TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Priidu Melnik 185728IADB

VELOMARKET OÜ PARANDUS- JA HOOLDUSTÖÖDE HALDUSRAKENDUSE LOOMINE

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magister

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: Priidu Melnik

29.04.2021

Annotatsioon

Käesoleva töö eesmärgiks on luua lahendus veebirakenduse kujul firmale Velomarket OÜ. Rakendus lahendab parandus- ja hooldustööde haldamise probleemi, olemasolevat tööprotsessi arvestades.

Luuakse rakendus, mis võimaldab kasutajatel kiirelt luua uusi töökäske, hallata ja muuta töökäskudega seotud informatsiooni ning koostada töötasude arvestamist. Tulemuseks on kiirem ja efektiivsem viis tööde haldamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 23 leheküljel, 4 peatükki, 13 joonist.

Abstract

The objective of this project is to create web application for managing repair and maintenance jobs of Velomarket OÜ. Application solves the problem of managing repair ja maintenance jobs considering existing work processes.

An application will be created which enables users to quickly create new job orders, manage and change data related to job orders and formulate earnings. The result is quicker and more effective way to manage jobs.

The thesis is in estonian and contains 23 pages of text, 4 chapters, 13 figures.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* - rakenduse programmeerimise liides
CSS	*Cascading Style Sheets* - veebilehe stiili muutmise keel
HTML	*Hypertext Markup Language* - veebilehe struktureerimise keel
HTTP	*Hypertext Transfer Protocol* - protokoll andmete edastamiseks interneti vahendusel
JSON	*JavaScript Object Notation*- JavaScripti objekti kujul põhinev andmestandart
JWT	*Json Web Token* – standart JSON formaadis ligipääsu tagamiseks
LINQ	*Language Integrated Query* – päringukeel C# programmeerimiskeeles
NoSQL	*Not only SQL* – mitterelatsioonilised andmebaasid
repair shop management software	töökoja haldustarkvara
repository layer	repositooriumite kiht ehk hoidla kiht
REST	*Representational State Transfer* – Standart, kuidas andmeid edastada
service layer	teenuskiht
SPA	*Single Page Application* - üheleherakendus
SQL	*Structured Query Language* – relatsiooniliste andmebaaside päringukeel
UOW	*Unit Of Work* – kogus operatsioone, mis tehakse ühe transaktsiooniga
Virtual DOM	*Virtual Document Object Model* – virtuaalne dokumendi objektide vorm
XML	*Extensible Markup Language* – laiendatav märgistuse keel

Sisukord

Jooniste loetelu

Sissejuhatus

Käesoleva lõputöö raames luuakse rakendus rattaremondi haldamiseks ettevõttes Velomarket OÜ. Eesmärgiks on luua töötav lahendus, mis võimaldaks töökojas panna kirja tehtud parandused ja hooldused, arvestada mehaanikute palka.

Velomarket OÜ põhiprobleemiks on lahenduse puudumine, mis suurendaks töö kiirust ja efektiivsust. Talvel on klientuur küll väike, kuid kevadel ja suvel muutub arv niivõrd suureks, et on keeruline olemasoleva lahendusega töötada. Kirjutamise hetkel pannakse kogu informatsioon tööde kohta kirja käsitsi mehaanikute poolt Exceli tabelisse. See on ebaefektiivne lahendus rattapoe jaoks, pika andmesisestusprotsessi ja ajakuluka palgaarvestussüsteemi tõttu.

Töös püstitatakse probleem, vastavalt sellele selgitatakse eesmärki. Kirjeldatakse lähteolukorda ja leitakse sobivad töövahendid ja lahenduskäigud probleemi lahendamiseks. Selgitatakse erinõudeid ja Velomarket OÜ soove.

Autor valmistab ette lahenduse kirjelduse vastavalt ettevõte soovidele, pakkudes välja ja analüüsides erinevaid võimalusi. Pärast lahenduse kirjeldust määrab autor ära tehnilise lahenduskäigu ja põhjendab oma valikuid. Tulemusena valmib süsteem, mis lahendab käesoleva probleemi.

