TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Enricqe Rene Sissask IADB1941371

Arveldussüsteemi loomine Uptime OÜ näitel

Bakalaureusetöö

Juhendaja: Einar Kivisalu
MSc

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Enricqe Rene Sissask

05.01.2023

Annotatsioon

Käesolevas lõputöö eesmärgiks on leida Uptime OÜ ettevõttele lahendus igakuiselt tekkivate arvete haldamiseks. Ettevõttes on olnud kasutusel vanem aegunud süsteem, mis enam ei kata ettevõtte ärilisi vajadusi.

Probleemi lahendamiseks kogus autor esialgsed süsteemi nõuded ja seejärel analüüsis pärandsüsteemi. Samuti uuriti sarnaseid olemasolevaid süsteeme, mida saaks ettevõte kohe kasutusele võtta. Kõigil uuritud süsteemidel olid omad plussid ja miinused aga need ei olnud kooskõlas ettevõtte nõuetega.

Seejärel tegi autor uue MVP arvete haldussüsteemi, mis oleks kooskõlas ettevõtte nõuetega. Pärast rakenduse valmimist vahetati pärandsüsteem uue vastu.

Tulemuste kaardistamiseks viidi läbi uue rakenduse kasutajate seas veebiküsitlus. Sellest tulenes, et kõik uue rakenduse kasutajad on sellega väga rahul ning leiavad, et see aitab neil igapäevaste arvete haldustegevustega seoses väga palju aega kokku hoida. Samuti on rakenduse haldamine ja muudatuste tegemine ettevõttele tulevikus palju kergem ja soodsam.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 5 peatükki, 12 joonist, 1 tabel.

Abstract

Creation of a web application for invoice management in Uptime OÜ

The main objective of this thesis is to solve Uptime's invoice management problem. The company generates massive amounts of invoices every month and they were using an obsolete legacy in-house system to manage invoices. Unfortunately, the legacy system doesn't cover all the company's business needs and it's missing a lot of features. Allocating resources to modernize and to add new features to the application isn't commercially viable, thus they need a new system to manage their invoices.

Firstly, the author gathered initial functional and non-functional requirements for the system. Secondly, an examination of the existing system was conducted to map its main drawbacks. Thirdly, similar existing systems were researched to check if any of them are in correspondence with company's needs. All the researched systems had different pros and cons, but in the end, they were not suitable for the company since they didn't have features that were required.

Afterwards a new MVP system was created according to the requirements. During the development of the new application, a strong emphasis was placed on making the application maintainable.

After replacing the legacy application with the new one a survey was conducted among the users in order to map the results of the new application. The survey results show that the new application's UI and UX is better. Most of the users answered that the new system is helping them to save a lot more time when managing invoices since it's a lot faster and easier to use, thus making it profitable for the company. It's also a lot easier and cheaper for the company to maintain the application and to implement features in the future because the new application is modernized, and its technological stack is in correspondence with the company's developer technological skills.

The thesis is in Estonian and contains 27 pages of text, 5 chapters, 12 figures, 1 table.

Lühendite ja mõistete sõnastik

AJAX	*Asynchronous Javascript and XML*, asünkroonne Javascript ja XML
API	Application Programming Interface, kasutajaliides
ASP.NET Core	Raamistik veebirakenduste kirjutamiseks
AutoMapper	Teek, mis lihtsustab objektide vahelist kaardistamist loodud profiilide põhjal
Azure	Microsofti poolt pakutavad pilveteenuste keskkond
Azure Active Directory	Microsofti poolt pakutav pilvepõhine identiteedilahendus
Azure Key Vault	Pilvepõhine salasõnade hoiustamise keskkond
Dockeri pilt	Fail, mis käivitab dockeri konteineris oleva koodi
DRY	*Don't repeat yourself*, koodi kirjutamise printsiip, mis rõhutab, et koodi kirjutamisel tuleks vältida duplikatsioone
HTML	*HyperText Markup Language*, hüperteksti märgistuskeel
HTTP	*Hypertext Trasnfer Protocol*, hüperteksti edastusprotokoll
IP	*Internet Protocol*, arvutivõrgus oleva seadme identifikaator
LINQ	*Language-Integrated Query*, päringukeel
LTS	*Long-term support*, pika ajalise toetusega
Microsoft Graph	Rakendusliides, millelt saab küsida erinevate pilveteenuste kohta andmeid
MVC	*Model-view-controller*, mudel-vaade-kontroller disainimuster
MVP	*Minimal viable product*, minimaalne töötav toode
.NET	Microsofti poolt pakutav tarkvaraarendusplatvorm
OpenID Connect	Autentimise protokoll, mis tugineb OAuth 2.0 protokollil
OWASP	*Open Web Application Security Project*, projekt mille eesmärgiks on tõsta teadlikkust rakenduste turvaohtude eest
PL/SQL	Protseduuriline keele laiendus struktuurpäringukeelele
Razor süntaks	Märgistussüntaks, mille abil on võimalik kirjutada C# koodi esisüsteemi vaadete sisse
Serilog	Teek, mis lihtsustab logimist
SQL	Struktuurpäringukeel
Veebihaak	HTTP protokollil põhinev tagasikutse

XML	*Extensible Markup Language*, laiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Uptime OÜ on pidevalt edasi arenev tarkvara arendusega tegelev ettevõte. Seoses pideva arenguga on ettevõtte käive tunduvalt tõusnud ning neil oleks vaja sissetulekute kui ka väljaminekute haldamise jaoks firmasisest süsteemi. Siiamaani on ettevõte kasutanud arvete haldamiseks vanemat süsteemi. Kahjuks pärandsüsteem ei kata kõiki ettevõtte vajadusi ning seetõttu on neil uut süsteemi vaja.

Käesolev lõputöö annab põhjaliku ülevaate uue arveldussüsteemi veebirakenduse loomisest Uptime OÜ ettevõtte näitel.

Töö esimeses peatükis tutvustatakse ettevõtet ning nende probleemi lähemalt. Seejärel püstitatakse eesmärk ning kirjeldatakse metoodikaid, mille abil eesmärk täidetakse. Samuti kirjeldatakse töö skoopi ehk projekti ulatust.

Teises jaotises analüüsitakse detailsemalt olemasolevat ettevõttes kasutatavat süsteemi ning tuuakse välja selle peamised puudused. Samuti püstitatakse uue lahenduse lähtetingimused ja antakse ülevaade uue rakenduse nõuetest. Seejärel uuritakse, et kas hetkel on turul sarnaseid lahendusi, mis oleksid kooskõlas püstitatud lähtetingimuste ja nõuetega.

Kolmandas osas kirjeldatakse loodud süsteemi arhitektuurilisi valikuid ning tutvustatakse erinevaid võtteid ja tehnoloogiaid, mida autor kasutas tagasüsteemi, esisüsteemi, andmebaasi, pideva integratsioon ja pidevvalmiduse loomisel. Samuti antakse ülevaade ka rakenduse testimise ja selle käitumise jälgimise võimaluste kohta.

Neljandas peatükis antakse ülevaade valminud rakenduse tulemuste kohta, kirjeldatakse kuidas uus süsteem on ettevõttele kasumlik ja samuti analüüsitakse rakenduse kasutajate vahel läbi viidud küsimustiku tulemusi. Samuti kirjeldatakse uue süsteemi tuleviku arendusplaane.

1.1 Ettevõtte taust

Uptime on ettevõte, mis loodi aastal 1992 ning selle peamisteks eesmärkideks on erinevatele klientidele pakkuda teenuseid alates analüüsist ja disainist, kuni prototüüpide loomise, tarkvaraarenduse, monitooringu ja hoolduseni välja. Samuti on organisatsiooni eesmärgiks nii klientide kui ka töötajatega pikaajaliste suhete loomine ning nende heaolu tagamine [1]. Uptime keskendub peamiselt Microsofti platvormide poolt loodud ärilahendustele ning on olnud Microsoftiga juba mitu aastat kuldpartnerid [2].

1.2 Probleemi tutvustus

Probleem seisneb selles, et ettevõttel tekib iga kuu väga palju arveid nii erinevate klientidega kui ka firmas kasutavate teenuste tagajärjel, näiteks pilveteenuste, majutusteenuste, integreeritud keskkondade ja litsentside puhul. Kõikide arvete haldamiseks oleks vaja tsentraalselt firmasisest arvete haldussüsteemi, milles oleks projektijuhtidel, tiimijuhtidel ja teistel isikutel võimalik ligi pääseda, et neil oleks detailne ülevaade firma sisse- ja väljaminekute kohta.

Praegune ettevõttes kasutusel olev arveldussüsteem on väga vana ning selles esineb palju vigu ning see ei kata kõiki ärilisi vajadusi. Samuti puudub ettevõttel vabu ressursse, et pärandsüsteemi hooldada ja sinna uusi funktsioone juurde lisada.

1.3 Eesmärk

Antud töö eesmärgiks on analüüsida olemasolevat arvete haldussüsteemi ja sarnaseid teisi loodud lahendusi. Samuti kaardistada ettevõtte vajadused ning nende põhjal luua uus arveldussüsteem. Uue lahenduse puhul on tähtis, et seda oleks võimalik tulevikus kergelt ja odavalt hooldada, et ettevõte ei peaks firmaväliselt ressursse sisse ostma, et selles uusi muudatusi teha. Süsteem peab olema kooskõlas tänapäevaste kui ka ettevõttes ette määratud koodi kirjutamise parimate praktikatega.

1.4 Metoodika

Esialgu analüüsis autor olemasolevat süsteemi ning tõi välja selle peamised puudused. Seejärel kogus autor ettevõtte arhitektilt, projekti-, tiimi-, tehnoloogia- ja tegevjuhilt esialgsed uue rakenduse funktsionaalseid kui ka mittefunktsionaalseid nõudeid. Uue süsteemi nõuete kogumine ettevõtte töötajatelt toimus intervjuu käigus. See toimus perioodiliselt ehk kui autor sai mingi etapi valmis, vaadati see koos üle ning anti tagasisidet selle kohta ja vajadusel muudeti esialgseid nõudeid või lisati uusi.

Autor analüüsis ka sarnaseid olemasolevaid lahendusi ning tõi välja nende peamised eelised ja puudused. See annab ülevaate, et kas turul on olemas sarnaseid lahendusi, mis kataksid ettevõtte vajadusi ning mida saaks ettevõte kohe kasutusele võtta.

Rakenduse valmimisel võeti uus süsteem ettevõttes kasutusele ning autor viis rakenduse kasutajate vahel läbi veebiküsitluse. Selle abil sai paremini kaardistada uue rakenduse tulemusi ja koguda tagasisidet, et oleks ka ülevaade uue rakenduse puuduste kohta.

1.5 Töö skoop

Käesoleva lõputöö skoobis on uue süsteemi MVP versiooni loomine ja selle kasutusele võtmine ettevõttes. Autori teha oli uue rakenduse taga- ja esisüsteem. Samuti oli töö skoobis luua andmebaas ning disainida selle mudelite skeem. Skoobis oli ka pidevvalmiduse ja pideva integratsiooni lisamine ning rakenduse seadistamine Azure keskkonnas. Samuti kuulus töö skoopi ka rakenduse kasutajate käest nõuete kogumine.

Lõputöö skoopi ei kuulu kasutajaliidese disaini loomine. See oli juba ettevõtte poolt varasemalt loodud Figma keskkonnas ning selle alusel tegi autor esisüsteemi kujunduse. Samuti ei kuulu töö skoopi rakendusele integratsiooni testide loomine. Sellele loodi ainult unit testid.

