TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Teele Kaldaru 185762IADB

Postisaadetisi vedavate kaubaautode kulujaotussüsteemi arendamine

Bakalaureusetöö

Juhendaja: Toomas Lepikult

PhD

Tallinn 2021

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: Teele Kaldaru

17.05.2021

Annotatsioon

Logistikaettevõte Post11 haldab paljusid kaubaautosid, mida kasutatakse mitmete klientide ehk veebipoodide postisaadetiste toimetamiseks sihtpunkti. Iga auto sisaldab endas erinevate klientide saadetisi. Iga teatud perioodi järel on vajalik klientidele nende kauba transportimise eest arveid koostada. Teada on iga autoga kaasnenud kogukulu, kuid vastavalt saadetiste raskusele, tüübile või mahule on vaja kogukulu klientide vahel võrdeliselt ära jagada.

Käesoleva töö raames arendati logistikalahendusi pakkuva ettevõtte veebirakendusse kaubaautode kulujaotussüsteem, mis automatiseeriks ja lihtsustaks kaubaautodega seonduvate kulude proportsionaalset jaotamist. Süsteemi eesmärk oli asendada varasemalt käsitsi tehtavat protsessi ning suurendada ettevõtte tööefektiivsust.

Kuna kaubaautode kulude jaotamisel on mitmeid reegleid ja arvutusviise, siis pidi vaadeldav süsteem olema süvitsi läbi mõeldud. Kõik erijuhud ja tingimused pidid olema kaetud ning vigade tekkimise tõenäosus minimaalne. Suurim probleem oli süsteemi põhimõttest aru saamine ning selle alusel võimalikult töökindla ning mugavalt kasutatava süsteemi arendamine.

Töö tulemusena valmis planeeritud veoautode kulujaotussüsteem. Süsteemi testiti nii arendustiimi kui ka finantsosakonna liikmete poolt ning võeti veebirakenduses kasutusse. Süsteem võimaldab automatiseerida varem manuaalselt tehtud tööd ning lihtsustab oluliselt ettevõtte töötajate toiminguid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 6 peatükki, 4 joonist, 1 tabelit.

Abstract

Development of a Cost Allocation System for Trucks Transporting Postal Items

The logistics company Post11 manages many commercial vehicles, which are used to deliver mail sent from many customers, in ohter words online stores. Each vehicle contains shipments from different customers. After a certain period, it is necessary to invoice customers for the transportation of their goods. The total cost of each car is known, but according to the weight, type or volume of the shipments, the total cost needs to be shared proportionally between customers.

In the framework of this work, a commercial vehicle cost allocation system was developed in the web application of a company providing logistics solutions, which would automate and simplify the proportional distribution of commercial vehicle-related costs. The purpose of the system was to replace the previously manual process and increase the company's work efficiency.

As there are many rules and calculation methods for allocating the costs of vehicles, the system in question had to be thoroughly thought through. All special cases and conditions had to be covered and the likelihood of errors was cut to minimal amount. The biggest problem was understanding the principles of the system and developing a system that is as reliable and convenient as possible.

As a result, the planned cost allocation system for trucks was completed. The system was tested by both the development and financial teams and actually put into operation. The system enables the automation of previously manual work and significantly simplifies the operations of the company's employees.

The thesis is in Estonian and contains 26 pages of text, 6 chapters, 4 figures, 1 table.

Lühendite ja mõistete sõnastik

.NET	*network enabled technologies.* Raamistik rakenduste loomiseks.
API	*application programming interface.* Protokoll rakenduste omavaheliseks suhtlemiseks.
ACID	*atomicity, consistency, isolation, and durability.* Andmebaasi tehingute omaduste kogum, mille eesmärk on tagada töökindlus samaaegsel kasutamisel ning tehniliste rikete korral.
AJAX	*asynchronous javascript and xml.* Kogum veebiarenduse komponente ja tehnikaid, mida kasutatakse interaktiivsete veebirakenduste loomiseks.
ASP.NET	*active server pages network enabled technologies.* Raamistik dünaamiliste veebirakenduste ja veebiteenuste loomiseks.
C#	*C sharp*. Microsofti poolt loodud programmeerimiskeel.
CSS	*cascading style sheets*. Veebilehtede kujunduskeel.
DOM	*document object model*. Liides, mis käsitleb veebidokumenti puustruktuurina, kus iga sõlm on objekt, mis tähistab dokumendi teatud osa.
EF	*Entity Framework*. Avatud lähtekoodiga objektide ja relatsioonide kaardistamise raamistik.
HTML	*hypertext markup language*. Veebilehtede märgistuskeel.
HTTP	*hypertext transfer protocol*. Veebiprotokoll andmete edastamiseks üle võrgu.
LINQ	*language-integrated query*. Tehnoloogiate kogumi nimi, mis põhineb päringuvõimaluste integreerimisel C# keelde.
MVC	*model-view-controller*. Tarkvara arhitektuuri muster.
MVVM	*model-view-viewmodel*. Tarkvara arhitektuuri muster.
NuGet	Paketihaldur, mis on loodud arendajatele korduvkasutatava koodi jagamiseks.
ORM	*object-relational mapping*. Programmeerimistehnika andmete teisendamiseks ühildumatute süsteemide vahel.
REST	*representational state transfer*. Tarkvara veebiteenuste arhitektuuristiil.
SQL	*structured query language*. Programmeerimiskeel, mida kasutatakse andmebaasiga suhtlemiseks.
VB.NET	*Visual Basic.NET*. Microsofti poolt loodud programmeerimiskeel.

XAML	*extensible application markup language*. Microsofti väljatöötatud deklaratiivne XML-põhine keel, mida kasutatakse struktureeritud väärtuste ja objektide initsialiseerimiseks.
XML	*extensible markup language*. Märgistuskeel, mis määratleb reeglite kogumi dokumentide kodeerimiseks nii inimesele kui ka masinale arusaadavas vormingus.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Veebipoodide populaarsus maailmas kasvab kiiresti ning üha rohkem tellitakse erinevaid tooteid välismaalt. Selle tõttu tekib vajadus töökindlamate ning kiiremate infosüsteemide järele, mis aitaksid suurt kogust logistikaettevõtete terminale läbivaid postisaadetisi hallata. Postisaadetiste transportimise ning käsitlemisega kaasneb suur hulk andmeid ka teiste selleks vajalike objektide, näiteks postipakke vedavate autode kohta.

Käesoleva töö raames luuakse logistikaettevõtte veebirakendusse süsteem, mis aitaks säilitada ning struktureeritult hallata postisaadetisi transportivate kaubaautodega kaasnevaid kulusid. Sellise ülesandega süsteem hetkel ettevõttes puudub. Autodega kaasnevate maksumuste arvutatamist ja seotud osapoolte vahel proportsionaalselt jaotamist tehakse ettevõttes käsitsi. Samuti ei salvestata kogutud andmeid andmebaasi ning seetõttu püsib oht, et tähtsad andmed kaovad või need aetakse omavahel segamini.

Veoautode kulujaotussüsteemi eesmärk on manuaalse töö vähendamine ning selle tulemusena ettevõtte töötempo kiirendamine ja inimlike vigade tekkimise vältimine. Eesmärgi saavutamiseks arendatakse autode kulusid haldav süsteem, mis integreeritakse ettevõtte olemasolevasse veebirakendusse. Parima lahenduse teostamiseks kaasatakse arutellu ettevõtte arendustiim ja finantsosakonna liikmed, et arendatav süsteem oleks võimalikult kasutajasõbralik ning tööefektiivne.

Esmalt antakse antud töös ülevaade ettevõtte olemasolevast infosüsteemist, partneritest ja klientidest. Samuti kirjeldatakse arendatava kulujaotussüsteemi tausta ja täpsustatakse ülesande püstitust ning eesmärke. Järgmises peatükis tutvustatakse süsteemi nõudeid ja reegleid. Seejärel antakse ülevaade süsteemi realiseerimiseks kasutatud tehnoloogiatest ja tööriistadest. Töö lõpus võrreldakse uut süsteemi endise metoodikaga, kirjeldatakse süsteemi testimist ning edasiarenduse võimalusi.

2 Ülevaade ettevõtte infosüsteemist

Arendatav süsteem kuulub ettevõttele Post11, mis tegeleb mitmesuguste piiriülese e-kaubanduse logistikaga seotud lahenduste pakkumisega [1]. Ettevõte kasvas välja kaubamärgi Omniva all turul esindatud Eesti Posti Grupi rahvusvaheliste operatsioonide osakonnast. Aastast 2016 tegutseb see Post11 nime all ja kuulub Omniva kontserni sidusettevõttena [2]. Post11 peakorter asub Eesti pealinnas Tallinnas. Peale Eesti tegutseb Post11 ka Inglismaal, Saksamaal ning Ameerikas. Post11 on loonud partnerlussuhteid erinevate turuliidritega, et võimaldada teenuse pakkumist laialdaselt üle maailma suuremates tööstuspiirkondades, seal hulgas näiteks Ühendkuningriikides ja Hiinas [3].