1 Probleemi kirjeldus ja uus lahendus

1.1 Detailne kirjeldus

Velomarket OÜ tegeleb jalgrataste, elektritõukerataste ja varuosade müügiga. Samuti tegelevad nad nende paranduse, täiendamise ja hooldusega. Lisatöödega tegelemisel tuleb jälgida, milline ratas kuulub, millisele kliendile, ja kui palju on mehaanikud kindlale rattale töid sooritanud. Nende andmete põhjal arvestatakse, milline tuleb kliendi arve ja kui palju töötajatele palka makstakse.

Selleks on Velomarketil süsteem, mis hetkel on andmesisestuse ja palgaarvestuse osas tülikas. Arvepidamine algab sellest hetkest, kui klient annab oma jalgratta või muu kergliiklusvahendi üle Velomarketile. Kliendi soovide põhjal sisestatakse Microsoft Exceli dokumendi töökäsu malli töökäsu number, kliendi andmed, ratta kirjeldus, probleemide kirjeldus ja nimekiri töödest, mida soovitakse teostada. Tööde nimekiri on juba varasemalt ära kirjeldatud, koos nende töötundide koefitsientidega, mille põhjal toimub mehaanikute palgaarvestus ja arve koostamine. Lisaks on mallis veel väljad selle jaoks, kas töökäsk on garantii juhtum ja kas väljavahetatud osasid soovitakse tagasi saada.

Eelnevast protsessist saadud andmete põhjal koostatakse formaat, mis on sobiv paberile trükkimiseks. Trükitud paber kinnitatakse kliendi liiklusvahendi külge. Mehaanik lisab paberile oma nime ja teostab tööd, vajadusel lisab töid paberile juurde kirjutusvahendiga ja kui mehaanik vahetub lisatakse järgmise mehaaniku nimi. Pärast tööde valmimist ja ratta üleandmist kliendile, võetakse paberdokument ja selle põhjal täidetakse tabelit Google Sheet pilvefaili, kus on kirjas ka kõik eelnevad töökäsud. Hiljem lisatakse paberdokument, mille külge on klambriga kinnitatud töökäsu arve, jooksva kuu töökäskude kuhja. Järgmise kuu alguses arvutab palgaarvestuse eest vastutav töötaja nende põhjal mehaanikute palgad. Tasub veel mainida, et osa klienditeenindaja tasust tuleb müüdud töö maksumuse ja koefitsiendi korrutisest. Klienditeenindaja koefitsient on erinev mehaaniku koefitsiendist.

Antud kirjelduse põhjal saab järeldada järgnevaid probleeme olemasolevas süsteemis:

- Toimub liigne andmesisestus. Kõigepealt täidetakse tabel, mille andmete põhjal täidetakse mall printimiseks. Pärast tööde valmimist sisestatakse paberil olevad andmed uuesti, et lõplikud andmed digitaalsel kujul talletada.

- Tehtud tööde leidmine on keeruline. Juhul, kui klient peaks soovima garantii tööd teha, peab teenindaja otsima paberil olevat dokumenti või otsima Google Drive'i salvestatud faili, et veenduda kliendi soovide õigsuses.

- Töötajate lisamisel töökäsule võib koefitsientide arvestus töötaja kohta segamini minna. Olenevalt sellest, kui arusaadavalt on mehaaniku lisainformatsiooni paberile lisanud võib sõltuda andmete korrektsus.

- Palgaarvestus on aeganõudev ja võivad kergelt tekkida vead tähelepanematuse tõttu. Iga töökäsk tuleb arvestajal ükshaaval üle vaadata. Tuleb kokku liita kokku mehaaniku koefitsientide ja ühe töö maksumuse korrutised ühes töökäsus ning terves kuus. Ühel töökäsul võib olla mitu erinevat töötajat, mis võib suurendada võimalust vea tekkeks tähelepanematusest.

- Mitme töökoja puhul on andmete haldamine keeruline. Velomarketil on lähitulevikus plaanis avada uus pood, millele lisaks tuleks ka töökoda. Andmete haldamine ja kasutamine erinevate töökodade vahel on keeruline, kui tööprotsessis on oluline roll paberdokumendil.