2 Analüüs

Käesolevas peatükis analüüsitakse pärandsüsteemi ja ka teisi olemasolevaid lahendusi ning tuuakse välja nende peamised tehnilised kui ka ärilised eelised ja puudused ning hinnatakse, et kas need süsteemid kataksid kõiki ettevõtte vajadusi.

2.1 Olemasoleva süsteemi analüüs

Selleks et paremini kaardistada pärandsüsteemi puuduseid ja vigu, viis autor läbi olemasoleva süsteemi analüüsi. Selle käigus analüüsiti süsteemi koodibaasi ja viidi läbi suuline intervjuu pärandsüsteemi kasutajaga. Vana süsteemi analüüs annab parema ülevaate, et mida uues süsteemis vältida ning paremini teha saaks.

2.1.1 Pärandsüsteemi tutvustus

Olemasolev süsteem on väga vana ning loodi 2003. aastal ning viimane juurdearendus tehti süsteemile 2011. aastal. Rakenduse tagasüsteem on kirjutatud PHP skriptimiskeeles ning andmebaasiks on valitud Oracle. Andmebaasist päringute tegemiseks kasutati PL/SQL päringukeelt. Esisüsteem on loodud HTMLi, CSSi ja Javascripti abil. Selleks et hallata koodi ja failide muudatusi, kasutatakse tsentraliseeritud versioonihaldustarkvara Apache Subversion.

2.1.2 Koodibaasi analüüs

Selleks et oleks parem ülevaade pärandsüsteemi kohta, uuris töö autor lähemalt selle koodibaasi. Pärandsüsteem on kirjutatud vastavalt oma ajastule. See tähendab, et rakenduse kirjutamise ajal järgiti erinevaid koodi kirjutamise printsiipe, arhitektuurilisi disaine ja kasutati erinevaid disainimustreid kui tänapäeval. Kuna rakenduse viimane uuendus oli üsna ammu, siis on paljud tehnoloogilised lahendused aegunud. Selle tõttu ei tööta paljud esisüsteemi funktsionaalsused osades veebibrauserites korrektselt ja kasutajad peavad veebilehtede kuvamiseks kasutama spetsiifilisi veebibrausereid.

Pärandsüsteemi loomise hetkel ei olnud dünaamiline ega reageeriv veebilehe disain eriti aktuaalne. See on tingitud sellest, et sellel ajastul üldiselt kasutati sarnase suurusega kuvareid [3]. Seega loodi pärandsüsteemi veebidisain vastavalt olemasolevatele kuvarite suurustele ning ei arvestatud dünaamilise ega reageeriva disainiga. Selle tõttu on tänapäeval väiksemate kui ka suuremate kuvarite peal veebirakenduse kuvamine tunduvalt tülikam, sest paljud veebilehe elemendid on paigast ära.

Vana süsteemi lähtekoodi lähemalt uurides on näha, et koodi kirjutamisel on läbisegi kasutatud eesti ja inglise keelt, seda nii funktsioonide, muutujate, failide, andmebaasi skeemide ja dokumentatsiooni nimedes. See muudab koodi lugemise keerukamaks ning selle tõttu on keeruline juurdearendusi teha arendajatel, kes ei oska eesti keelt. Samuti leidub lähtekoodis paljusid koodi duplikatsioone. Martin väidab, et koodi duplikatsioonid on tarkvara süsteemide puhul kõige kurja juur. See muudab koodi lugevuse ja hallatavuse kordades keerulisemaks ning selle tõttu võib tekkida ka koodidefekte. Selle vältimiseks peaks koodi kordused refaktoreerima eraldi funktsioonideks või klassideks ning neid taaskasutama [4].

Tänapäeval järgivad kõik ettevõtte töötajad ettevõtte tarkvara arhitekti poolt loodud koodi kirjutamise printsiipe. See omakorda aitab suurel määral uue kui ka olemasoleva projekti kirjutamisel aega kokku hoida ning aitab üldist koodi kvaliteeti tõsta.

Pärandsüsteemil on olemas logimise võimalus aga selle puhul logitakse kõik asjad teksti faili. See lahendus omakorda muudab spetsiifiliste logide otsimise ja filtreerimise keeruliseks. Samuti puuduvad süsteemil automaattestid, mille tõttu võib uute muudatuste puhul kergelt tekkida uusi süsteemivigu.

Rakendusel puudus ka selle seadistamise kohta dokumentatsioon. Selle tõttu ei saanud autor seda lokaalses arvutis üles seadistada, et rakendust järgi proovida ja lähemalt analüüsida.

2.1.3 Veebirakenduse puudused kasutaja perspektiivist

Selleks et paremini kaardistada pärandsüsteemi puuduseid kasutaja perspektiivist, viis töö autor ühe pärandsüsteemi kasutajaga läbi suulise intervjuu. Selle käigus leiti, et tihtipeale ei olnud võimalik pärandsüsteemi kasutada, sest tekkisid rakenduse ühendusega probleemid. Pärandsüsteemil oli ka palju aegunud funktsioone, mida kasutajad enam ei

kasutanud ning need muutsid rakenduses navigeerimise keeruliseks. Tihtipeale oli vaja kogenud kasutajatel enda aega allokeerida uute töötajate õpetamiseks. Samuti ei olnud võimalik pärandsüsteemis olemasolevate arvete põhjal uusi luua ning puudus võimalus läbi kasutajaliideese olemasolevate arvetele määratud projekti muuta, selle jaoks pidi eraldi uue arve looma või otse andmebaasis muudatusi tegema.

Samuti oli kasutaja perspektiivist rakendus aeglane, mis tulenes sellest, et arvete, klientide ja projektide vaadetes laetakse ja kuvatakse kasutajale kõiki andmeid korraga. See omakorda suurendab ka andmebaasi koormust, sest päritakse suures koguses andmeid.

2.2 Uue lahenduse nõuded

Selleks et autoril oleks täpsem ülevaade ettevõtte uue arveldussüsteemi vajadustest, kogus autor ettevõtte kasutajatelt nõudeid. Need jagunevad funktsionaalseteks ja mittefunktsionaalseks nõueteks. Nõudeid koguti töö käigus iteratiivselt.

Funktsionaalsed nõuded selgitavad, et mida süsteem peab tegema. Need kirjeldavad täpsemalt, et milline on süsteemi käitumine ja väljundid teatud sisendite puhul [5].

Uue lahenduse funktsionaalseteks nõueteks on:

- Rakenduses peab olema delegeeritud sisselogimissüsteem, milles peab olema kaheastmeline autentimine.
- Rakendusele pääsevad ligi ainult kasutajad, kes on ettevõtte Azure Active Directory grupis.
- Rakendusel peab olema seadistatud pidevvalmidus ja pidevintegratsioon.
- Rakendusel peavad olema seadistatud automaattestid.
- Peab olema võimalus muuta, lisada ja kustutada kliente, projekte ja arveid.
- Arvete vaates peab olema kasutajatel võimalus filtreerida arveid klientide, projektide, tiimijuhtide ja kuupäeva põhjal.
- Arvete lehel peab olema võimalus eksportida filtrite põhjal arveid Exceli failiks.
- Projektide vaates peab olema võimalus otsida projekte nende nime, seotud kliendi või tiimijuhi nime järgi.
- Klientide vaates peab olema võimalik otsida kliente nende nime järgi.

- Arvete vaates peab nägema filtrite põhjal määratud arvete aruannet.
- Süsteem peab uue arve loomisel või vana arve muutmisel saatma sellest teavituse rakenduses seadistatud raamatupidajale.
- Kasutajal peab olema võimalus muuta lehe keelt.

Mittefunktsionaalsed nõuded omakorda pigem kirjeldavad, et kuidas süsteemi kvaliteedimõõtmed on seotud süsteemi lahendusega. Samuti iseloomustavad need nõuded tihtipeale ka süsteemi piiranguid [6].

Uue süsteemi mittefunktsionaalseteks nõueteks on:

- Tagasüsteem peab olema kirjutatud C# programmeerimiskeeles.
- Uus süsteem peab olema dokumenteeritud vastavalt ettevõtte poolt loodud dokumentatsiooni mallile.
- Rakenduse esisüsteem peab vastama ettevõtte poolt loodud esisüsteemi disainile.
- Rakenduse arhitektuur ja koodistruktuur peavad olema kooskõlas ettevõttes kasutatava koodi kirjutamise parima praktikate reeglitega.
- Uuel rakendusel peab olema kindel ülevaade rakenduse käitumise kohta, seega on monitoorimine ja logimine väga vajalikud.
- Rakendusel peab olema administraatori menüü, milles on võimalik konfigureerida rakenduse seadeid, näiteks vaikimisi valuuta väärtust, eemaldada ja lisada administraatoreid ja Active Directory gruppe.
- Rakendus peab olema majutatud ettevõtte Azure keskkonnas.
- Rakenduse esisüsteem peab olema dünaamilise ja reageeriva disainiga.

2.3 Sarnased olemasolevad lahendused

Autor uuris lähemalt kolme turul olemasolevat arvete haldamise tarkvara ning tutvustas nende peamiseid eeliseid ja puuduseid. See annab parema ülevaate, et kas hetkel on turul lahendusi, mis oleksid kooskõlas ettevõtte nõuetega.

2.3.1 Enty

Enty on raamatupidamistarkvara, mis pakub võimalust hallata ettevõtte arveid, mis automaatselt uue arve loomisel saadetakse kliendile. Samuti on sellega võimalik näha erinevaid aruandeid arvete kohta ja antakse võimalus monitoorida visuaalselt ettevõtte

finantsilist aktiivsust. Rakenduse puhul saab küll arveid hallata aga puudub võimalus lisada informatiivseid kommentaare arvete kohta ning ei paku ülevaadet arvete esitajate kohta ning samuti ei leidnud autor, et platvorm pakuks statistikat arvete esitajate ehk projekti-ja tiimijuhtide kohta [7]. Platvormi teenuse kasutamine on üsna kallis. Ettevõttele sobiv Pro versioon, mis toetaks kuni 50 arvete dokumenti kuus, hinnaks oleks 2244 eurot aastas [8].

2.3.2 Odoo

Odoo on avatud lähtekoodiga äritarkvara lahenduste platvorm. See pakub erinevaid võimalusi finantside, kliendisuhete ja müügitehingute haldamiseks. Samuti saab seal ka e-poode luua ja hallata. Raamatupidamise moodulis on võimalik luua nii kliente kui ka tooteid mille põhjal luuakse ja saadetakse arveid [9]. Samuti on selles olemas võimalus luua arveid erinevates valuutades. Selle platvormi abil on ka võimalik olemasolevaid kui ka uusi mooduleid ise muuta ja juurde programmeerida. Mooduleid luuakse Pythoni programmeerimiskeeles [10].

Autor ei leidnud, et sellel lahendusel oleks võimalus säilitada arve loomishetkel oleva valuuta väärtus eurodes, vaid näidatakse avre vaatamise hetkel oleva kursi põhjal ainult. Kursi väärtuseid saab küll manuaalselt spetsiifilise päeva kohta luua aga selline tegevus oleks väga ajarohke ja tülikas. Samuti ei olnud võimalik valida spetsiifilist APIt millest valuuta väärtused võetakse.