2016. aastal kogutud andmete põhjal käsitletakse kuus üle 3 miljoni postisaadetise, mille kogukaal ulatub 300 tonnini. Pakke saadetakse üle 80 erinevasse riiki. Peamised sihtkohad on Venemaa, Ukraina, Valgevene, Rootsi, Poola, Soome, Norra, Tšehhi ja Slovakkia. Peamine saadetiste päritoluriik on Hiina [4].

2.1 Infosüsteem

Post11 infosüsteem jaguneb kaheks osaks: veebirakenduseks ning terminali rakenduseks. Veebirakendus hõlmab endas põhiosa kogu informatsioonist ja seal on võimalik teostada suuremat osa toimingutest. Terminali rakendus on loodud kasutamiseks postiterminalides pakkide triipkoodide skaneerimiseks mõeldud pultides. Selle kasutusvõimaluste arv on väiksem ja kasutajaliidese disain lihtsam. Terminali rakendus on mõeldud spetsiaalselt kaubakottide ning pakkide haldamiseks.

2.1.1 Veebirakendus

Post11 veebirakendus on mõeldud kasutamiseks erinevatele Post11 töötajatele ja klientidele. Veebirakenduse kasutajatele on antud vastavad rollid ning nendele rollidele antud õigused määravad selle, millistele rakenduse komponentidele kasutajale juurdepääs on võimaldatud.

Veebirakenduses saab kontrollida suuremat osa ettevõtte infosüsteemis ja terminalis toimuvat. Peamiselt kasutatakse seda reaalselt käsitletud postisaadetiste informatsiooni dokumenteerimiseks ja nende tarnimisega seotud transpordivahendite haldamiseks. Lisaks sellele saab veebirakenduses luua ja modifitseerida mitmeid äriloogikas olulisi reegleid, mis on vajalikud selleks, et saadetiste seisukorda ning nendega seotud kulusid lihtsamalt hallata ja et suuremat osa nendega teostatavaid toiminguid oleks võimalik automaatselt täide viia. Tähtsamad komponendid, mida veebirakendus sisaldab on järgmised:

- Tarneüksuste (*shipment*) loomine, muutmine ja kustutamine. Üksuste loendivaates on võimalik neid erinevate parameetrite alusel filtreerida.
- Tarneüksustele postisaadetiste peale laadimine ehk postisaadetiste sidumine vastava tarneüksusega. Peale laetud saadetisi saab vajadusel eemaldada. Samuti on tarneüksuse detailvaatest võimalik alla laadida selle üksusega seotud saadetiste infosedeleid.
- Saadetiste jälgimine paki koodi alusel. Post11 veebirakendusele ligipääsuõigustega isikud saavad näha ka süsteemisiseseid toiminguid, mis pakkidega on tehtud.
- Iga paki tarnega kaasnevad erinevad kulud ning Post11 esitab mitmetele osapooltele arveid. Veebirakenduses on olemas koht, kus pakkide omaduste põhjal määratakse automaatselt pakiga seonduvad maksumused.
- Klientide haldamise süsteem, kus saab lisada, muuta ja kustutada kliente ning nende andmeid.
- Saadetistega seotud kaebuste lisamine, detailide haldamine ja muutmine ning kustutamine. Kaebused on tavaliselt seotud paki hiljaks jäämise ja kahjustada saamisega, kuid põhjuseid on teisigi.
- Detailne ülevaade terminalides toimuva kohta. Näha on kõikides terminalides hetkel käsitletavad tarneüksused ja informatsioon nendes olevate pakkide staatuste kohta.
- Kaubaautode terminalikülastuste loendivaade, nende kohta informatsiooni lisamine, autodele saadetiste laadimine ehk pakkide sidumine vastava autoga. Kaubaautode saabumis- või väljumisaegade registreerimine.

- Kaubaautode süsteemi sisestamine, nende teabe muutmine ja kustutamine. Neile kindla ajagraafiku määramine ehk millal teatud auto terminali jõuab või sealt väljub.
- Raportite koostamine, haldamine, kuvamine ja kustutamine.
- Sõnumite ja teavituste loendivaade ja nendega seotud detailide kuvamine.
- Kasutajate lisamine, detailide haldamine ja kustutamine. Kasutajatele rollide määramine. Samuti saab samas blokis muuta rollidele antud õigusi.

2.1.2 Terminali rakendus

Post11 terminali rakendus on eelkõige mõeldud kasutamiseks terminalides postipakkide triipkoodide skaneerimiseks mõeldud pultides. Seetõttu on rakenduse kujundus minimalistlik ning omane mobiilirakendustele, sest puldi ekraani suurus on sarnane mobiiliekraaniga. Terminali rakendus on loodud postiterminalide läbivate pakkide tegevusvoo dokumenteerimiseks. Saadetistega tehtavad sammud salvestatakse terminali rakenduse abil infosüsteemi. Nii on iga paki kohta alati teada, millised etapid sellega juba läbitud on ning mis tuleb veel teostada. Puldiga saadetise triipkoodi skaneerides kuvatakse puldi ekraanile antud saadetise järgmine vajalik etapp. Paki tegevusvoo sammud ja nende järjekord on igal pakil individuaalsed ning need sõltuvad mitmetest paki parameetritest, näiteks selle tüübist, saatjast ja sihtkohast.

2.2 Kliendid

Post11 kliendid on erinevad veebipõhist jaemüügiteenust pakkuvad ettevõtted, millest valdav osa asub Hiinas. Klientide ajend Post11 teenuste kasutamiseks on saadetiste kohaletoimetamise vastutuse edasi andmine selle eesmärgiga loodud firmale, et vähendada enda halduses olevat infohulka.

Klientide poolt vaadatuna on süsteemi esimene kiht nende endi veebipood, kust lõppkliendid kaupu tellivad. Tellitud kaupade andmed peavad veebipoest saama Post11 infosüsteemi, et nendega oleks võimalik edasi tegeleda.

Kliendid saavad Post11-ga koostööd teha läbi API integratsiooni või kasutades otse Post11 veebirakendust. API integratsiooni puhul kasutab klient enda klientrakendust, kus on võimalik sisestada ning hallata tellimusi. Tagarakendusena on kasutusel siiski Post11

infosüsteem ning sinna saavad klientrakendused ligi API vastu päringuid saates. Saadetiste info salvestub Post11 andmebaasi.

Juhul kui klient ei soovi enda klientrakendust arendada, on võimalik kasutada Post11 veebirakendust. Võrreldes teiste rakenduse kasutajatega on klientidel üsna piiratud võimalused. Tähtsamad klientidele saadaval olevad toimingud on enda või oma alamklientide nimele tarneüksuste loomine ja nende haldamine, saadetiste jälgimine paki koodi alusel ning enda saadetistega seotud kaebuste lisamine, muutmine ja kustutamine.

2.3 Partnerid

Post11 teeb koostööd erinevate kulleri- ja transporditeenust pakkuvate firmade ja riiklike postiteenuse ettevõtetega. Transporditeenust pakkuvatelt partneritelt ostetakse sisse saadetisi vedavate kaubaautode kasutamise võimalust. Post11 ei oma ise ühtegi selle eesmärgiga kasutatavat autot. Veoautosid kasutatakse peamiselt pakkide postiterminalide ja lennujaamade vaheliseks sõidutamiseks.

Kullerifirmadest ja riiklikest postiettevõtetest partnerid vastutavad alates paki sihtriiki jõudmisest, et see jõuaks edukalt lõppkliendini. Kuna nemad vastutavad saadetise teekonna viimase miili (*last mile*) eest, nimetatakse neid ettevõtteid *last mile* partneriteks. *Last mile* partneri valimine teatud saadetisele on kindlaks määratud erinevate reeglitega, mille parameetriteks on saadetise hind, kaal, sihtriik ja veebikaubanduse teenusepakkuja. Partner määratakse süsteemi poolt igale saadetisele eraldi ning selle alusel toimub pakkide sorteerimine ning kaubaautodele peale laadimine.

2.4 Taustainfo

Logistikaettevõtte Post11 terminale läbivad igapäevaselt väga paljud erinevate Hiina veebipoodide ehk klientide saadetisi transportivad veoautod. Post11 ei oma füüsiliselt ise ühtegi veosõiduvahendit, vaid neid renditakse partnerettevõtetelt. Iga teatud perioodi tagant, tavaliselt iga kuu lõpus esitavad partnerid Post11-le autodega kaasnenud kulude kohta arveid. Iga auto kulud on erinevad ning need sõltuvad erinevatest parameetritest, näiteks auto suurusest ja vanusest, kütuse tüübist, läbitud vahemaa pikkusest või autoga tööülesannete täitmise ajal tekkinud kahjustuste remontimise vajadusest.

