022/Nodejs-Node.js.html. [Kasutatud 11.05.2022].

[46] C. Details, „Palletsprojects Flask : List of security vulnerabilities,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.cvedetails.com/vulnerability-list/vendor_id-17201/product_id-57169/year-2019/Palletsprojects-Flask.html. [Kasutatud 11.05.2022].

[47] C. Details, „Djangoproject Django : List of security vulnerabilities,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.cvedetails.com/vulnerability-list/vendor_id-10199/product_id-18211/year-2022/Djangoproject-Django.html. [Kasutatud 11.05.2022].

[48] C. Details, „Vmware Spring Framework : List of security vulnerabilities,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.cvedetails.com/vulnerability-list/vendor_id-252/product_id-96553/year-2022/Vmware-Spring-Framework.html. [Kasutatud 11.05.2022].

[49] C. Details, „Microsoft .net Core : List of security vulnerabilities,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-43007/year-2022/Microsoft-.net-Core.html. [Kasutatud 11.05.2022].

[50] A. Insignares, „React Pros and Cons: What are the Advantages and Disadvantages of ReactJS?,“ 10 03 2021. [Võrgumaterjal]. Loetud aadressil: https://www.koombea.com/blog/react-pros-and-cons-what-are-the-advantages-and-disadvantages-of-reactjs/. [Kasutatud 20.03.2022].

[51] N. Sakovich, „Best Frontend Frameworks for Web Development of 2021–2022 | SaM Solutions,“ 10 12 2021. [Võrgumaterjal]. Loetud aadressil: https://www.sam-solutions.com/blog/best-frontend-framework/. [Kasutatud 20.03.2022].

[52] R. Brandsness, „The Pros and Cons of jQuery,“ 16.06.2019. [Võrgumaterjal]. Loetud aadressil: https://medium.com/@rachelbrandsness/the-pros-and-cons-of-jquery-e7cca3e9e210. [Kasutatud 20.03.2022].

[53] Altexsoft, „The Good and the Bad of Angular Development,“ 25.03.2022. [Võrgumaterjal]. Loetud aadressil: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/. [Kasutatud 26.03.2022].

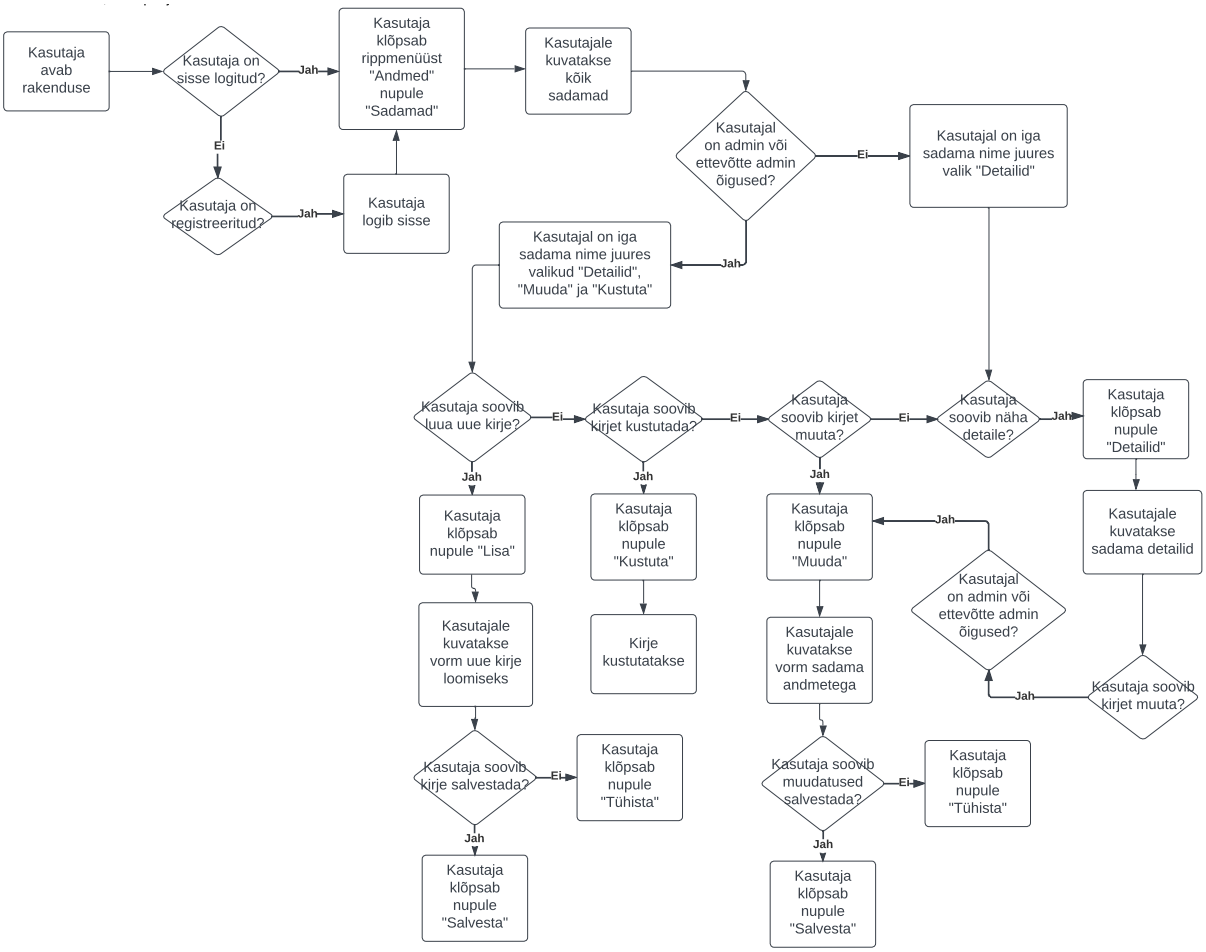
[54] J. Singh, „Why We Should And Should Not Use .NET Core,“ 11.03.2019. [Võrgumaterjal]. Loetud aadressil: https://www.c-sharpcorner.com/blogs/why-should-and-should-not-we-use-net-core. [Kasutatud 26.03.2022].