2.3.3 Aktiva Merit

Aktiva Merit on raamatupidamistarkvara, milles on võimalik ettevõtte finantse hallata. See pakub head võimalust näha aruandeid ettevõtte arvete ja müügianalüüsi kohta. Rakenduses saab ka kergelt hallata erinevaid kliente kui ka projekte.

Rakenduse puhul autor ei leidnud, et see toetaks olemasolevate arvete, klientide ega projektide sisse importimist. Samuti puudus võimalus näha arvete väärtust teistes valuutades.

2.3.4 Sarnaste lahenduste võrdluse kokkuvõte

Tänapäeval on turul väga palju erinevaid ettevõtetele arvete haldamise jaoks loodud lahendusi. Autor testis ja analüüsis kolme neist, et saada ülevaade nende võimalustest ja

uurida, et kas need oleksid kooskõlas ettevõtte vajaduste ja nõuetega. Neil kõigil olid omad plussid ja miinused ning pakkusid erinevaid võimalusi arvete haldamiseks. Nendest ei leidnud autor ühtegi sobivat lahendust ettevõttes olevale probleemile.

2.4 Uue ja vana süsteemi funktsioonide võrdlus

Uue rakendus loomine ei ole ainult platvormi vahetus, sellesse on lisatud palju uusi funktsioone, mis ei olnud pärandsüsteemis. Samuti on paljud pärandsüsteemi funktsioonid ümber tehtud, et uue rakenduse kasutamine oleks turvalisem ja vigade leidmine oleks võimalikult mugav ja kiire. Tabel 1 iseloomustab peamisi mõlema süsteemi funktsioonide sarnasusi ja erinevusi.

Tabel 1. Pärandsüsteemi ja uue süsteemi funktsioonide võrdlus.

Rakenduse funktsioon	**Uus süsteem**	**Vana süsteem**
Emaili teel teavituste saatmine	Uute arvete loomisel, muutmisel ja kustutamisel saadetakse arve muutja emaili konto pealt automaatselt teavitus raamatupidajale	Puudub
Dünaamilised ja reageerivad veebilehed	Olemas	Puudub
Kaheastmeline autentimine	Olemas	Puudub
Logimine	Logitakse andmebaasi, millest on mugav SQLi abil spetsiifilisi logisid otsida	Logitakse ainult tekstifaili
Töötajate töötundide logimine	Puudub, kuna seda funktsionaalsust enam ettevõttes ei kasutata	Olemas
Arvete muutmine pärast nende loomist	Olemas ning võimalik teha läbi kasutajaliidese	Puudulik, projekti ei saa pärast arve loomist enam muuta ja seda peaks manuaalselt otse andmebaasis tegema
Arvete kopeerimine	Olemas, eelnevate arvete põhjal on võimalik uusi koostada	Puudub

Rakenduse funktsioon	**Uus süsteem**	**Vana süsteem**
Automaattestid	Olemas	Puudub
Erinevate keelte tugi	Toetab nii eesti kui ka inglise keelt	Toetab ainult eesti keelt
Võimalus luua arveid erinevates valuutades	Olemas	Puudub
Klientide, projektide ja arvete haldamine	Olemas	Olemas
Päringute monitoorimine	Olemas	Puudub
Aruannete koostamine	Olemas	Olemas
Lehekülgede numeratsioon	Olemas, kasutatakse lõputut kerimist ehk kerimisel küsitakse uusi andmeid serverist ja need lisatakse Javascripti abil loendisse	Puudub ning selle tõttu on paljud lehed aeglased, sest korraga päritakse kõik andmed

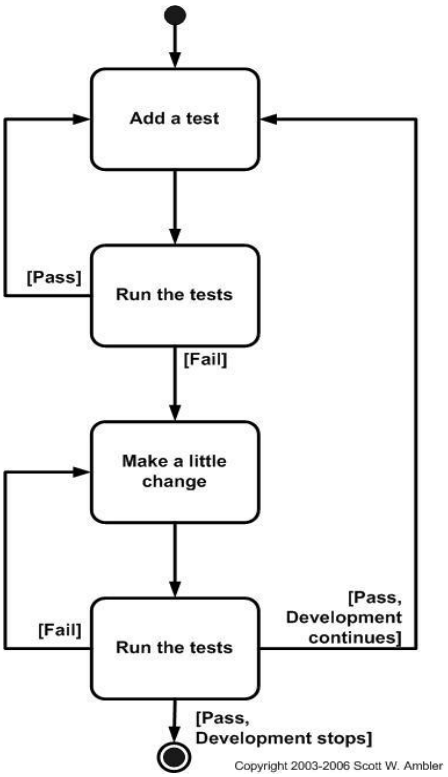
3 Rakenduse realiseerimine

Järgnev peatükk annab üldise ülevaate rakenduse arhitektuuri ning andmebaasi kohta. Samuti kirjeldatakse rakenduse esi- ja tagasüsteemi ning kuidas need omavahel seotud on. Tagasüsteemi arenduse puhul selgitab autor, et milliseid tehnoloogiaid kasutati, kuidas loodi sisselogimissüsteem, pidevvalmidus ja pidev integratsioon**.** Ühtlasi kirjeldatakse ka võimalusi kuidas saab uut rakendus monitoorida ja selgitatakse kuidas uuele rakendusele teste kirjutati.

3.1 Arendusmetoodika

Uue rakenduse loomisel kasutas autor peamiselt agiilset arendusmetoodikat. Arendus toimus iteratiivselt ehk kui autor sai mingi tööjupi valmis, vaadati see koos ettevõttega üle ning vajadusel muudeti seda.

Rakenduse arendusel kasutas autor *test-driven development* tarkvaraarenduse metoodikat. See põhineb sellel, et arendusel luuakse esialgu unit test, mis kirjeldab äriloogikat ning käivitatakse see test ja kinnitatakse, et see ebaõnnestub. Seejärel lisatakse äriloogika kood, mis ei ole üldiselt lõplik. Pärast muudatusi käivitatakse test uuesti ning nüüd peaks see läbima. Pärast edukat läbimist vaadatakse lisatud äriloogika kood uuesti üle ning vajadusel refaktoreeritakse see ja seejärel käivitatakse uuesti testid (Joonis 1). Selline arendusviis aitab varakult leida koodi defekte, sest teste käivitatakse tihedalt ja uute äriloogika funktsioonide jaoks luuakse üldiselt eraldi uued testid [11].



Joonis 1. Testi poolt juhitud arenduse tsükkel [11].

3.2 Andmebaas

Rakenduse andmeid hoitakse relatsioonilises SQL andmebaasis ning andmebaasi ennast hoiustatakse Azure pilvesüsteemi SQL serveris. Azure keskkond võimaldab kiirelt ja lihtsalt luua erinevaid SQL andmebaaside instantse. See on antud rakenduse puhul väga tähtis, sest erinevaid keskkondi on mitu tükki ning Azure keskkonna abil on nende haldamine väga mugav ja kiire. Samuti võimaldab see teenus uuendada ja muuta andmebaasimootori versioone automaatselt ning saab võimaldada automaatselt andmebaasi varukoopiate loomist [12].

Samuti on võimalik Azure SQL andmebaase kergelt skaleerida ning panna vastavalt vajadusele andmebaas võimekama masina peale. Võimekust mõõdetakse DTU ühikutes, mis on kombinatsioon mälu, protsessori, sisendi ja väljundi kirjutamise kiiruse mõõtudest [13]. Uuel süsteemil valiti andmebaaside DTU arvuks esialgu 5 aga see osutus tihtipeale liiga aeglaseks ja mindi üle 20 DTU peale, mis oli uue rakenduse jaoks ideaalne.

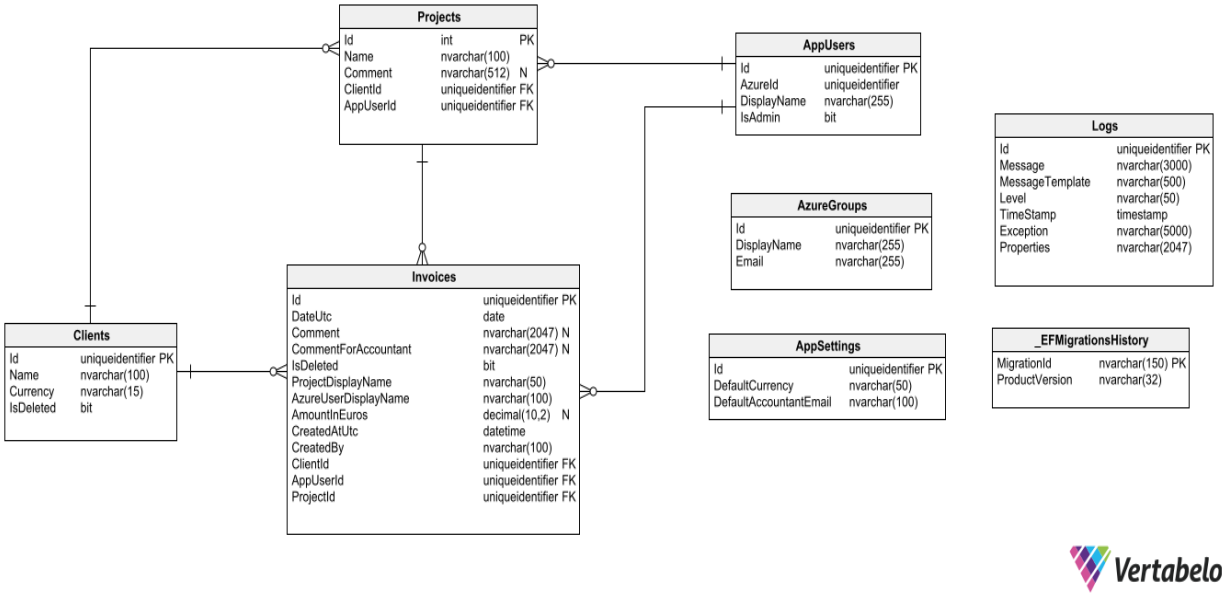
Kuna andmebaasis hoitakse sensitiivseid andmeid, on tähtis, et see oleks tugevalt turvatud. Selle jaoks on andmebaasi ligipääsemiseks seadistatud tulemüüri reeglid, mille

abil on võimalik piirata IP-aadresse, mis andmebaasile ligi saavad [14]. Sinna on lisatud veebirakenduse IP-aadressid, et rakendus saaks andmebaasiga suhelda. Samuti on seal ka ettevõtte domeeni IP-aadress, et oleks võimalik vajadusel ka manuaalselt andmebaasile ligi pääseda, et näiteks logisid lugeda või hooldusi läbi viia.

Andmebaasi olemite ja skeemi loomiseks kasutas autor koodi põhist lähenemist, mis tugineb Entity Framework raamistikul. Selle abil on võimalik migreerida koodis kirjeldatud ärimudelid andmebaaside olemiteks. Sellel viisil luuakse andmebaasi eraldi migratsioonide tabel, milles hoitakse migratsioonide kohta kirjeid ning on vajadusel võimalik erinevate versioonide peale andmebaasi skeemi migreerida [15].