Post11 esitab iga kuu aja tagant partnerite poolt esitatud arvete alusel omakorda klientidele ehk Hiina veebipoodidele arveid, et katta nende kaupade vedamisega kaasnenud kulud. Kulud on vaja klientide vahel proportsionaalselt ära jagada, sest igasse autosse laetakse peale mitmete erinevate klientide saadetised ning nendega kaasnevad kulud võivad olla erinevad. Näiteks kui autole on laetud peale kahe kliendi tooted ja ühe kliendi toodete osakaal on 80% ning teisel 20%, siis ei ole õiglane kogu maksumus nende klientide vahel võrdselt pooleks jagada. Iga klient peaks maksma täpselt nii palju kui tema kaupade transportimisele kulus, mitte vähem või rohkem. Üle maksvad kliendid võivad logistikaettevõtet hakata nägema ebasoodsa koostööpartnerina ning koostöö lõpetada.

2.5 Ülesande püstitus

Hetkel puudub antud ettevõtte veebirakenduses süsteem, mis veoautodega seonduvaid kulusid haldaks ja seda tööd tehakse käsitsi. Manuaalse töö peamisteks miinusteks on suur ebatäpsuste ja vigade tekkimise tõenäosus ning aeglane töökiirus. Kuna autode maksumusega seotud andmed ei ole korrapäraselt struktureeritud ning säilitatud, on võimalik olukord, et partnerid esitavad ettevõttele suuremaid arveid kui need tegelikkuses olema peaksid. Samuti puudub võimalus autode kulusid ja nende sõltuvust erinevatest muutujatest analüüsida ning tuleviku jaoks statistilisi järeldusi teha.

Antud töö raames luuakse logistikaettevõttele postisaadetisi vedavate kaubaautode kulude jaotamise süsteem. Süsteemi abil saab üles laadida üheses formaadis veoautodega seonduvaid kuluraporteid, mille alusel saab partnerite poolt esitatud arvete korrektsust kontrollida ning jagada autodega kaasnenud maksumused õiglaselt ja proportsionaalselt klientide vahel ära.

Automaatne kulujaotussüsteem edendab oluliselt ettevõtte töökiirust ning loob võimaluse kaubaautodega kaasnenud kulusid struktureeritult hallata ja säilitada. Edaspidi saab salvestatud andmeid kasutada tulevate kulude suurusjärgu ennustamisel ja erinevate partnerite pakkumiste võrdlemisel, et teha ettevõttele võimalikult soodsaid tehinguid.

3 Analüüs

3.1 Kaubaautode külastuste vaate kirjeldus

Arendatav kulujaotussüsteem lisatakse Post11 veebirakendusse kaubaautode terminalikülastuste vaatesse. Igale süsteemi loodud autole on määratud kindel sõidugraafik. Külastuste loendivaadet kuvades genereeritakse jooksvalt käesoleval päeval eeldatavasti terminali jõudvate või sealt väljuvate autode külastusajad. Käesoleva päeva potentsiaalseid külastusi andmebaasi lõplikult ei salvestata. Kaubaauto terminalikülastus salvestatakse baasi alles siis, kui see on registreeritud ehk sellele on määratud reaalsed andmed, näiteks auto tegelik saabumise või väljumise aeg.

Terminalikülastuste loendivaates on vaikimisi kuvatud kõik käesolevale päevale planeeritud või juba registreeritud kaubaautode külastused. Vaates on võimalus avada külgribana avanev filter, milles saab külastusi filtreerida järgmiste parameetrite alusel: auto nimi, tarneüksuse number, saabumise või väljumisega seotud dokumendi number, teenusepakkuja ning saabumis- ja väljumisaegade vahemik.

3.2 Funktsionaalsed nõuded

Arendatava süsteemi funktsionaalsed nõuded püstitati projekti arendustiimiga arutades ning analüütiku poolt koostatud ülesande täpsustuste põhjal. Nõuded kooskõlastati ettevõtte finantsosakonna töötajatega, kes varasemalt tegid sama tööd käsitsi ning kuulati ära nende mõtted ja soovitused, mida arendatav süsteem võiks sisaldada.

3.2.1 Nõuded klientrakendusele

Veebirakenduse kasutajaliidesele on määratud järgmised nõuded:

- Maksekirjete malli alla- ja üleslaadimise nuppude kuvamine kaubaautode külastuste vaates. Nupud peavad olema kuvatud vaid teatud õigustega kasutajatele.
- Pärast maksekirjete tabeli üleslaadimist peab kasutajale olema kuvatud teavitus protsessi õnnestumise või ebaõnnestumise kohta. Teavituses peab olema selgelt

näidatud edukalt salvestatud maksekirjete arv. Juhul kui mõne rea valideerimine nurjus, peab teavituses olema kuvatud probleemsete ridade arv ning nupp veateateid sisaldava faili allalaadimiseks.

- Maksekirjete tabeli kuvamine terminalikülastuse detailvaates. Kui külastusel puuduvad maksekirjed, siis tabelit ei kuvata.
- Kaubaauto detailvaates kuvatud külastuste tabelisse peab olema lisatud veerg külastuse koguhinna kohta, mis kalkuleeritakse kõikide auto kuluobjekti kuuluvate maksekirjete hindade summeerimisel.

3.2.2 Nõuded serverirakendusele

Veebirakenduse serverirakendusele on määratud järgmised nõuded ja tingimused:

- Süsteem peab genereerima korrektsete andmetega täidetud Exceli malli valitud külastuste põhjal ning ignoreerima külastusi, millel puuduvad vajalikud andmed või millele ei ole võimalik enam ühtegi maksekirjet luua.
- Kasutaja peab saama süsteemi üles laadida vaid .xlsx ja .xls laiendustega faile. Muude failitüüpide puhul peab süsteem tagastama korrektse veateate.
- Süsteem peab valideerima maksekirjete tabeli väljades sisalduvad andmed. Lubatud ei ole andmed, mida ei saa teisendada ettenähtud andmetüübiks ning kõik kohustuslikud lahtrid peavad olema täidetud.
- Andmete valideerimise ebaõnnestumise puhul peab süsteem koostama uue Exceli faili, kuhu on lisatud kõik probleemsed read kasutaja poolt sisestatud andmetega ning veateated iga rea kohta. Kasutajal peab olema võimalus fail alla laadida.
- Süsteem peab koostama maksekirjed kõikide korrektsete andmeridade põhjal üleslaetud tabelis isegi siis, kui tabelis sisaldub mõni rida, mille põhjal ei saa makserida luua. Seejuures ei tohi maksekirjeid luua siis, kui mõni ebakorrektne maksekirje kuulub nendega samasse kuluobjekti.

3.3 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded täpsustavad süsteemi käitumist ning ei kirjelda konkreetseid omadusi või funktsionaalsust. Sellistele nõuetele tuleb tähelepanu pöörata, et oleks

võimalik hinnata ka süsteemi toimimise kvaliteeti, mitte lihtsalt tegevusi, mida süsteem võimaldab [5]. Süsteemile määratud mittefunktsionaalsed nõuded on järgmised:

- Arendatav süsteem peab olema kasutajale kättesaadav veebirakenduse vormis ning töötama kõikides laiemalt levinumates internetibrauserites.
- Kulujaotussüsteem peab olema sisse ehitatud olemasolevasse veebirakendusse ja sobima kokku ülejäänud süsteemi osadega. Edasine arendus ning võimalikud muudatused tulevikus peavad olema võimalikult lihtsasti tehtavad.
- Süsteem peab olema töökindel ning vigade tekkimise tõenäosus minimaalne. Kuna süsteemi kasutamine on seotud ettevõtte finantsseisuga, ei saa lubada suurel hulgal vigade tekkimist.
- Süsteem peab olema kasutajasõbralik ning kaasaegse ülesehitusega. Veateated peavad olema konkreetsed ning lihtsasti arusaadavad. Nuppude disaini valimisel peab silmas pidama, et see oleks intuitiivne ning kasutajal ei tekiks nuppude eesmärgist aru saamisega probleeme.
- Andmete lugemine ja analüüsimine peab toimuma võimalikult kiiresti, sest andmemaht võib olla mahukas. Selleks tuleb süsteemist elimineerida potentsiaalselt ajakulukad protsessid, näiteks andmebaasist andmete pärimine tsüklis.

3.4 Autovisiitide maksustamise reeglid

Kaubaautodega seotud kulude kogusumma arvutamiseks tuleb kõik selle autoga seotud maksekirjed üheaegselt süsteemi üles laadida. Ühe auto kohta käivad maksekirjed võivad olla seotud mitmete terminalikülastuste või nende vahele jäävate peatustega ehk stoppidega (*stops*), sest ühte ja sama veoautot võidakse kasutada erinevate marsruutide puhul.