[55] S. Telgave, „What ASP.NET Core Is And Advantages Of Using It,“ 26.07.2018. [Võrgumaterjal]. Loetud aadressil: https://www.c-sharpcorner.com/article/what-is-asp-net-core-and-advantages-of-using-asp-net-core-how-to-setup-asp-net/. [Kasutatud 26.03.2022].

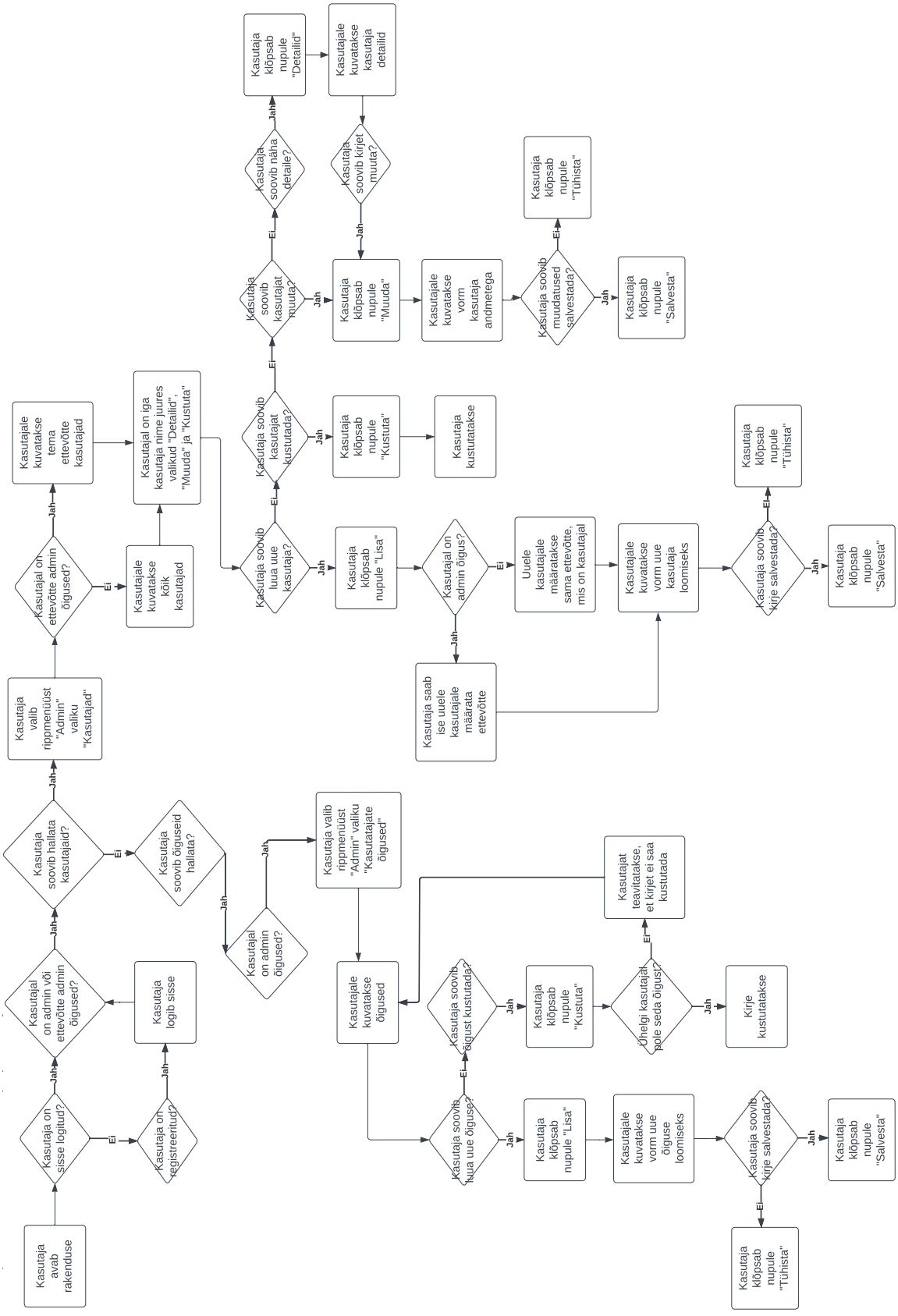
[56] Microsoft, „Overview of Entity Framework Core - EF Core,“ 25.05.2021. [Võrgumaterjal]. Loetud aadressil: https://docs.microsoft.com/en-us/ef/core/. [Kasutatud 27.03.2022].