Uue rakenduse andmebaas koosneb 8st tabelist (Joonis 2). Kõike rakenduse spetsiifilisi tabeleid hoitakse ühes skeemis. Iga keskkonna jaoks on loodud eraldi iseseisev andmebaasi instants. Rakenduse tabeliteks on:

- Invoices - Iseloomustab ettevõtte arveid.
- Projects – Kirjeldab ettevõtte arvetega seotud projekte.
- Clients – Iseloomustab ettevõtte projektidega seotud kliente.
- AppUsers – Kirjeldab rakenduse Active Directory kasutajaid. Tabelisse lisatakse kirje ainult siis kui uus kasutaja külastab esimest korda uut rakendust või kui administraator annab neile ligipääsu manuaalselt läbi administraatori menüü.
- AzureGroups – Määrab ära Active Directory grupid, mille kasutajad saavad veebirakendusele ligi pääseda.
- AppSettings – Kirjeldab rakenduse üldseadeid, et millisele meiliaadressile saadetakse arvete loomisel, muutmisel ja kustutamisel teavitus ning samuti määrab ära rakenduse vaikimisi käideldava ühiku.
- Logs - Rakenduse logid, mis hoiustatakse struktureeritud kujul.
- _EFMigrationHistory – Hoiustatakse rakenduse migratsiooni ajalugu.



Joonis 2. Uue rakenduse andmemudelid.

3.3 Tagasüsteemi arendus

Järgnev alapeatükk annab ülevaate uue rakenduse tagasüsteemi arendusest. Samuti tutvustatakse rakenduse arhitektuuri ning erinevaid teeke ja raamistike, mida autor kasutas rakenduse loomisel.

3.3.1 Tagasüsteemi arhitektuur

Uue rakenduse tagasüsteem on kirjutatud C# programmeerimiskeeles ning selle peamiseks põhjuseks on, et ettevõttes on peaaegu kõik tagasüsteemi arendajatest C# programmeerimiskeeles väga pädevad ja kogenud, seega rakenduse hooldamiseks kui ka uute funktsioonide lisamiseks saab kasutada oma ettevõtte töötajaid, mis on autori hinnangul palju odavam kui väljastpoolt ettevõtet ressursse sisse osta, et muudatusi teha.

Tagasüsteemi arhitektuur peamiselt tugineb *domain-driven* disainimustril, mis põhineb agiilsel lähenemisele tarkvara kirjutamisel. See arhitektuuriline disainimuster koosneb peamiselt neljast osast [16].

Esimeseks osaks on domeenikiht, mis on kogu süsteemi keskne osa ja selles on autor defineerinud kogu süsteemi ärireeglid ja vajadused. Selles osas on määratud erinevad kontseptsioonid, käitumised ja kindlad abstraktsed klassid, mida teised kihid peavad

pärima. Samuti on selles kihis rakenduse domeenimudelid, mis kirjeldavad äri olemust [17]. See kiht ei sõltu teistest kihtidest.

Teiseks osaks on infrastruktuuri kiht, mis sõltub domeenikihist. See koosneb peamiselt andmebaasi olemite seadistustest, erinevatest andmebaasi reeglitest ning andmebaasi olemite omavaheliste suhete kohta. Selles kihis on ka väliste teenuste kui ka rakenduse enda ärireeglite seadistus, mis on defineeritud domeenikihis.

Samuti on infrastruktuuri kihis rakenduse andmekontekst, mille abil suheldakse andmebaasiga. Selle lisamiseks kasutas töö autor rakenduses Entity Frameworki teeki, mis on objektide-relatsiooniline kaardistaja. Selle kasutamine aitab tunduvalt vähendada rakenduse ja andmebaasi vahelist sõltuvust ning muudab andme suhtluse tunduvalt lihtsamaks [18], kuna Entity Framework võimaldab kasutada andmebaasi päringute jaoks LINQ tehnoloogiat, mille abil saab päringuid puhta SQLi asemel kirjutada C# keeles. Sel viisil on koodil kompileerimise ajal tüübi kontroll olemas ning see aitab vigaseid päringuid kiiremini tuvastada [19]. Samuti aitas see tehnoloogia autoril abstrakteerida erinevaid päringuid väiksemateks päringuteks, mis on oma funktsioonides. Selle abil on võimalik taaskasutada koodi ning jälgida DRY koodi printsiipi.

Kolmandas kihis on veebirakendusliides. See kiht koosneb veebirakenduse seadistustest, autentimise loogikast, kontrolleritest ja vaadetest. Kõik rakenduse kontrollerid on turvatud OpenId Connecti protokolli põhise autentimisega. Kontrollerite eesmärgiks on vastata esisüsteemis tehtud päringutele vaadetega. Päringu puhul valideeritaks sisend ning vea puhul vastatakse kliendile veateavitusega. Kui valideering on edukas siis sisend muudetakse AutoMapperi teegi abil domeeni objektiks ja seejärel saadetakse see infrastruktuuri kihti edasi, milles asub äriloogika. Seejärel rakendatakse äriloogikat ja saadetakse kontrollerile vastus tagasi. Vastus muudetakse kontrolleris uuesti presentatsioonikihi mudeliks ja mudelite väljad lisatakse vaadetesse ning seejärel saadetakse kliendile vastus. Selline lähenemine aitab eraldada äriloogikat presentatsiooni loogikast, mis omakorda aitab taaskasutada sama äriloogikat teiste rakenduste puhul, näiteks kui soovitakse sama rakendust mobiilile, siis peaks peamiselt ainult uue presentatsiooni looma selle jaoks, sest äriloogika ja presentatsioonikiht ei sõltu üksteisest.

Neljandas kihis on loodud unit testid. Need testivad äriloogika spetsiifilisi funktsioone, veebirakenduseliidese kontrollereid ning andmebaasist pärimise loogikat.

Samuti lisas autor ka viienda kihi, mis pole otseselt *domain-driven* mustris kirjeldatud aga rakenduse puhul oli see vajalik. See kiht koosneb ressursifailidest, mille eesmärgiks on hoiustada erinevate keelte tõlkeid, mida veebirakendus peab toetama. Kõik tõlked ja nendele vastavad kultuuritüübid registreeritakse rakenduse käivitamisel ning neid märgistatakse vaatemudelite puhul atribuudina ning vaadete puhul lisatakse tõlge otse vaate sisse Razor süntaksi abil.

3.3.2 .NET raamistik

Tagasüsteemi rakendus toetub .NET Core raamistikule. See raamistik on Microsofti poolt loodud vabatarkvaraline arendusplatvorm ning erinevalt selle eelpärijast, .NET Framework-st, töötab see nii Linuxi, Windowsi kui ka Mac OS operatsioonisüsteemide peal. Selle abil on arendajatel võimalik erinevaid veebi-, mobiili- ja pilverakendusi arendada [20].

Käesolevas süsteemis valiti .NET Core 6 raamistiku versioon, see valiti kuna arenduse hetkel oli see kõige uuem LTS versioon. Selle versiooni puhul tagab Microsoft pikaajalise uuenduste toetuse. See tähendab, et kui leitakse raamistikus mingi koodidefekt, siis Microsoft plaanib selle ära parandada. Seda raamistiku versiooni toetatakse kuni 2024 novembrini [21].

3.3.3 ASP.NET Core raamistik

Süsteemi veebirakenduse kiht on kirjutatud ASP.NET Core 6 raamistikule tuginedes. See raamistik koosneb erinevatest teekidest, mille abil on võimalik luua kiireid ja skaleeruvaid veebirakendusi. ASP.NET Core raamistik tugineb .NET Core raamistikul.

3.3.4 Delegeeritud autentiseerimise lisamine

Uue rakenduse nõudeks oli, et rakendusele pääseksid ligi ainult ettevõtte Active Directory spetsiifilises grupis olevad kasutajad. Selle jaoks lisas autor OpenID Connecti protokollide põhise sisselogimissüsteemi, mille puhul delegeeritakse autentimine Azure Active Directory identiteedi platvormi poole. Selle abil on võimalik kontrollida, et kas rakenduse kasutaja kuulub ettevõtte Active Directory instantsi [22].

Esialgu navigeerib kasutaja rakenduse veebilehele. Seejärel kontrollib veebirakenduse vahevara, et kas kasutaja on autenditud, kui kasutaja ei ole hetkel autenditud, suunatakse

ta Azure Active Directory sisselogimise lehele. Kui kasutaja autendib ennast ning annab nõusoleku rakenduses kasutavate privaatsuspoliitika nõuetele, et rakendus võib kasutaja andmeid küsida ja neid oma süsteemis kasutada, suunatakse kasutaja edasi veebirakenduses määratud sisselogimise võrguaadressile koos identifitseerimis-, ligipääsu- ja värskendustõendiga. Identifitseerimistõend hoiustab endas informatsiooni kasutaja kohta. Ligipääsutõendiga on võimalik küsida Active Directory organisatsiooni instantsi kohta ressursse nii enda kui ka teiste kasutajate ja gruppide kohta. Värskendustõendiga on võimalik uusi tõendeid küsida. Seejärel valideeritakse tõendid ja kui need on valiidsed siis need krüpteeritakse ja lisatakse kasutaja veebibrauserisse küpsistena [22].

Seejärel tuleb kontrollida, et kas kasutaja on spetsiifilises Active Directory grupis. Selle jaoks tehakse andmebaasi päring ning küsitakse kõiki rakenduse administraatorite poolt seadistatud Active Directory grupid. Nende gruppidega tehakse eraldi päring Microsoft Graph rakendusliidesesse, millest küsitakse Active Directory gruppide ja kasutajate kohta informatsiooni. Päringuga kontrollitakse, et kas kasutaja on eelnevalt seadistatud gruppides. Kui kasutaja on nendes gruppides siis suunatakse ta veebirakenduse lehele, kus tal on ligipääs andmetele. Juhul kui kasutaja on ettevõtte Active Directory liige aga ta pole rakenduses seadistatud Active Directory gruppides siis talle näidatakse vealehte, milles kirjeldatakse, et tal puuduvad õigused, et ressursile ligi pääseda. Juhul kui kasutaja on rakenduses seadistatud administraatoriks siis eraldi gruppidesse kuuluvust ei kontrollita ja antakse koheselt ligipääs.

Selle lahenduse puhul tekkis esialgu suureks probleemiks tõendite uuendamine, sest kui esialgu küsitakse tõendeid, siis nende kehtivus on ainult 1 tund ja 30 minutit ning pärast seda ei saa enam pilvesüsteemist nende tõenditega ressursse küsida, sest nad on aegunud [23]. Selleks pidi autor igal süsteemi päringul kontrollima küpsises olevaid tõendeid, et kindlaks määrata, et kas tõend on aktiivne veel või mitte. Kui tõend ei ole enam aktiivne siis küsitakse värskendustõendiga uued tõendid ning vanad tõendid asendatakse uutega.

3.3.5 Graph API-st ressursside pärimine

Rakenduses kasutajate ja gruppide kohta päritakse andmeid Micorosft Active Directory-st. See teostatakse Graph API abil, mis põhineb REST arhitektuuril. See võimaldab küsida andmeid Microsofti poolt pakutavate pilveteenuste kohta. Selleks et rakendus

saaks andmeid küsida, pidi autor registreerima rakenduse ettevõtte Active Directory keskkonnas ning lisama ettevõtte organisatsiooni instantsi andmed rakendusse [24].

Päringute tegemiseks kasutati Microsofti Graph klienti, millesse lisatakse rakenduse kasutaja ligipääsutõend. Selle kasutamiseks lisas autor rakendusele Microsofti Graph tarkvaraarenduskomplekti. Graph kliendi abil oli autoril võimalik lihtsalt ja mugavalt päringuid teha ning samuti on nii päringud kui ka päringute vastused tugevalt tüübitud ehk ei pea päringute ega vastuste jaoks eraldi mudeleid looma, sest need on juba defineeritud tarkvaraarenduskomplektis [25].