Eristatakse kahte erinevat tüüpi autosid: terminali saabuvad ja terminalist väljuvad autod. Saabuvate autode puhul registreeritakse ainult nende terminali jõudmise aeg. Nende puhul pole oluline, milline oli nende eelnev marsruut ehk milliseid kohti need enne terminali jõudmist läbisid. Terminalist väljuvate autode puhul on aga kindlalt teada nende sihtkoht ja ka sihtkohale eelnevad vahepeatused ehk stopid. Seega andmebaasis eksisteerivad stopid ainult väljuvate autode kohta.

Ühe auto kõikidest maksekirjetest saadakse auto kuluobjekt, mis salvestatakse andmebaasi. Maksekirjed grupeeritakse kuluobjektideks järgmiste parameetrite alusel: auto tegelik numbrimärk või mõni muu identifikaator, auto tegelik saabumis- või väljumisaeg, maksetüüp ja auto sõidusuund ehk kas tegemist on saabuva või väljuva autoga. Ühte kuluobjekti kuuluvatel maksekirjetel on kõik need parameetrid ühesugused.

Kõikide veoautoga seotud kulude ehk ühe kuluobjekti kogusumma välja arvutamiseks on kolm viisi, mis on kindlaks määratud kasutaja poolt üleslaetavas Exceli tabelis. Autoga kaasnenud kulud saab välja arvutada ühe terminalikülastuse raames transporditud kaubaaluste arvu, kauba kogukaalu või saadetiste erikaalu alusel.

3.4.1 Maksumuse arvutamine kauba kaalu alusel

Kauba kaalu alusel arvutati veoautode kulusid ka enne kulujaotussüsteemi arendamist. Teada oli ühiku ehk kilogrammi hind ning auto laadungi kogukaal kilogrammides. Auto kogukaalu saamiseks korrutati ühiku hind kaaluga.

Süsteemis aga arvutatakse esmalt välja iga maksekirje kogusumma. Igale maksekirjele vastab kas üks terminalikülastus, kui tegemist on saabuva autoga, või stopi külastus, kui tegemist on väljuva autoga. Külastusel registreeritud laadungi kaal korrutatakse maksekirje ühiku hinnaga. Terve kuluobjekti kogusumma saamiseks summeeritakse kõik sellesse kuluobjekti kuuluvate maksekirjete maksumused.

Eraldi terminalikülastuste ja stoppide külastuste maksumuse arvutamise eesmärk on klientidele võimalikult täpsete arvete koostamine. Näiteks juhul kui mõne kliendi kaubad eemaldatakse teatud vahepeatuses autolt, pole antud kliendil mõtet maksta kogu alguspunktist lõpp-punktini läbitud vahemaa eest.

3.4.2 Maksumuse arvutamine kaubaaluste arvu alusel

Kaubaaluste arvu alusel saab maksumust arvutada vaid juhul, kui tegemist on terminalist väljuva autoga. Sel juhul jääb auto teekonnale üks või mitu vahepeatust ehk stoppi. Igal väljuval veoautol on alati vähemalt üks stopp, mis on auto lõplik sihtpunkt. Enamasti laetakse väljuvatele autodele kaup peale kaubaalustel, mis sisaldavad endas teatud hulgal postisaadetisi. Selliste väljuvate autode saadetiste hulka arvestatakse kaubaaluste kujul.

Väljuvate autodega seotud maksekirjed on alati seotud ka kindla stopiga. Juhul kui stoppi minemiseks ühtegi kaubaalust määratud ei ole, siis selles stopis ei laeta ühtegi kaubaalust autost maha. Kaubaalusteta stoppidele maksekirjeid ei looda.

Ühe maksekirje kogusumma arvutamiseks on teada ühiku hind ehk kaubaaluse hind ning kaubaaluste arv autos. Kogumaksumuse saamiseks korrutatakse kaubaaluste arv ühiku hinnaga. Kuluobjekti kogumaksumuse arvutamiseks liidetakse kõik sellesse objekti kuuluvate maksekirjete hinnad kokku.

3.4.3 Maksumuse arvutamine erikaalu alusel

Erikaalu alusel arvutatakse maksumust vaid terminali sisenevatele ehk saabuvatele autodele. Erikaalu põhjal maksumuse arvutamiseks on vajalik, et üleslaetud failis oleks kasutaja poolt ära määratud auto tegelik hind ehk selle autoga kauba transportimisele reaalselt kulunud koguhind. Auto tegelik hind on üldiselt suurem süsteemis registreeritud hinnast.

Erikaalu määratakse tavaliselt sellisele kaubale, mille ruumala ja kaal ei ole üksteisega proportsioonis. Selline olukord võib tekkida näiteks siis, kui kaubaautoga veetakse vatti. Vatt võib täita kogu auto, nii et ühtegi muud kaupa ei ole võimalik autosse laadida. Seejuures kaalub vatt oma mahuga võrreldes aga väga vähe.

Erikaalu alusel arvutamisel on teada auto tegelik koguhind ning autoga teostatud terminalikülastustel registreeritud laadungi kaal. Esmalt arvutatakse välja ühe kilogrammi hind tegeliku koguhinna põhjal. Kõikide antud auto transporditud saadetiste kaalud summeeritakse ning tegelik hind jagatakse saadud summaga. Seejärel arvutatakse välja iga maksekirje hind ehk süsteemis registreeritud kaal korrutatakse saadud ühiku hinnaga.

4 Realisatsioon

4.1 Andmevahetus

Andmevahetus klient- ning serverirakenduse vahel toimub REST (*Representational State Transfer*) meetodit kasutades. REST on arhitektuuristiil veebis arvutisüsteemide vahel standardite pakkumiseks, mis lihtsustab süsteemide vahelist andmevahetust [6]. REST-iga ühilduvaid süsteeme nimetatakse sageli ka RESTful süsteemideks. RESTful süsteemid kasutavad andmete lugemiseks, kirjutamiseks, muutmiseks ja kustutamiseks HTTP protokolli [7].

RESTful süsteemid on olekuta (*stateless*), mis tähendab, et server ja klient ei pea üksteise oleku detaile teadma. Nii saavad server kui ka klient saadetud sõnumitest aru eelmisi kirjeid arvestamata. Olekuta süsteemid saadavad ja salvestavad andmeid ressurssidena (*resources*), mis kirjeldavad erinevaid objekte, dokumente ja muid andmekogusid.

REST arhitektuuri põhimõte on see, et kliendid saadavad serverisse päringuid (*requests*), et lugeda või modifitseerida andmeid. Ühte REST lõpp-punkti (*endpoint*) saavad paljud erinevad kliendid samu päringuid saata. Päring koosneb tavaliselt HTTP tegusõnast (*verb*), päisest (*header*), ressursi aadressist ja valikulisest sõnumikehast (*message body*), mis sisaldab andmeid. Kõige rohkem kasutatakse nelja põhilist HTTP tegusõna [6]:

- GET – spetsiifilise ressursi kohta informatsiooni hankimine või ressursside kogumi pärimine,
- POST – uue ressursi loomine,
- PUT – ressursi uuendamine,
- DELETE – ressursi kustutamine.

4.2 Rollid ja õigused

Arendatava süsteemi klientrakenduse komponendid peavad olema nähtavad vaid teatud rolli omavatele kasutajatele. Üks kasutaja võib omada infosüsteemi raames mitut erinevat

rolli. Rollidele on määratud kindlad õigused, mis annab võimaluse teatud kasutajate eest vajalikke komponente ja informatsiooni peita. Kuna rakendus sisaldab paljusid erinevaid kasutajarolle, siis ei ole mõistlik ainuüksi käesoleva süsteemi arenduse raames luua täiesti uus roll. Selle asemel loodi uus õigus maksekirjete kuvamiseks ja koostamiseks. Loodud õigust on võimalik kasutajate haldusmenüüs omistada erinevatele olemasolevatele rollidele.

Maksekirjete haldamise õigus on määratud vaid Post11 töötajatele ning administraatoritele. Teistele veebirakenduse kasutajatele, näiteks klientidele ja partneritele, see funktsionaalsus kättesaadav ei ole.

Klientrakenduse poolel kontrollitakse õiguse olemasolu enne maksekirjetega seotud komponentide kuvamist. Kui kasutajal on antud õigus olemas, siis kuvatakse vastavad komponendid kasutajale. Vastasel juhul on need kasutaja eest peidetud.

Serverirakenduses on kaubaautodega seotud API päringuid vahendavatele funktsioonidele lisatud volitusatribuut, mis kontrollib enne funktsiooni käivitamist, kas päringu saatjal on vastav õigus olemas. See on vajalik selleks, et kõrvalistel isikutel puuduks võimalus teostada neile keelatud toiminguid. Kui kasutajal puudub õigus päringu tegemiseks, tagastatakse veateade ning funktsiooni ei käivitata.