[57] Microsoft, „Querying Data - EF Core,“ 11.03.2021. [Võrgumaterjal]. Loetud aadressil: https://docs.microsoft.com/en-us/ef/core/querying/. [Kasutatud 27.03.2022].

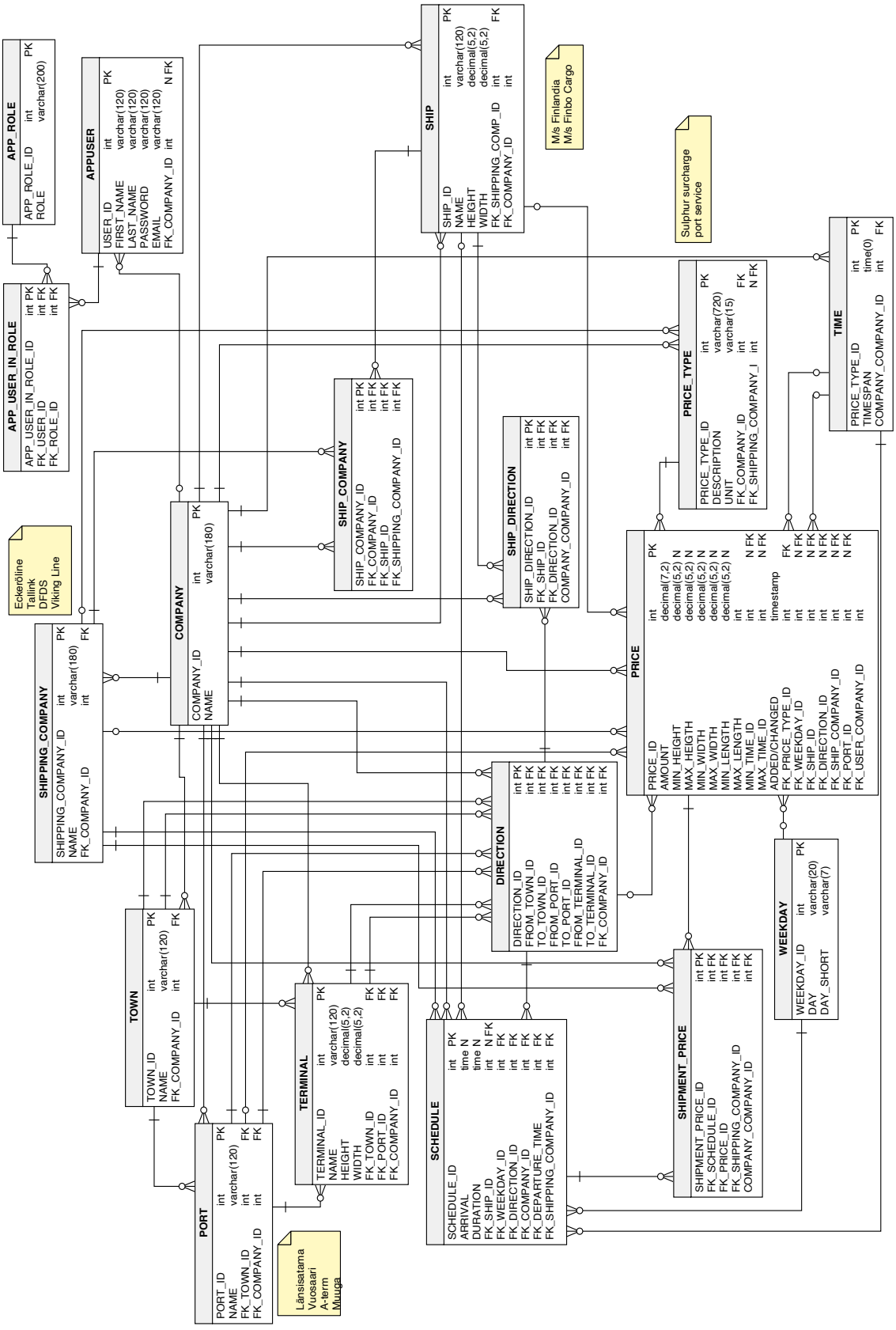
Lisa 1 –Andmete vaatamise, lisamise, muutmise ja kustutamise plokkskeem



Lisa 2 – Kasutajate ja nende õiguste haldamise plokkskeem



Lisa 3 – Olemi-suhte diagramm



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

Mina, Desiree Himuškin

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Laevapileti hinna arvutamise veebirakendus suuremõõtmelisi vedusid korraldavale ettevõttele“, 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.

15.05.2022

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.