3.3.6 Pidevvalmiduse ja pidev integratsiooni lisamine

Uue rakenduse puhul oli tähtis, et sellel oleks pidevvalmidus ja pidev integratsioon. Need aitavad omakorda produktiivsust tõsta ja testida, et uued muudatused koodi äriloogikat katki ei teeks ning võimaldab kiiremini ning automaatselt uusi versioone väljastata [26].

Pideva integratsiooni puhul on loodud koodivaramusse skript, mis käivitatakse iga kord kui koodivaramu harult lisatakse uus kood peaharusse. Selle puhul kontrollib skript automaatselt koodivaramu virtuaalmasinas, et kas kood kompileerib ja käivitatakse kõik rakenduse testid. Kui testid õnnestuvad pakitakse kood artefaktiks (*artifact*), mis on kogum koodi binaarfailidest. Seda on võimalik uue versiooni väljastamisel kasutada teistes skriptides.

Pidevvalmiduse korral, kui soovitakse uut versiooni turule väljastada, tuleb valida koodivaramus vastav keskkond ning manuaalselt tööle panna skript, mis võtab viimati loodud artefakti ning esialgu lisab skript automaatselt selle artefakti konfiguratsiooni failidesse erinevad salasõnad ja vajalikud parameetrid. Keskkondadele vastavad parameetrid, mis ei ole sensitiivsed, lisatakse otse koodivaramu skripti muutujate hulgast. Ülejäänud saladused, mis ei tohi avalikusele lekkida lisatakse läbi Azure Key Vaulti. See on Microsofti poolt loodud pilveteenus, mis lihtsalt integreerub DevOpsi koodivaramu keskkonnaga ning selle abil on võimalik lisada skriptidesse erinevaid saladusi, nii et need ei oleks nähtavad kasutajate poolt, kellel ei ole Azure Key Vaulti ressursile ligipääsu [27].

Kui vajalikud parameetrid on lisatud siis ehitatakse rakenduse dockeri faili põhjal artefaktist dockeri pilt. Loodud pilt saadetakse edasi ACRi, mis on Microsofti poolt pakutav pilveteenus, milles on võimalik erinevaid dockeri pilte ja konteinereid hoiustada

ning hallata. ACR-st omakorda saadetakse uue dockeri pildi ilmumisel veebihaagi abil teavitus Azure rakendusesse, et on ilmunud rakendusele uus versioon ja vana tuleb selle vastu ära vahetada. Seejärel tõmmatakse automaatselt rakenduse hoidlas uus versioon alla ja pannakse see üles.

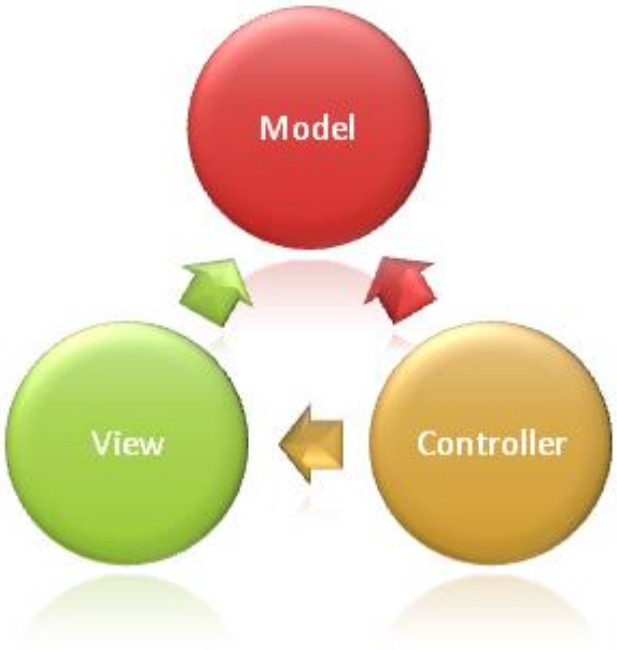
3.4 Esirakenduse arendus

Rakenduse presentatsioonikiht on loodud ASP.NET Core MVC raamistiku abil, mis järgib MVC arhitektuurilist disainimustrit, mis koosneb kolmest osast (Joonis 3).

Esimeseks osaks on presentatsiooni mudelid. Uue rakenduse puhul siis näiteks, projekti, arvete või klientide mudelid.

Teiseks osaks on vaade, mis on kasutajaliides ehk see mida kasutaja lehele minnes näeb. Vaadetesse lisatakse presentatsiooni mudeli väljad. Vaated on HTML mallid, millel on sisseehitatud Razori süntaks. Selle abil on võimalik presentatsiooni loogikat kirjutada C# keeles.

Kolmandaks osaks on kontroller, mis vastab kliendi päringutele. Päringute põhjal genereeritakse serveri poolel vaade ning see väljastatakse kasutajate.



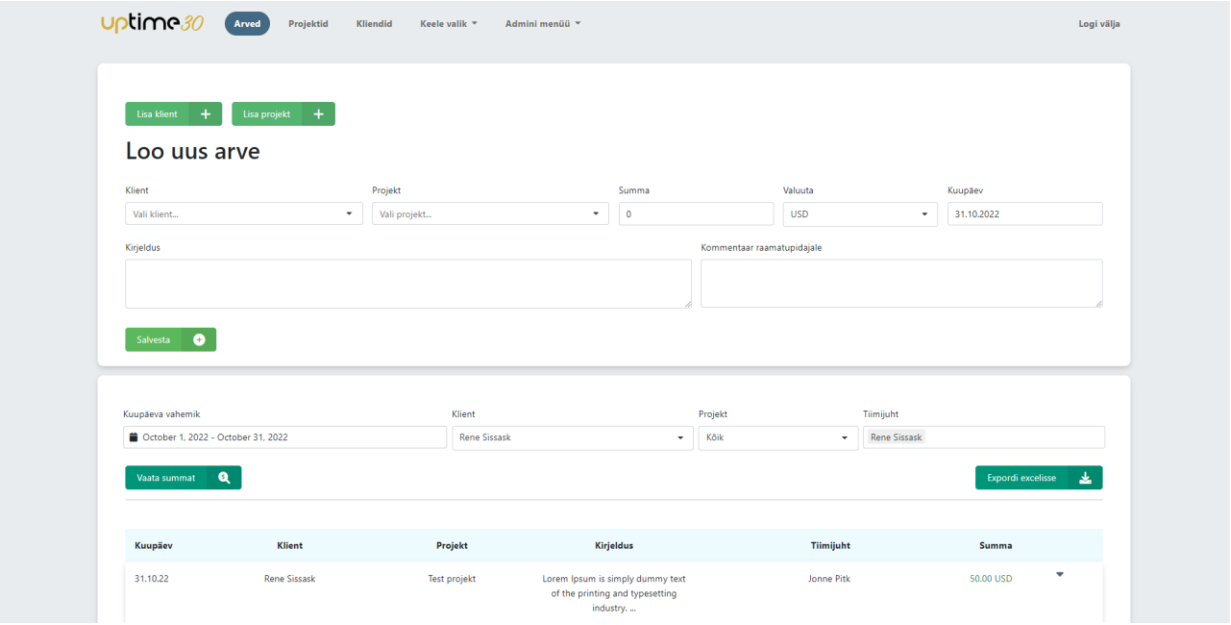
Joonis 3. Mudel-vaade-kontroller muster [28].

Kasutajaliides on loodud HTMLi, CSSi ja JavaScripti abil. Selleks et veebilehed oleksid reageeriva disainiga kasutas autor Bootstrapi 5 raamistikku. See tööriistakomplekt koosneb erinevatest valmistehtud komponentidest, CSSi klassidest ja ka Javascripti funktsioonidest. Nende abil sai autor lihtsalt ja kiirelt uue rakenduse vaated luua.

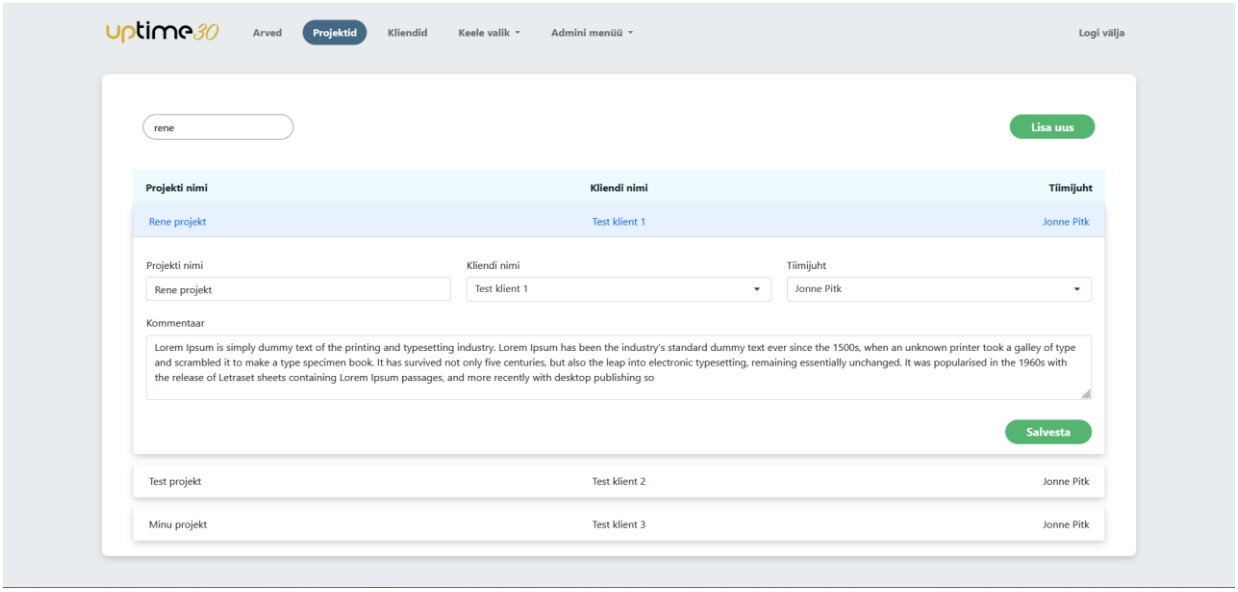
Selleks et veebilehed oleksid dünaamilised, kasutas autor Javascripti jQuery teeki. Sellega saab kiirelt ja mugavalt HTMLi dokumendi elemente manipuleerida [29]. Samuti saab sellega kergelt asünkroonseid HTTP päringuid teha ajaxi tehnikate abil. Selle jaoks kasutati jQuery ajax meetodit [30]. Seda kasutas autor arvete filtreerimiseks ehk kui muudetakse arvete filtreid, tehakse päring serverisse ja vana arvete sektsioon vahetatakse serverist tulnud uue sisu vastu. Samuti kasutas autor ajaxi tehnikaid vormide puhul. Kui klientide, projektide või arvete vorm täidetakse ära, saadetakse selle sisu serverisse ja selle vastus renderdatakse olemasolevasse loendisse, ilma et peaks veebilehte värskendama. See muudab kasutajale veebilehe kasutamise kordades kiiremaks ja mugavamaks. Samuti kasutas autor väiksemal määral esisüsteemi loomisel järgnevaid teeke:

- *Moment* – Selle abil sai autor kergemini valideerida ja muuta rakenduses näidatavate kuupäevade formaati.
- *Selectize* – Selle abil lisas autor rakendusse vormidele valiku väljad (*drop-down list*). Samuti sai sellega tekstipõhiselt filtreerida valikute loendit ning toetas mitme valiku valimist.
- *Date range picker* – Autor kasutas seda, et luua esisüsteemis kalendreid, mille põhjal filtreeritakse arvete loendi tulemusi.