1.2 Olemasolevad lahendused

1.2.1 Repero

Repero on veebipõhine süsteem parandustööde informatsiooni talletamiseks. Rakendust saab kasutada erinevat tüüpi liiklusvahendite parandamisel. Seal on võimalik üles märkida klientide ja inventuuri andmed. Kliendile saab saata telefoni või e-maili teel kirja töö arengutest. On olemas võimalus näha statistikat tehtud töödest [1]. Esimene samm süsteemi tööprotsessis on kliendi andmete sisestamine või tema otsimine. Järgmisena sisestatakse parandatava eseme andmed, mis lisatakse uuele parandustöö vormile. Pärast töö valmimist saab koostada arve, millele lisatakse kasutatud vahendite ja töötundide maksumus. Hiljem saab veel teavitada klienti töö staatusest ja lisada kommentaare ning väiksemaid detaile töö kohta. Saab lisada töötajaid ja hiljem nende palku vaadata [2].

Rakendus on olemuselt ja võimaluste poolest ilmselt kõige ligilähedasem valik Velomarketi probleemi lahendamiseks. Antud lahenduse puuduseks on see, et ei ole võimalik lisada eeldefineeritud tööde nimekirja, koos koefitsientide ja maksumustega. Samuti puudub kindel formaat töökäsu trükkimiseks, mis on hetkel on Velomarketi tööprotsessis oluline. Selle lisaks on veel puudub otsene arvestamine mitme töökojaga.

1.2.2 Shopmoneky

Shopmonkey on parandustööde haldamise ja arveldamise veebirakendus sõiduvahendite ja spetsialiseeritud toodete jaoks. Seal on võimalik eelnevalt koostada tööde malle, saab sisestada inventuuri ja seda paranduste põhiselt hallata, saab vaadata statistikat tehtud töödest ja summadest, kuvab töötasude arvestust. Lisaks on seda rakendust võimalik ühendada kindlate raamatupidamistarkvaradega [3].

Rakenduses põhiliseks miinuseks on see, et otseselt suunatud ainult Ameerika Ühendriikide ja Kanada kasutajatele, mis võib takistada maksmist ja võimalust kasutada lisafunktsioone. Kaasarvutud on rakenduse võimaluste ja funktsioonide valik hetkel liiga suur. Selle kasutuselevõtt eeldaks Velomarketi puhul tööprotsesside muutmist ja suuremat töötajate koolitamist.

1.2.3 Autofutur

Autofutur on haldusrakendus, mis on eelkõige mõeldud motoriseeritud sõiduvahenditega paranduse ja varuosade müügiga tegelevatele ettevõtetele. Võimalik on töökäskude haldus ja laohaldus. Saab hallata kliendi andmeid ja planeerida tööde aegu. Lisaks on lahendused eraldi varuosade tellimiseks. Saab ühendada valitud kindlate raamatupidamistarkvaradega [4].

Sarnaselt Shopmonkey tarkvarale on antud lahenduse valik esialgu liiga suur. Rakenduse kodulehel ei ole välja toodud, mis on toote kasutamise hinnad, mis olenevalt maksumusest võib takistuseks saada. Lisaks ei ole võimalust arvutada tööde müükide tasu.

Eelpool välja toodud lahendused on võimaluste ja funktsionaalsuse poolest kõige sobivamad lahendused, kuid siiski puudustega. Otsides Google’i otsingumootorist märksõna-

dega “töökoja tarkvara” või “*repair shop management software*” on võimalik leida rohkelt erinevaid lahendusi, mis on suunatud autoparandusele ja Velomarketi jaoks töö kirjutamise hetkel liiga suurte lahendustega.

1.3 Uus lahendus

Töö kirjutamise hetkel eksisteerivad lahendused on, kas liiga suured või ei paku sobilikku funktsionaalsust. Velomarketil oleks eelkõige vaja süsteemi, mis arvestaks nende palgaarvestus metoodikaga, muudaks andmetega tegelemise kiiremaks ja vähendaks sõltuvust paberdokumentidest. Neil ei ole hetkel soovi minna uuele süsteemile, mis hõlmaks kogu nende äritegevust. Selline soov tuleneb eelkõige sellest, et neil hetkel on olemas lahendus veebipoe ja inventuuri jaoks ning kassasüsteem koos arvete koostamisega.