4.3 Klientrakendus

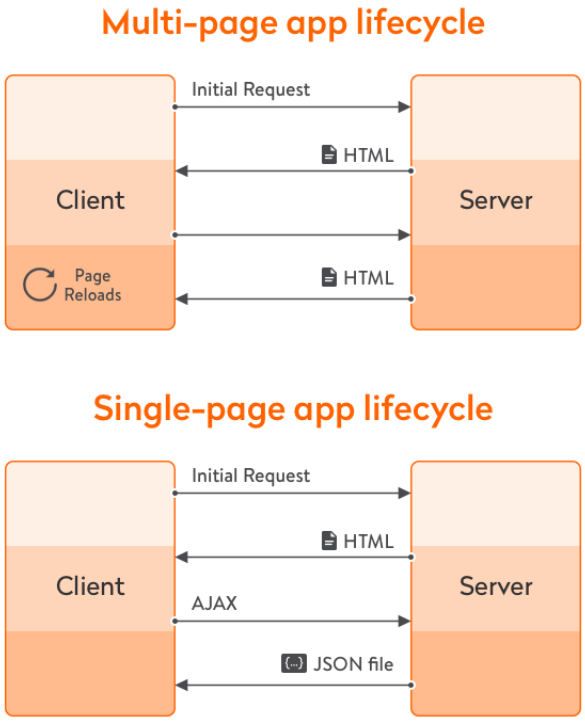
Post11 klientrakendus on ehitatud üheleherakendusena (*single-page application*) ning ühtse koodistiili eesmärgil realiseeriti sarnaselt ka töös käsitletav süsteem.

Üheleherakendus on veebirakendus, mis kasutajaga suhtlemisel uuendab veebilehte dünaamiliselt, selle asemel, et serverist tervet uut lehte alla laadida. Rakendus jookseb täielikult kasutaja internetibrauseris, luues sujuva kasutajakogemuse ilma katkestusteta eri lehtede vahel. Kogu informatsioon laetakse alla rakendust esimest korda käivitades või dünaamiliselt vastuseks kasutaja tegevustele ning lisatakse vajadusel lehele. Selline veebileht ei lae end kasutussessiooni jooksul kordagi uuesti ega suuna kontrolli edasi uuele lehele. Üheleherakendust kasutades toimub tihtipeale taustal dünaamiline suhtlus veebiserveriga [8].

Traditsiooniline lähenemine veebirakenduste loomiseks on olnud mitmeleherakendus (*multi-page application*). Sel juhul laetakse veebileht uuesti iga kord, kui selle sisu värskendatakse. Mitmeleherakenduste puhul kasutatakse AJAX meetodit, mis võimaldab väiksemaid komponente uuendada kogu veebilehte uuesti laadimata. Selle tehnoloogia arendamine on aga ajakulukas ning keerukam, kui üheleherakenduste puhul [9]. Üheleherakendustel on mitmeleherakenduste ees veel mitmeid eeliseid:

- Kuna üheleherakendused ei uuenda kogu lehte, vaid ainult vajalikke komponente, paraneb oluliselt veebisaidi kiirus.
- Üheleherakendused saavad vahemällu kohalikke andmeid salvestada. Serverisse saadetakse vaid üks päring ning saadud andmed salvestatakse. Nii saab veebilehte kasutada ka võrguühenduseta ning ühenduvuse taastumisel sünkroonitakse salvestatud andmed uuesti serveriga.
- Chrome pakub erinevaid JavaScripti raamistikele välja töötatud tööriistu, mille abil on arendajatel lihtsam üheleherakendusi ehitada, testida ja parandada [10].

Joonisel 1 on kujutatud üheleherakenduse ning mitmeleherakenduse elutsükkel.



Joonis 1. Üheleherakenduse ja mitmeleherakenduse elutsükkel [9].

4.3.1 Klientrakenduse raamistik

Klientrakendus loodi ülejäänud veebirakendusega sarnaselt Angular raamistikku kasutades. Angular on TypeScriptis kirjutatud arendusplatvorm üheleherakenduste loomiseks kasutades HTML-i (*Hyper Text Markup Language*) ja JavaScripti või mõnda keelt, mis kompileerub JavaScriptiks, näiteks TypeScripti [11]. Angulari platvorm koosneb järgmistest osadest:

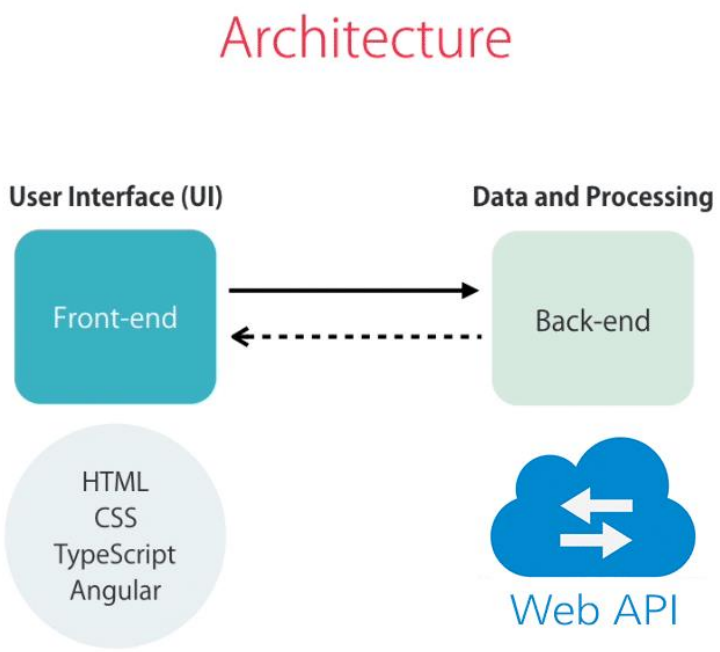
- Komponendipõhine raamistik ulatuslike veebirakenduste loomiseks.
- Sisseehitatud teekide kogum, mis hõlmab paljusid erinevaid funktsioone suunamiseks (*routing*), vormide haldamiseks ning kliendi ja serveri vaheliseks suhtluseks.
- Komplekt arendajatele mõeldud tööriistadest, mis hõlbustavad koodi kirjutamist, testimist ja uuendamist.

Angular on sobilik platvorm erinevate suurustega rakenduste arendamiseks, võimaldades hästi ka ettevõtte tasemel arendamist. Angulari eesmärk on koodi värskendamine võimalikult lihtsaks muuta, et uute komponentide integreerimine mööduks suurema vaevata. Kuna Angulari kasutusringkonda kuulub üle 1,7 miljoni arendaja, teekide autori ja sisulooja, siis leiab probleemide tekkimisel neile internetist hõlpsasti lahenduse [12].

Angulari raamistik on loodud AngularJS baasil. AngularJS ehk Angular 1.x on JavaScripti põhine avatud lähtekoodiga raamistik klientrakenduste arendamiseks. Selle eesmärk on lihtsustada nii rakenduste väljatöötamist kui ka testimist, pakkudes raamistikku kliendipoolse MVC (*model-view-controller*) ja MVVM (*model-view-viewmodel*) arhitektuuridele koos erinevate veebirakendustes laialdaselt kasutatavate komponentidega.

Arhitektuuriliselt on uuemad Angulari versioonid AngularJS-st erinevad. Kui AngularJS arhitektuur põhineb MVC või MVVM baasil, siis Angular kasutab peamiselt komponente (*components*) ehk HTML mallidega direktiive (*directives with templates*). Esineb kahte tüüpi direktiive: struktuursed direktiivid, mis muudavad DOM-i (*data object model*) kujundust selle elemente asendades ning atribuutdirektiivid, mis muudavad DOM-i käitumist ja selle kujundust [13].

Joonisel 2 on esitatud Angular raamistikus arendatud veebirakenduse lihtsustatud arhitektuur.



Joonis 2. Angular raamistiku lihtsustatud arhitektuur [14].

Kuna Angular on kirjutatud TypeScriptis, siis soovitatakse Angulari raamistikku kasutades samuti TypeScripti kasutada. Seda on tehtud ka käesoleva süsteemi kliendirakenduse puhul. TypeScript kompileerub JavaScriptiks, kuid sellele on lisatud staatilised tüübid, mis võimaldavad kirjutada töökindlamat koodi kui JavaScriptis. Tänu staatilistele tüüpidele on lihtsam tulevikus koodi ümber kujundada ning vigade tekkimise tõenäosus on oluliselt väiksem [15].

4.3.2 Koodi ja stiili kontrollimine

Kliendirakenduse arendamise hõlbustamiseks on kasutusele võetud erinevad tööriistad, mis aitavad süntaksi- ning stiilivigu koodis vältida. Selleks on kasutusel Visual Studio arenduskeskkonda integreeritav laiendus Resharper, mis on loodud arvutitarkvara ettevõtte JetBrains poolt.