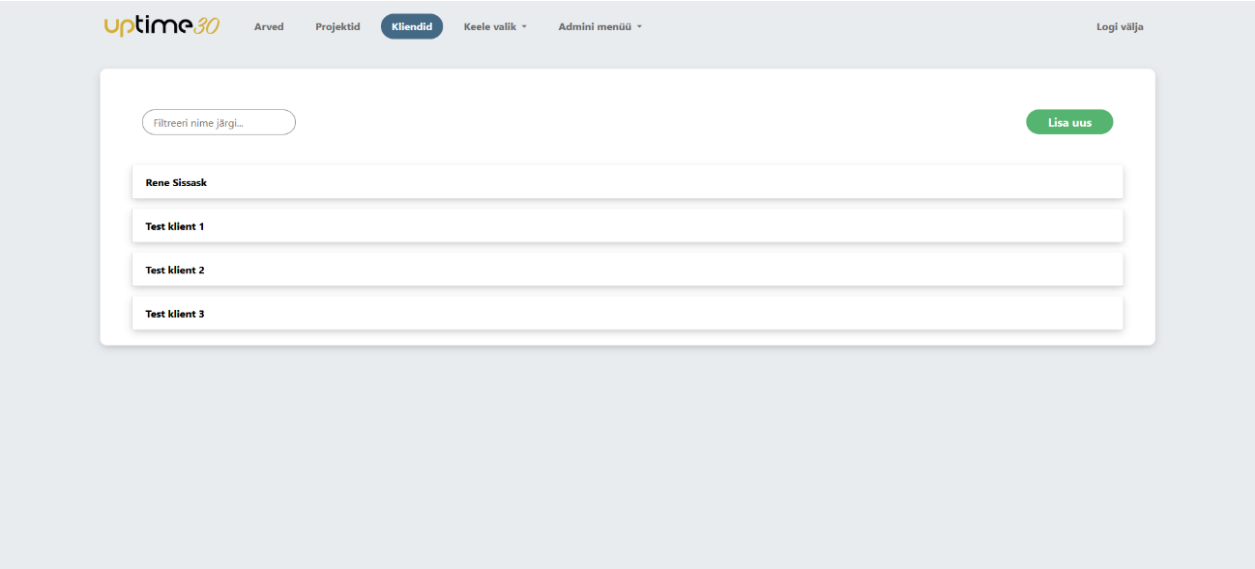
Selleks et esisüsteemis loodud komponente taaskasutada, koondas autor kõik korduvad esisüsteemi osad poolikuteks vaadeteks (*partial view*). See on razori märgistusfail, mida saab taaskasutada teistes peavaadete märgistusfailides. Poolikud vaated aitavad tunduvalt duplikatsiooni vähendada ning samuti koondades suuremad märgistusfailid väiksemateks komponentideks, muudab see koodi hallatavuse ja lugevuse paremaks [31]. Joonistel 4-6 on välja toodud uue rakenduse arvete, projektide ja klientide vaated.



Joonis 4. Uue rakenduse arvete lehe vaade.



Joonis 5. Uue rakenduse projektide lehe vaade.



Joonis 6. Uue rakenduse klientide lehe vaade.

3.5 Unit testid

Erinevalt pärandsüsteemist on uues rakenduses loodud palju unit teste. Nende abil on võimalik tagada, et uute muudatuste lisamisel põhi äriloogika funktsioonid töötaksid õigesti ja et oleks võimalikult varakult võimalik defekte leida. Samuti aitavad need kaasa kõrge koodi kvaliteedi hoidmisel, sest kui teste on raske luua siis see on indikaator sellele, et testitav funktsioon on liiga kompleksne ning selle peaks väiksemateks funktsioonideks refaktoreerima.

Testidega on kaetud peamiselt kogu äriloogika, andmete pärimisega seotud loogika ja väljastamisega seotud meetodid ehk kontrollerid. Rakenduse testid on loodud tuginedes xUniti raamistikule. Autor valis just selle raamistiku sest selle abil luuakse iga testi puhul testklassist uus instants. See tähendab, et iga testi käivitamisel saab taaskasutada konstruktoris olevat koodi ja vältida duplikatsiooni teket [32].

3.6 Logimine

Selleks et ettevõttel oleks rakenduse käitumise kohta kindel ülevaade, lisas autor rakendusele logimise võimaluse. Rakenduses logitakse kõik sisse- ja väljaminevad päringud, erindeid ja turvalisusega seotuid tegevusi, näiteks kui keegi üritab uues süsteemis ennast autentida kasutajaga, kes ei ole rakenduses seadistatud gruppides.

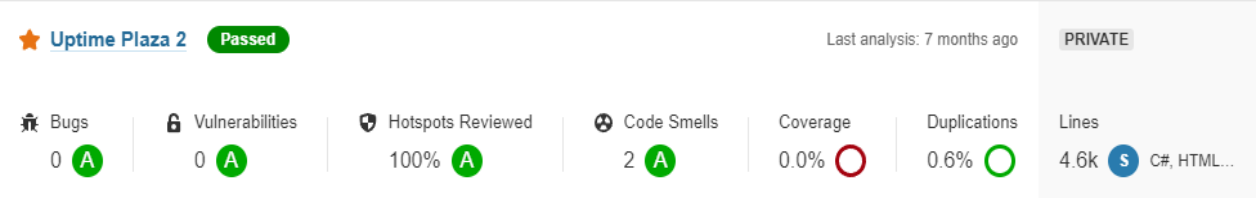
Logid hoitakse struktureeritud kujul ning neid hoitakse andmebaasis. See logimise viis valiti kuna sellega on kõikidel logidel kindel struktuur, mis muudab spetsiifiliste logide otsimise kergemaks ja kiiremaks. Samuti saab selle abil lisada logidesse täiendavaid välju, näiteks milline kasutaja päringu tegi. Need muudavad erinevate vigade otsimise ja silumise lihtsamaks [33].

Logimise funktsionaalsuse lisamiseks kasutas autor Serilog teeki, sest see toetab struktureeritud kujul logimist.

3.7 SonarQube

Selleks et veenduda kõrges koodi kvaliteedis ja rakenduse turvalisuses, kasutas autor SonarQube tööriista, mille abil on võimalik staatiliselt analüüsida koodibaase. Analüüsi käigus kontrollitakse, et kas koodibaas vastab üldistele koodi kirjutamistavadele ning turvalisuse puhul otsitakse, et kas süsteemil on OWASPi nimekirjas välja toodud turvaohte [34]. Samuti annab analüüs ülevaate ka koodikorduste kohta.

Uue rakenduse analüüs läks edukalt ja selle tulemusel leiti koodiread, milles esinevad ainult koodilõhnad (*code smells*) ja koodikordused (Joonis 7). Enamus neist sai autor ära refaktoreerida, mis omakorda muudab koodi hallatavuse ja lugevuse paremaks.



Joonis 7. Uue rakenduse koodibaasi SonarQube analüüsi tulemus.

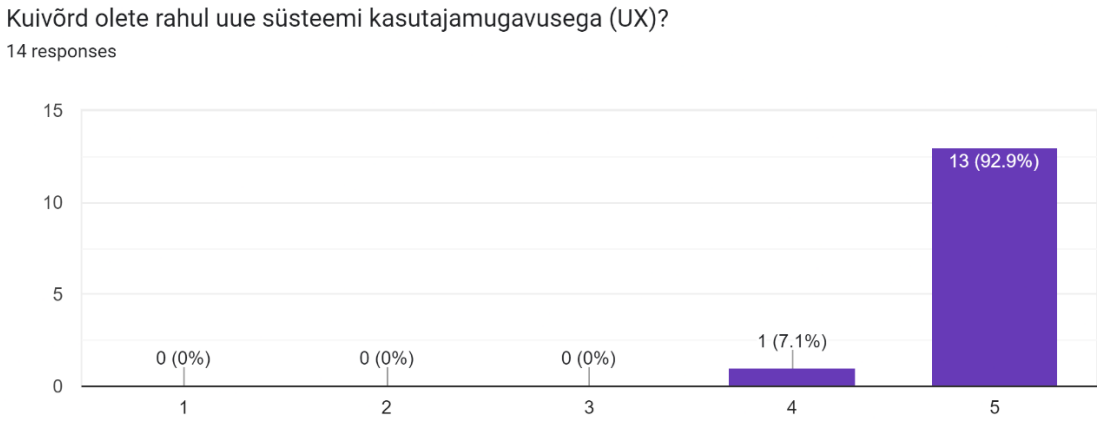
4 Tulemused

Järgnev peatükk annab ülevaate uue süsteemi tulemustest. Samuti kirjeldatakse projekti tuleviku plaane.

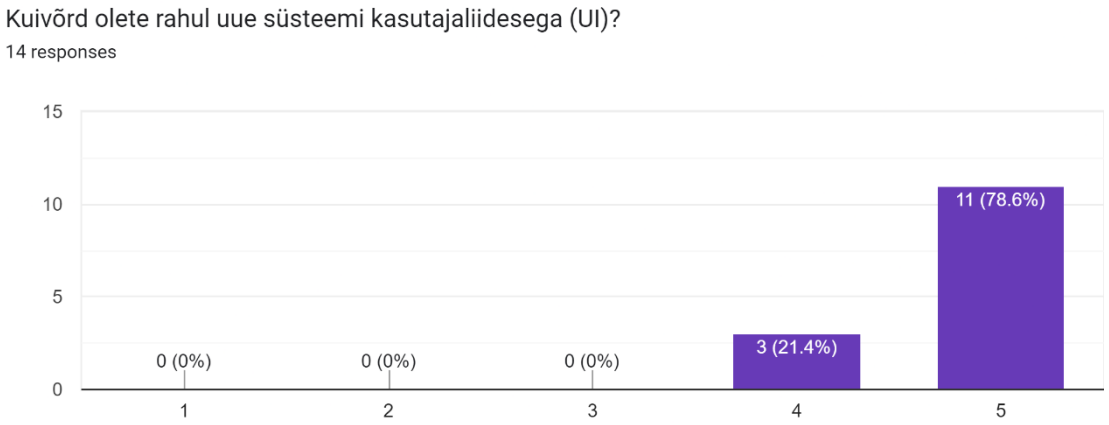
4.1 Lahenduse tulemused

Lõputöö käigus sai ettevõttes välja vahetatud vana arveldussüsteem ning võeti kasutusele uus arveldussüsteemi MVP versioon. Selleks et paremini analüüsida uue veebirakendusi tulemusi, viis autor läbi uue süsteemi rahulolu veebiküsitluse (Lisa 2). Selle eesmärgiks oli võrrelda uut süsteemi vana süsteemiga ning teada saada, et kuivõrd rahul on kasutajad uue süsteemiga. Küsitluse sihtgrupiks olid projekti- ja tiimijuhid, kes olid uut rakendust vähemalt 5 kuud kasutanud. Veebiküsitluses osales 14 uue süsteemi kasutajat. Küsitlus oli anonüümne ja see viidi läbi Google Forms keskkonnas.

Esialgu küsiti kasutajate käest, et kuivõrd rahul on nad uue süsteemi kasutajamugavusega ning kasutajaliidesega. Selle küsitluse tulemustest võib järeldada, et peamiselt kõik kasutajad on väga rahul uue süsteemi kasutajamugavusega kui ka kasutajaliidesega (Joonis 8-9). See tuleneb sellest, et uue esisüsteemi loomisel pööras autor palju tähelepanu sellele, et veebilehed oleksid dünaamilised, reageerivad ning näeksid head välja.

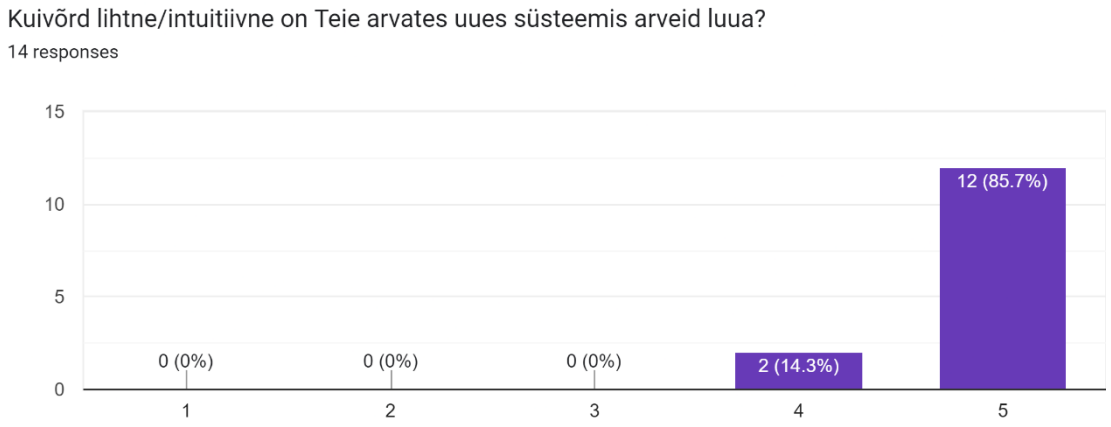


Joonis 8. Küsitluse tulemus, mis näitab kuivõrd on uue süsteemi kasutajad rahul selle kasutajamugavusega.



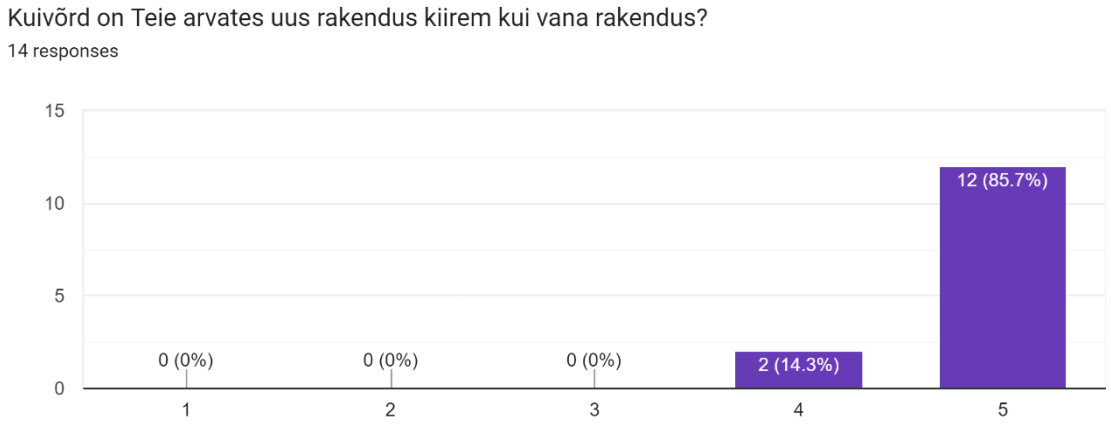
Joonis 9. Küsitluse tulemus, mis näitab, et kuivõrd rahul on uue süsteemi kasutajad selle kasutajaliidesega.

Kuna pärandsüsteemis oli kasutajate sõnul väga keeruline ja tülikas uusi arveid luua, keskendus autor uue süsteemi loomisel, et selle kasutamine oleks kasutajale võimalikult lihtne. Kolmanda küsimuse tulemusest on näha, et küsitluse vastajate arvates on uues süsteemis väga lihtne kasutajatel uusi arveid luua (Joonis 10).



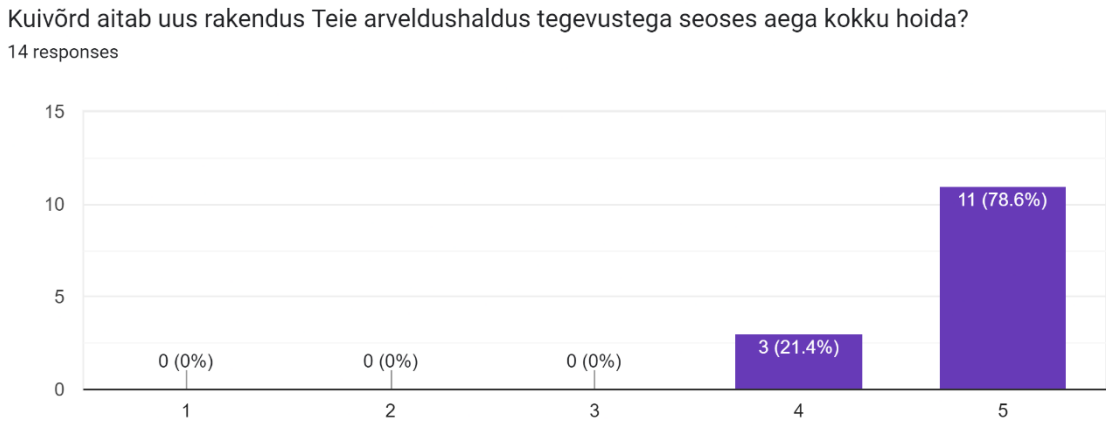
Joonis 10. Küsitluse tulemus, mis näitab, et kui lihtne on kasutajate arvates uues süsteemis arveid luua.

Seejärel küsiti, et kuivõrd on uus rakendus kiirem kui vana rakendus. Sellest tulenes, et paljude vastajate arvates on uus rakendus palju kiirem kui vana (Joonis 11). Selle peamiseks põhjuseks on see, et uue rakenduse päringud on paremini optimeeritud ja osasid klasside instantse hoitakse vahemälus.



Joonis 11. Küsitluse tulemus, mis näitab, et kui kiire on kasutajate arvates uus süsteem võrreldes vana süsteemiga.

Selleks, et täpsemalt kaardistada uue rakenduse ärilised tulemused, küsiti kasutajate käest, et kuivõrd rohkem aitab uus rakendus arveldushaldus tegevustega aega kokku hoida. Tulemustest on näha, et enamus vastajate hinnangul aitab uus süsteem aitab neil aega palju rohkem kokku hoida (Joonis 12).



Joonis 12. Küsitluse tulemus, mis näitab, et kui palju aitab uus rakendus arveldushaldus tegevustega seoses aega kokku hoida.

Samuti küsiti tagasiside jaoks kasutajatelt, et kas uuel rakendusel on puuduseid. Sellest tulenes ainult 2 puudust. Esiteks võiks uue arve lisamisel emaili teavituses olla lisaks arve detailidele ka konkreetne arve link, millele vajutamisel avatakse lisatud või muudetud arve vaade. Teiseks võiks olla võimalik projektide vaates lisada projektidele baasarve, mille valimisel arvete vaates täidetakse automaatselt kõik arve projektiga seotud vormi lüngad. See tuleneb peamiselt sellest, et enamus klientide puhul tekkivad identsed arved ning ainult arvesumma on erinev. See võimalaks rakenduse kasutajatel kiiremini uusi arveid luua.

Küsitluse tulemustest saab järeldada, et uus süsteem on parem kui pärandsüsteem. Samuti on see rohkem äriliselt kasumlik ettevõttele, kuna selle kasutajad säästavad rohkem aega ning saavad tegeleda prioriteetsemate ülesannetega.

Samuti on ettevõttel uue rakenduse puhul selle hooldamine ja uute muudatuste lisamine kergem, kiirem ja soodsam. See tuleneb sellest, et uus rakendus on põhjalikult dokumenteeritud ja kirjutatud programmeerimiskeeles mille jaoks on ettevõttel ressursse olemas.

4.2 Projekti tulevik

Autor kavatseb uut rakendust edasi arendada. Esialgu on plaanis lisada arvete vaatesse graafikud. Nende abil saaks visuaalselt näha ettevõtte projekti- või tiimijuhi poolt sisse

toodud kasumit. Samuti on plaanis lisada rakendusse kõik küsitlusest kogutud soovitused. Seejärel optimeeritakse rakenduse andmebaasi päringuid, lisades võimalusel vajalikke indekseid andmebaasi tabelitele. Samuti optimeeritakse kontrollerite päringuid, lisades vaated vahemällu, et vähendada andmebaasi päringute kogust.

Järgnevalt on kavas võtta uus rakendus kasutusele ka teistes ettevõtte harukontorites, mis asuvad välismaal. Selle jaoks tuleb lisada rakendusele keeletõlkeid juurde ning iga harukontori jaoks peab oma instantsi rakendusest looma ning vastavalt nende vajadustele selle ära seadistama. Samuti tuleb iga ettevõtte harukontori jaoks koodivaramus eraldi haru luua. See võimaldaks lisada vastavalt ettevõtte harukontorite vajadustele unikaalseid muudatusi nii et uued muudatused ei mõjutaks teiste harukontoritele seadistatud rakendust.

5 Kokkuvõte

Uptime OÜ ettevõttele on äriliselt tahtis, et neil oleks firmasisene süsteem, mis võimaldaks teatud töötajatel firma sisse- ja väljaminekuid hallata. Ettevõttes kasutusel olev pärandsüsteem selleks ei sobinud, kuna ettevõttel polnud vabu ressursse, et seda edasi arendada ning selles esineb palju vigu.

Lõputöö raames analüüsiti põhjalikult pärandsüsteemi, ning toodi välja selle peamised puudused. Samuti uuriti ka olemasolevaid sarnaseid süsteeme. Nendest ei leidnud autor ühtegi lahendust, mis kataksid ettevõtte vajadusi. Seejärel loodi püstitatud nõuete kohaselt uus rakendus ning selle valmimisel võeti see ettevõttes kasutusele.

Uue süsteemi tulemuste kaardistamiseks viis töö autor uue rakenduste kasutajate seas läbi veebiküsitluse. Sellest tulenes, et rakenduse kasutajad on uue süsteemiga väga rahul ning samuti aitab uus rakendus ettevõtte töötajatel tunduvalt aega kokku hoida arvete haldamise tegevustega seoses, seega on rakendus ettevõttele äriliselt kasumlik. Samuti sai autor küsitlusest head tagasisidet, et mida võiks tulevikus rakendusele juurde lisada.

Nii ettevõte kui ka autor plaanivad loodud projektiga edasi tegeleda. Esialgu on plaanis lisada kõik soovitused, mis koguti küsitluse käigus. Seejärel on kavas optimeerida rakenduses tehtavaid päringuid, et need oleksid veelgi kiiremad. Lõpuks paigaldatakse rakendus teistesse ettevõtte kontoriharudesse.