Resharper osutus valituks just seetõttu, et sama tööriista kasutavad ka teised projekti tiimi kuuluvad arendajad ning tiimis töötamise puhul on tähtis, et kõik liikmed hoiaksid koodi

kirjutamisel sarnast stiili. Nii püsib rakenduse kood arusaadavana ning vigade leidmine on lihtsam.

Resharper lihtsustab koodi kirjutamist erinevatel viisidel ning toetab programmeerimiskeeli C#, VB.NET, XAML, ASP.NET, ASP.NET MVC, JavaScript, TypeScript, CSS, HTML, ja XML. Resharper teostab koodi kvaliteedi analüüsimist jooksvalt ning võimaldab tuvastada vigu ja halba stiili kirjutamisel ajal. Lisaks sellele, et see annab arendajale koheselt teada, kui koodis on probleeme, pakub see välja erinevaid variante probleemide automaatseks lahendamiseks, mille seast arendaja saab endale sobivaima valida. Samuti pakub Resharper soovitusi stiili osas, näiteks muutujate ja klasside nimede määramisel.

Resharper võimaldab kiiret navigeerimist ja elementide otsimist üle terve lahenduse. Väga lihtsalt on võimalik üles leida mistahes fail, tüüp või tüübi liige või navigeerida konkreetselt elemendilt selle kasutusviiside, põhi- ja tuletatud elementide või funktsioonide juurde.

Resharperis on vaikimisi määratud kindlad reeglid koodistiili osas, kuid neid on võimalik enda eelistuste kohaselt modifitseerida. Samuti saab importida või eksportida valmisolevaid stiilireeglite komplekte. Seda tehti ka käesoleva töö puhul, sest teistel tiimiliikmetel olid kindlad reeglid varasemalt paika seatud [16].

4.3.3 Klientrakenduse kujundus

Klientrakenduse disainimiseks kasutati Bootstrapi. Bootstrap on esirakenduse (*frontend*) raamistik veebirakenduste ja veebilehtede kujundamiseks. Bootstrap sisaldab HTML-i ja CSS-i põhiseid kujundusmalle tüpograafia, vormide, nuppude, tabelite, navigeerimise, moodulite ja paljude teiste disainielementide jaoks. Samuti sisaldab see erinevaid JavaScripti pistikprogramme. Bootstrap võimaldab lihtsasti arendada seadme ekraanisuurusele kohanduvaid rakendusi. Bootstrap on rakendustesse lihtsasti integreeritav ning see ühildub kõikide kaasaegsete veebibrauseritega [17].

Bootstrapi kasulikkus põhineb ka selle kohandatavusel. Bootstrapi koduleheküljel on võimalik valida, millist funktsionaalsust soovitakse alla laadida. Samuti saab kohandada erinevaid komponentide parameetreid, näiteks suurust, värvi ja fondistiili. Seda on tehtud ka käesoleva töö raames arendatud süsteemi puhul. Vaikimisi Bootstrapi sisse ehitatud

parameetrid on muudetud projektiga kohanduvaks, et hoida ühtset kujundust terves veebirakenduses.

4.4 Serverirakendus

Serverirakenduse ehk süsteemi tagarakenduse (*back-end application*) realiseerimiseks on kasutatud .NET raamistikku ning C# programmeerimiskeelt. Arenduskeskkonnana kasutati Visual Studiot.

4.4.1 Serverirakenduse loomine .NET platvormil

.NET on arendusplatvorm, mis koosneb tööriistadest, programmeerimiskeeltest ja teekidest mitmesugust tüüpi rakenduste loomiseks. .NET-ist on erinevaid implementatsioone, millest igaüks võimaldab .NET-koodi käivitada erinevates operatsioonisüsteemides:

- .NET Framework on .NET-i algne implementatsioon. See toetab veebisaitide, teenuste, töölauarakenduste ja muu käitamist Windowsis.
- .NET Core on platvormide vaheline rakendus veebisaitide, teenuste ja konsoolirakenduste kasutamiseks Windowsis, Linuxis ja macOS-is. .NET Core on saadaval avatud lähtekoodina.
- Xamarin/Mono on .NET-i implementatsioon rakenduste käivitamiseks suuremates mobiilioperatsioonisüsteemides, sealhulgas iOS-is ja Androidis [18].

.NET Standard on API-de baaskomplekt, mis on ühine kõigile .NET-i rakendustele. Iga implementatsiooniga võib lisanduda ka täiendavaid API-sid, mis on iseloomulikud operatsioonisüsteemidele, milles see töötab. Näiteks .NET Framework on mõeldud ainult Windowsile ning seega sisaldab API-sid Windowsi alamtaseme seadistustele juurdepääsemiseks.

.NET rakendusi on võimalik kirjutada C#, F# ja Visual Basic keeltes [19]. Antud süsteem on kirjutatud C# keeles. C# on üldotstarbeline objekt-orienteeritud (*object-oriented*) ja tugevalt tüübitud (*type-safe*) programmeerimiskeel. C# ja selle teostused toetavad tarkvaraarenduse põhimõtteid nagu tugev tüüpimine, massiivi piiride kontrollimine, väärtustamata muutujate kasutamise avastamine ning automaatne mälukoristus. C# on

sobilik rakenduste kirjutamiseks erinevates süsteemides, alates keerulistest operatsioonisüsteeme kasutavatest süsteemidest ja lõpetades manussüsteemidega [20].

4.4.2 Andmebaas

Post11 veebirakendus kasutab andmete salvestamiseks PostgreSQL andmebaasisüsteemi. PostgreSQL on avatud lähtekoodiga objekt-relatsiooniliste andmebaaside haldamise süsteem. See loodi aastal 1986 Berkeley ülikoolis POSTGRES projekti raames ning seda on aktiivselt arendatud üle 30 aasta. PostgreSQL kasutab SQL keelt, toetab erinevate funktsioonide kasutamist ja protseduuride salvestamist. See suudab korraga hallata suuremahulisi andmehulki ning võimaldab samaaegset kasutamist paljude kasutajate poolt. PostgreSQL töötab kõigis suuremates operatsioonisüsteemides, on ACID põhimõttega ühilduv alates 2001. aastast ning sellel on võimsad lisandmoodulid, näiteks georuumiliste andmebaaside laiendaja PostGIS [21].

Andmede haldamiseks koodis ja objektide kaardistamiseks kasutati *Entity Frameworki*. EF on Microsofti toetatud avatud lähtekoodiga ORM-i raamistik .NET rakendustele. See võimaldab töötada andmetega domeenispetsiifiliste klasside objektide abil, keskendumata andmebaasis olevatele tabelitele ja veergudele, kuhu andmed on salvestatud. EF on vahekiht andmebaasisüsteemi ning rakenduse ärikihi (*business-layer*) vahel. See võimaldab kasutada LINQ päringuid andmebaasist andmete hankimiseks. EF pakub migreerimiskäskude komplekti, mida saab käivitada NuGet paketihalduri (*NuGet Package Manager*) konsoolis või käsurea liideses, et luua või hallata aluseks oleva andmebaasi skeemi [22].

5 Tulemused

Antud töö tulemusena valmis postisaadetisi vedavate kaubaautode kulujaotussüsteem. Selle abil on võimalik infosüsteemist alla laadida eeltäidetud andmetega Exceli fail ning hiljem täielikult andmetega täidetud fail üles laadida. Selleks vajalikud alla- ning üleslaadimisnupud on esitatud joonisel 3. Üleslaetud faili alusel luuakse korrektsete andmete põhjal maksekirjed kõikide failis dokumenteeritud veoautode kulude kohta. Autode kulud on nendega seotud klientettevõtete vahel proportsionaalselt ära jagatud.

Joonis 3. Maksekirjete tabeli üles- ja allalaadimisnupud.

Kulujaotussüsteem on integreeritud olemasolevasse ettevõtte veebirakendusse veoautode terminalikülastuste haldamise komponenti. Seal on kasutajale hea ülevaade kõikide terminalikülastuste maksekirjetest ning kuluobjektide kogumaksumustest. Iga kaubaauto kulude summat kuvatakse ka auto detailvaates. Auto kogukulu leitakse kõikide selle autoga teostatud külastuste maksumuste summeerimise teel.

Joonisel 4 on kujutatud terminalikülastuse detailvaates antud külastusega seotud maksekirjete tabelit.

Actual truck ID	Direction	Destination	Cost type	Unit type	Unit price	Total price
123ABC	Inbound	—	TRAN	Kg	5.00 EUR	25.00 EUR
123ABC	Inbound	—	T1	Kg	5.00 EUR	50.00 EUR

Joonis 4. Maksekirjete tabel kaubaauto terminalikülastuse detailvaates.