Kasutatud kirjandus

[1] Uptime OÜ, [Võrgumaterjal]. Saadaval: https://www.uptime.ee/. [Kasutatud 20.10.2022].

[2] Uptime OÜ, „Uptime pälvis 16. aastat järjest Microsofti Kuldpartneri staatuse," [Võrgumaterjal]. Saadaval: https://www.uptime.ee/uptime-palvis-16-aastat-jarjest-microsofti-kuldpartneri-staatuse/. [Kasutatud 20.12.2022].

[3] K. Koishigawa, „A Brief History of Responsive Web Design," [Võrgumaterjal]. Saadaval: https://www.freecodecamp.org/news/a-brief-history-of-responsive-web-design/. [Kasutatud 25.11.2022].

[4] R. C. Martin, Clean Code: A Handbook of Agile Software Craftmanship, Boston: Pearson, 2008.

[5] P. Gorbachenko, „What are Functional and Non-Functional Requirements and How to Document These," [Võrgumaterjal]. Saadaval: https://enkonix.com/blog/functional-requirements-vs-non-functional/. [Kasutatud 15.11.2022].

[6] I. I. o. B. Analysis, Babok v3 A Guide to the Business Analysis Body of Knowledge, Toronto: International Institute of Business Analysis, 2015.

[7] J. Shmygaleva, „How to Automate Invoicing Process for Estonian Company: Invoice Solution on Enty," [Võrgumaterjal]. Saadaval: https://enty.io/blog/invoicing-on-enty. [Kasutatud 15.10.2022].

[8] Enty, „Pricing," [Võrgumaterjal]. Saadaval: https://enty.io/pricing. [Kasutatud 16.10. 2022].

[9] Odoo, „Accounting and Invoicing," [Võrgumaterjal]. Saadaval: https://www.odoo.com/documentation/16.0/applications/finance/accounting.html. [Kasutatud 15.11.2022].

[10] Odoo, „Building a Module," [Võrgumaterjal]. Saadaval: https://www.odoo.com/documentation/16.0/developer/howtos/backend.html. [Kasutatud 10.12.2022].

[11] S. W. Ambler, „Intoduction to Test Driven Development (TDD)," [Võrgumaterjal]. Saadaval: http://agiledata.org/essays/tdd.html#WhatIsTDD. [Kasutatud 19.11.2022].

[12] Microsoft Corporation, „What is Azure SQL Database?," [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql. [Kasutatud 10.11.2022].

[13] Microsoft Corporation, „DTU-based purchasing model overview," [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/azure/azure-sql/database/service-tiers-dtu?view=azuresql#database-transaction-units-dtus. [Kasutatud 17.10.2022].

[14] Microsoft Corporation, „Azure SQL Database and Azure Synapse IP firewall rules,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/azure/azure-sql/database/firewall-configure?view=azuresql. [Kasutatud 18.10.2022].

[15] Microsoft Corporation, „Code First Migrations,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/. [Kasutatud 17 10 2022].

[16] S. Smith, Architecting Modern Web Applications With ASP.NET Core And Microsoft Azure, Redmond: Microsoft Corporation, 2022.

[17] L. Nguyen, „Domain-Driven Design in ASP.NET Core applications,“ [Võrgumaterjal]. Saadaval: https://enlabsoftware.com/development/domain-driven-design-in-asp-net-core-applications.html. [Kasutatud 18.10.2022].

[18] Microsoft Corporation, „Entity Framework Core,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/ef/core/. [Kasutatud 20.10.2022].

[19] Microsoft Corporation, „Language Integrated Query (LINQ) (C#),“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/. [Kasutatud 14.11.2022].

[20] Microsoft Corporation, „What is .NET? Introduction and overview,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/dotnet/core/introduction. [Kasutatud 19.10. 2022].

[21] Microsoft Corporation, „.NET and .NET Core Support Policy,“ [Võrgumaterjal]. Saadaval: https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core. [Kasutatud 19.10.2022].

[22] Microsoft Corporation, „OpenID Connect on the Microsoft identity platform,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-protocols-oidc. [Kasutatud 18.10.2022].

[23] Microsoft Corporation, „Microsoft identity platform access tokens,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/azure/active-directory/develop/access-tokens#access-token-lifetime. [Kasutatud 29.11.2022].

[24] Microsoft Corporation, „Use the Microsoft Graph API,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/graph/use-the-api. [Kasutatud 10.11.2022].

[25] Microsoft Corporation, „Microsoft Graph SDK overview,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/graph/sdks/sdks-overview. [Kasutatud 18.12.2022].

[26] R. Mohanan, „What Is CI/CD? Definition, Process, Benefits, and Best Practices for 2022,“ [Võrgumaterjal]. Saadaval: https://www.spiceworks.com/tech/devops/articles/what-is-ci-cd/. [Kasutatud 10.11.2022].

[27] Microsoft Corporation, „Use Azure Key Vault secrets in Azure Pipelines,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/azure-key-vault?view=azure-devops&tabs=yaml. [Kasutatud 10.12.2022].

[28] Microsoft Corporation, „Overview of ASP.NET Core MVC,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-

us/aspnet/core/mvc/overview/_static/mvc.png?view=aspnetcore-7.0. [Kasutatud 19.10. 2022].

[29] O. Foundation, „What is jQuery,“ [Võrgumaterjal]. Saadaval: https://jquery.com/. [Kasutatud 19.10.2022].

[30] O. Foundation, „jQuery.ajax(),“ [Võrgumaterjal]. Saadaval: https://api.jquery.com/jquery.ajax/. [Kasutatud 20.10.2022].

[31] Microsoft Corporation, „Partial views in ASP.NET Core,“ [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/partial?view=aspnetcore-7.0. [Kasutatud 10.11.2022].

[32] .NET Foundation, „Shared Context between Tests,“ [Võrgumaterjal]. Saadaval: https://xunit.net/docs/shared-context. [Kasutatud 16.10.2022].

[33] T. Hombergs, „Saving Time with Structured Logging,“ [Võrgumaterjal]. Saadaval: https://reflectoring.io/structured-logging/. [Kasutatud 20.10.2022].

[34] S. SA, „OWASP Top 10 - We’ve got you covered!,“ [Võrgumaterjal]. Saadaval: https://www.sonarqube.org/features/security/owasp/. [Kasutatud 15.10.2022].

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

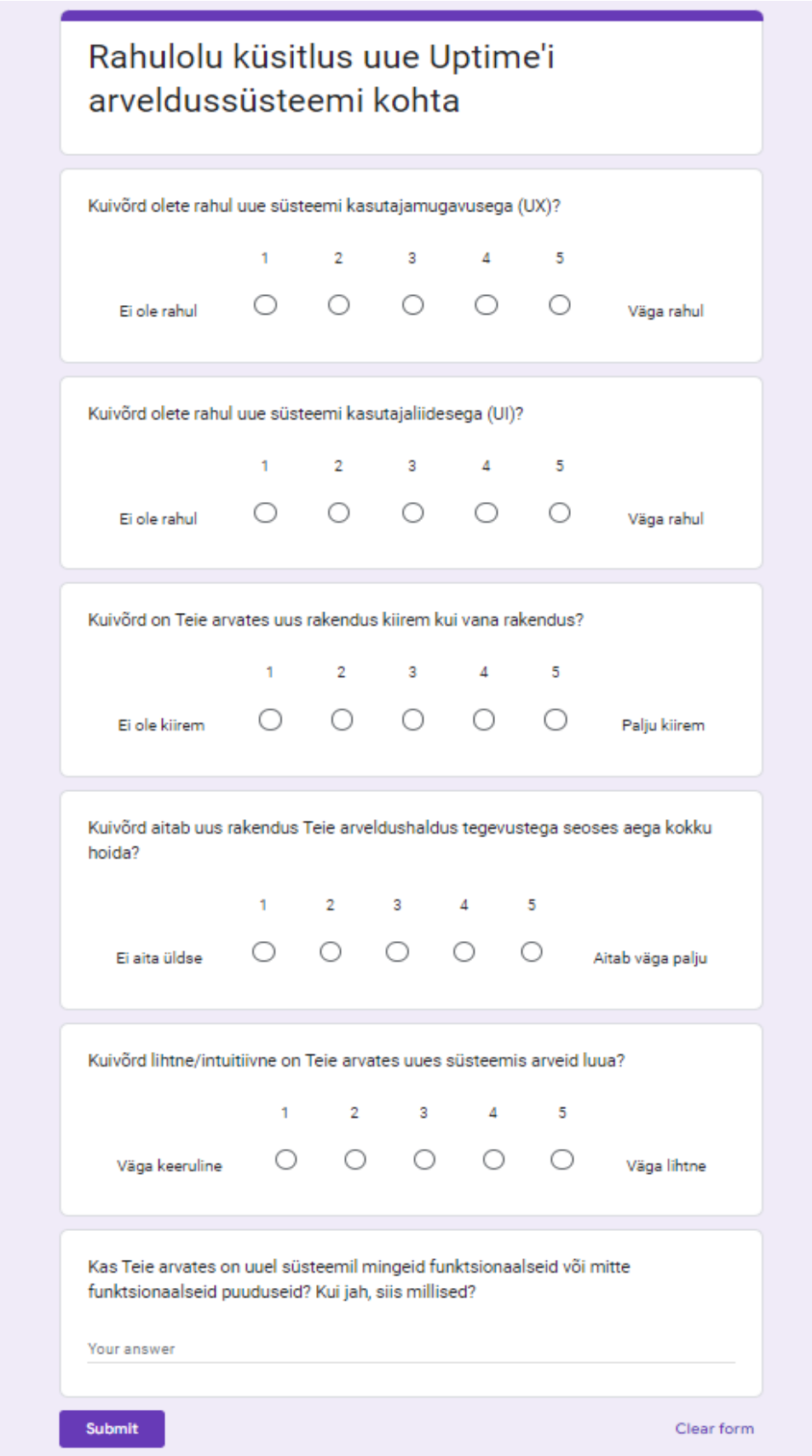
Mina, Enricqe Rene Sissask

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Arveldussüsteemi loomine Uptime OÜ näitel" , mille juhendaja on Einar Kivisalu
   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.

[05.01.2023]

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

Lisa 2 – Uue rakenduse kasutajate vaheline veebiküsitlus



Rahulolu küsitlus uue Uptime'i arveldussüsteemi kohta

Kuivõrd olete rahul uue süsteemi kasutajamugavusega (UX)?

	1	2	3	4	5	
Ei ole rahul	○	○	○	○	○	Väga rahul

Kuivõrd olete rahul uue süsteemi kasutajaliidesega (UI)?

	1	2	3	4	5	
Ei ole rahul	○	○	○	○	○	Väga rahul

Kuivõrd on Teie arvates uus rakendus kiirem kui vana rakendus?

	1	2	3	4	5	
Ei ole kiirem	○	○	○	○	○	Palju kiirem

Kuivõrd aitab uus rakendus Teie arveldushaldus tegevustega seoses aega kokku hoida?

	1	2	3	4	5	
Ei aita üldse	○	○	○	○	○	Aitab väga palju

Kuivõrd lihtne/intuitiivne on Teie arvates uues süsteemis arveid luua?

	1	2	3	4	5	
Väga keeruline	○	○	○	○	○	Väga lihtne

Kas Teie arvates on uuel süsteemil mingeid funktsionaalseid või mitte funktsionaalseid puuduseid? Kui jah, siis millised?

Your answer

Submit Clear form