5.1 Võrdlus varasema süsteemiga

5.1.1 Varasema metoodika kirjeldus

Varasemalt teostati veoautode kulude jaotamine ning haldamine käsitsi. Manuaalse töö peamised probleemid on suur vigade tekkimise tõenäosus ning aeglane töökiirus. Klientidele esitatavate arvete koostamiseks teostati järgmiseid toiminguid:

1. Veoautode terminalikülastuste info kopeeriti käsitsi veebirakenduse loendivaatest Exceli tabelisse.
2. Kuna kopeerimisel võisid väärtuste formaadid muutuda ebakorrektseks, pidi teostama korrektuure, näiteks hinna lahtritest kustutama teksti formaadis valuutad. Siin tekkis oht, et kustutatakse ka osa hinnast või terve lahtri väärtus. Samuti pidi jälgima, et kõik vastavad lahtrite väärtused saaksid kopeeritud õigesse tulpa.
3. Terminalikülastuste andmed filtreeriti Excelis partnerite nime järgi, et kuvada kõik antud partneri autodega teostatud külastused. See järel filtreeriti kuvatud külastused eraldi iga kliendi nime järgi. Siinkohal võis tekkida oht, et mõni partner või klient jäetakse filtreerimisel kasutamata.
4. Iga filtreerimise toimingu järel sai välja arvutada iga partneri autoga veetud iga kliendi kaupade koguhind. Summa arvutamiseks selekteeriti kõik vajalikud lahtrid maksumuste tulbas ja kasutati Exceli SUM() funktsiooni. Taaskord tekkis võimalus, et mõni vajalik lahter jääb selekteerimata ning summa arvutamisel arvestamata.
5. Pärast iga autoga veetud kaupade kogumaksumuse välja arvutamist liideti need iga kliendi kohta kokku ning saadud andmed sisestati raamatupidamisprogrammi.

Partnerite poolt esitatud arvete korrektsuse kinnitamist ei tehtud, vaid loodeti nende usaldusväärsuse peale. Sellega kaasnes aga risk, et partnerid esitavad liiga suuri arveid ning ettevõte maksab autode rentimise eest rohkem kui vaja oleks.

Enne kulujaotussüsteemi arendamist ei olnud veoautodega seotud kulud ettevõtte infosüsteemi salvestatud. Kuludega seotud andmed eksisteerisid kaudselt autode terminalikülastuste vaates ning väljaarvutatud hinnad raamatupidamissüsteemis. Raamatupidamissüsteem ei ole aga veebirakenduse andmebaasiga seotud. Seetõttu

puudus võimalus autode kuludega seotud andmeid analüüsida ning nende alusel statistilisi järeldusi teha.

5.1.2 Arendatud kulujaotussüsteemi eelised varasema metoodika ees

Kuna kaubaautode kulujaotussüsteem ehitati kindlate reeglite ja nõuete alusel, on vigade tekkimise tõenäosus oluliselt vähenenud. Selles veenduti süsteemi süvitsi testides. Enne süsteemi reaalselt kasutusele võtmist testiti seda nii arendaja kui ka testijate poolt ning valideeriti, et süsteem ei lubaks vigaseid andmeid salvestada ja teostaks kõik vajalikud toimingud korrektselt. Testimist kirjeldatakse detailsemalt järgmises alapeatükis.

Töötempo paranemist saab hinnata ligikaudselt. Selleks tuleb esmalt välja arvutada hinnanguline aeg, mis kulus varasema metoodikaga kulude välja arvutamiseks. Järgmises tabelis on kõrvutatud oletuslik aeg, mis võiks kuluda käsitsi toimingute teostamiseks 50 terminalikülastusega kogenud töötajal ja aeg, mis kulub samu toiminguid kulujaotussüsteemiga tegemiseks.

Tabel 1. Kaubaautode kulude jaotamise protsessi ajakulu manuaalse töö ja arendatud süsteemi puhul.

Toiming	**Ajakulu manuaalse tööga**	**Ajakulu süsteemi kasutades**
Töötlemata andmete kogumine Exceli tabelisse õigetesse lahtritesse	~ 1min	~ 1s
Käsitsi andmete töötlemine või vajalike andmete sisestamine Exceli tabelisse	~ 15min	~ 10min
Sisestatud andmete valideerimine	~ 15min	~ 3s
Iga partneri auto kulude välja arvutamine ja proportsionaalselt klientide vahel ära jagamine	~ 60min	~ 2s
Iga kliendi kulude summeerimine	~ 5min	~ 2s
Andmete sisestamine ja salvestamine infosüsteemi	~ 20min	~ 2s
Ajakulu kokku	**~ 116min**	**~ 10min 10s**

Seega võib hinnanguliselt väita, et ajavõit arendatud süsteemi kasutades on umbes 11,3 kordne. Selle arvestuse kohaselt kuluks igakuiseks kulude arvutamiseks aastas käsitsi tehes 1392 minutit ehk ligikaudu 23 tundi ning süsteemi kasutades 123 minutit ehk ligikaudu 2 tundi.

Arendatud kulujaotussüsteemi eeliseks varasema metoodika ees on ka see, et varem oli võimalik autode kogukulusid välja arvutada vaid kaalu alusel, kuid arendatud süsteemiga on seda võimalik teha ka kaubaaluste ja erikaalu põhjal.

5.2 Testimine

Testimine toimus jooksvalt töö autori poolt arenduse käigus kui ka pärast süsteemi valmimist ettevõtte testijate poolt testkeskkonnas. Testimise suund jagunes kolmeks: andmete valideerimine, süsteemi funktsionaalsuse testimine ning koormustestimine.

5.2.1 Andmete valideerimine

Andmete valideerimise testimise eesmärk on kontrollida, et süsteem tuvastaks kasutaja poolt üleslaetavast Exceli failist vigased ning lubamatud sisendid ning tagastaks nende kohta korrektsed veateated. Süsteem ei tohi luua uut kaubaauto kuluobjekti, kui selle maksekirjete seas on vähemalt üks vigane kirje.

Testimiseks täideti Exceli fail erinevate ebaõigete andmetega ja laeti süsteemi üles. Seejärel kontrolliti, et veateated oleksid korrektses vormingus ning tagastatud õige vea kohta. Samuti loodi selline fail, mis sisaldas ühe kuluobjekti alla käivaid maksekirjeid. Mõned neist olid korrektsed ning osad sisaldasid vigu. Kontrolliti, et sellisel juhul ühtegi maksekirjet süsteemi ei salvestataks ja kuluobjekti ei loodaks. Samuti testiti situatsioone, kus üleslaetav fail on vale laiendusega või ei sisalda ühtegi maksekirjet.

5.2.2 Funktsionaalsuse testimine

Funktsionaalsuse testimise puhul vaadati peamiselt, et kulude kogusumma arvutamisel ning kulude jaotamisel saadud tulemused oleksid korrektsed. Selleks laeti süsteemi üles nii kaalupõhise, kaubaaluste põhise kui ka erikaalupõhise hinna arvutamisega maksekirjeid. Saadud arvutustulemuste õigsust kontrolliti käsitsi üle arvutades.

Samuti valideeriti, et süsteem ei laseks salvestada õiges formaadis, kuid vastuolus olevaid andmeid. Näiteks peab süsteem tagastama veateate, kui samade andmetega kuluobjekt või maksekirje juba eksisteerib. Lisaks ei tohi ühe kuluobjekti raames olevatel maksekirjetel erineda teatud parameetrid, näiteks valuuta, hinna arvutamise viis ja tegelik auto kogumaksumus.

5.2.3 Koormustestimine

Koormustestimine teostati vaid töö autori poolt arenduse käigus. Esmalt testiti maksekirjete malli allalaadimise kiirust juhul, kui filtreerimisel kuvati autovisiitide vaates üle 700 visiidi. Sel juhul pidi süsteem läbi analüüsima üle 700 objekti ning salvestama kasutaja arvutisse Exceli faili, mis on eeltäidetud korrektsete andmetega. Kuna terminalikülastus võib sisaldada stoppe ning iga stopi kohta luuakse eraldi maksekirje rida, siis suurenes alla laetava malli suurus 800 reani. Sellise faili alla laadimine võttis aega ligikaudu 4 sekundit, mis on hea tulemus, sest tavaliselt piirdub maksekirjete tabeli suurus 50 reaga.

Üleslaadimise kiirust testiti eelnevalt alla laetud suuremahulist faili üles laadides. Mõned read täideti korrektsete andmetega, kuid suurem osa jäeti samasse seisu nagu need olid pärast alla laadimist või siis tühjendati ka eeltäidetud lahtrid. Ligikaudu 800-realise tabeli üles laadimiseks, selle kõikide ridade valideerimiseks, korrektsete andmetega ridadest maksekirjete ja kuluobjektide loomiseks ning veateadetega faili koostamiseks kulus süsteemil 16 sekundit. See tulemus on positiivne, kuna reaalselt nii suuri tabeleid üles laadima ei hakata ja aega kulub kordades vähem.

5.3 Edasiarenduse võimalused

5.3.1 Teavitusfunktsionaalsuse arendamine

Kulujaotussüsteemi täiustamise ning edasiarenduse võimalusi on mitmeid. Esiteks võiks süsteemi arendada teavitusfunktsionaalsuse, mis annaks ettevõtte töötajatele teada, kui mõne auto kohta pole maksekirjeid loodud, kuid see oleks vajalik. Kuna süsteem salvestab kõik loodud maksekirjed andmebaasi ning need on seotud kindlate kaubaautodega, oleks üsna lihtne analüüsida, millised saadetisi transportinud autod süsteemis veel eksisteerivad, kuid mille kohta pole ühtegi kirjet kulude kohta saadaval. Lisaks võiks teavituskomponent sisaldada raporteid, kus oleks täpselt välja toodud kõik puuduvate maksekirjetega kaubaautod ning nendega seotud visiitide detailid.

5.3.2 Süsteemi sidumine ärianalüüsi tarkvaraga

Ülevaadet kulujaotussüsteemi poolt salvestatud andmetest võiks tulevikus olla võimalik kuvada andmete visualiseerimiseks mõeldud programmis, näiteks Power BI. Nii oleks lihtsam analüüsida erinevate partnerite autode peale kulunud ressurssidest ja teha

statistilisi järeldusi, milliste partneritega on soodsam koostööd teha. Samuti saaks kogutud andmete põhjal teha hinnangulisi ennustusi tuleviku kulude osas ja tänu sellele ettevõttele paremaid valikuid teha.

Power BI on sarnase eesmärgiga loodud tarkvaralahendustest valitud just seetõttu, et tegemist on Microsofti teenusega ja muud ettevõtte süsteemid on samuti Microsofti teenuste baasil ehitatud. Power BI on tarkvarateenuste, -rakenduste ja -ühenduste kogum, mis seob üksteisega erinevaid andmeallikaid. Selle abil saab andmeid visualiseerida ja neid interaktiivselt hallata. Tarkvaraarenduses saab kasutada Power BI API-sid andmete kogumiseks andmehulkadesse või lisada enda programmi kohandatud raportite koostamise komponente [23].

6 Kokkuvõte

Antud töö eesmärgiks oli luua postisaadetisi vedavate kaubaautode kulude jaotamist ning haldamist lihtsustav süsteem, mis aitaks vähendada manuaalse töö tegemist. Seejuures pidi süsteem kiirendama ettevõtte töötempot ning vältima inimlike vigade tekkimist. Lisaks pidi süsteemi kasutajaliides olema intuitiivne ning kasutajasõbralik.

Eesmärkide täitmiseks arutleti süsteemi ülesehituse põhimõtteid nii ettevõtte arendustiimiga kui ka finantsosakonna liikmetega, kes seda süsteemi igapäevaselt kasutama hakkavad. Püstitati funktsionaalsed ning mittefunktsionaalsed nõuded arendatavale kulujaotussüsteemile. Seejärel tutvuti olemasoleva veebirakenduse, selle arhitektuuri ning kasutatud tehnoloogiatega. Kulujaotussüsteemi klientrakendus realiseeriti üheleherakendusena ning selleks kasutati Angular raamistikku. Süsteemi tagarakenduse ehitamiseks kasutati .NET platvormi ning C# programmeerimiskeelt. Andmevahetus klient- ja serverirakenduse vahel põhineb REST arhitektuuril. Sarnaselt ülejäänud veebirakendusega kasutab arendatud süsteem andmete salvestamiseks PostgreSQL andmebaasisüsteemi.

Pärast süsteemi valmimist testiti seda süvitsi nii töö autori kui ka testijate poolt, et veenduda süsteemi töökindluses ja -võimekuses. Testiti andmete valideerimise korrektsust, süsteemi funktsionaalsust ja andmete käsitlemise kiirust. Testimise tulemuste alusel saab öelda, et süsteem saavutas eesmärgi kiirendada ettevõtte igapäevast töötempot ning vähendada vigade tekkimise tõenäosust.

Arendatud süsteem integreeriti ettevõtte reaalsesse veebikeskkonda, kus see on kättesaadav kõikidele ettevõtte töötajatele, kellele on määratud vastavad õigused süsteemile ligi pääsemiseks. Finantsosakonna liikmed ehk süsteemi igapäevased kasutajad kinnitasid süsteemi kasutamismugavust.

Kasutatud kirjandus

[1] LinkedIn, *Post11 Ltd.: About* [Online]. Loetud aadressil: https://www.linkedin.com/company/post11/about. Kasutatud: 10.04.2021.

[2] Omniva, *Omniva kontern*. [Online]. Loetud aadressil: https://www.omniva.ee/meie/ettevottest. Kasutatud: 10.04.2021.

[3] Post11, *About*, 2016. [Online]. Loetud aadressil: https://www.post11.com/#About. Kasutatud: 10.04.2021.

[4] *P11 Video*, 2016. [Online]. Loetud aadressil: https://www.youtube.com/watch?v=EZ_FJY03zUo. Kasutatud: 10.04.2021.

[5] Tallinna Ülikool, HITSA, *Tarkvara analüüs ja testimine*. [Online]. Loetud aadressil: https://web.htk.tlu.ee/digitaru/testimine/chapter/tarkvara-arendusnouded/. Kasutatud: 10.05.2021.

[6] Codecademy, *What is REST?*. [Online]. Loetud aadressil: https://www.codecademy.com/articles/what-is-rest. Kasutatud: 10.04.2021.

[7] M. Elkstein, Blogger, *Rest Elkstein*, 2008. [Online]. Loetud aadressil: http://rest.elkstein.org/. Kasutatud: 10.04.2021.

[8] K. Lawson, Bloomreach, *What Is a Single Page Application and Why Do People Like Them so Much?*. [Online]. Loetud aadressil: https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html. Kasutatud: 10.04.2021.

[9] A. Padmanabhan, Devopedia, *Single Page Application*, 2020. [Online]. Loetud aadressil: https://devopedia.org/single-page-application. Kasutatud: 15.05.2021.

[10] A. Z., RubyGarage, *What's the Difference Between Single-Page and Multi-Page Apps*. 2018. [Online]. Loetud aadressil: https://rubygarage.org/blog/single-page-app-vs-multi-page-app. Kasutatud: 09.05.2021.

[11] Angular, *Introduction to the Angular Docs*. [Online]. Loetud aadressil: https://angular.io/docs. Kasutatud: 11.04.2021.

[12] Angular, *What is Angular?*. [Online]. Loetud aadressil: https://angular.io/guide/what-is-angular. Kasutatud: 11.04.2021.

[13] NgDevelop, *Angular History*, 2017. [Online]. Loetud aadressil: https://www.ngdevelop.tech/angular/history/. Kasutatud: 11.04.2021.

[14] DotNetTec, *Angular Architecture Overview*. [Online]. Loetud aadressil: https://dotnettec.com/angular-architecture/. Kasutatud: 10.05.2021.

[15] B. Wilson, Pluralsight, *8 essential skills for Angular web developers*. [Online]. Loetud aadressil: https://www.pluralsight.com/blog/software-development/essential-skills-angular-devs. Kasutatud: 11.04.2021.

[16] Jetbrains, *How ReSharper helps Visual Studio users*. [Online]. Loetud aadressil: https://www.jetbrains.com/resharper/. Kasutatud: 11.04.2021.

[17] W3schools, *Bootstrap Get Started.* [Online]. Loetud aadressil: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp. Kasutatud: 10.05.2021.

[18] Microsoft, *What is .NET Framework?.* [Online]. Loetud aadressil: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework. Kasutatud: 11.04.2021.

[19] Microsoft, *What is .NET?.* [Online]. Loetud aadressil: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet. Kasutatud: 26.04.2021.

[20] Wikipedia, *C Sharp*, 2012. [Online]. Loetud aadressil: https://et.wikipedia.org/wiki/C_Sharp. Kasutatud: 11.04.2021.

[21] PostgreSQL, The PostgreSQL Global Development Group, *About.* [Online]. Loetud aadressil: https://www.postgresql.org/about/. Kasutatud: 26.04.2021.

[22] Entity Framework Tutorial, *What is Entity Framework?*. [Online]. Loetud aadressil: https://www.entityframeworktutorial.net/what-is-entityframework.aspx. Kasutatud: 26.04.2021.

[23] Microsoft, *What is Power BI?*, 2021. [Online]. Loetud aadressil: https://docs.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview. Kasutatud: 09.05.2021.

[24] K. Shah, Thirdrock Techkno, *Single-Page Apps vs Multi-Page Apps: What To Choose For Web Development*, 2020. [Online]. Loetud aadressil: https://www.thirdrocktechkno.com/blog/single-page-apps-vs-multi-page-apps-what-to-choose-for-web-development/. Kasutatud: 09.05.2021.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks1

Mina, Teele Kaldaru

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Postisaadetisi vedavate kaubaautode kulujaotussüsteemi arendamine“, mille juhendaja on Toomas Lepikult
   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.

17.05.2021

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.