Sellest tulenevalt pakub diplomitöö autor lahenduseks välja uue veebirakenduse loomise, mis arvestaks kindlate Velomarketi vajadustega. Rakendus oleks piiritletud töökäskude loomise ja haldamisega ning nende põhjal töötajate lisatasude arvestamisega.

Uus lahendus oleks loodud hajusrakendusena, milles rakenduse erinevad osad lahutatud ja võivad asuda erinevates asukohtades, et mehaanikud ning kliendihaldurid saaksid kasutada seda erinevatest seadetest ja erinevates asukohtades. Täpsemalt kasutatakse klient-server lahendust, mis võimaldab hoida lahus kliendi poolt kasutatava funktsionaalsuse äriloogika ja andmete talletamise [5].

2 Lahenduse analüüs

2.1 Funktsionaalsed nõuded

Teenindaja funktsionaalsed nõudmised:

- Teenindajana soovin, et mul oleks oma kasutaja.
- Teenindajana soovin sisse- ja väljalogimise võimalust.
- Teenindajana soovin võimalust luua uus töökäsk.
- Teenindajana soovin võimalust muuta töökäsu andmeid.
- Teenindajana soovin välja trükkida töökäsu paberi peal.
- Teenindajana soovin töökäskude otsimise võimalust andmete põhiselt.
- Teenindajana soovin tööde otsimis võimalust töökäsu loomisel.
- Teenindajana soovin näha ja muuta enda kasutaja andmeid.
- Teenindajana soovin vajadusel töökäsku kustutada.
- Teenindajana soovin, et töökäsu kustutamisel küsitaks kinnitust kustutamiseks.
- Teenindajana soovin näha eraldi täitmises olevaid töökäske ja neid, mis ei ole.
- Teenindajana soovin näha oma müükide või tehtud tööde pealt teenitud tulu summat valitud ajavahemikus.

Administraatori funktsionaalsed nõudmised:

- Administraatorina soovin luua, hallata ja kustutada kasutajaid.
- Administraatorina soovin luua ja hallata töökodasid.
- Administraatorina soovin näha töötajate teenitud tulude summasid valitud ajavahemikus.
- Administraatorina soovin näha töötajatega seotud töökäske.

- Administraatorina soovin võimalust lisada, peita ridu ja hallata tööde nimekirja.

2.2 Mittefunktsionaalsed nõudmised

Teenindaja mittefunktsionaalsed nõudmised:

- Teenindajana soovin kasutada rakendust telefonis, kui ka arvutis
- Teenindajana soovin kasutada süsteemi eesti keeles.
- Teenindajana soovin kasutada süsteemi brauseris.

2.3 Tehnoloogia valik

Käesoleva töö eesmärgiks on luua hajus veebirakendus. See eeldab sobivate töövahendite valimist. Järgnevalt kirjeldatakse võimalike vahendeid ja põhjendatakse, mida ja miks valiti. Antud töös on vaatlusele võetud tehnoloogiad, mis on suuremate kasutajabaasidega ja mida autor lähemalt tunneb.

Hajus rakenduse realiseerimiseks kasutatakse klient-server mudelit. See on kolmekihiline mudel, kus on andmebaas informatsiooni talletamiseks, server, mis tegeleb äriloogika realiseerimise ja andmebaasist andmete pärimisega ning klientrakendus, mis on graafilise liidesega rakendus, mis suhtleb serveriga [5]. Selleks, et klientrakendus ja server saaksid omavahel suhelda, on vaja kokku leppida, kuidas seda teha. Üks lähenemine selle probleemile on REST (*Representational State Transfer*) API (*Application Programming Interface*) arhitektuur, kus kasutatakse HTTP (*Hypertext Transfer Protocol*) protokolli andmete edastamiseks. Andmete puhul kasutatakse tagastamise, loomise, muutmise ja kustutamise meetodeid. REST APIs on seisuta arhitektuur ehk sõnumi tähendus ei sõltu andmete seisust. Andmete töötlemine toimub andmekujude vahetamise põhjal [6]. JSON (*JavaScript Object Notation*) keel on populaarseim viis kujutada andmeid REST suhtluses [7]. JSON on andmevahetuseks mõeldud kergesti töödeldav keel. Selles kasutatakse võtiväärtus põhist süsteemi [8]. Suure populaarsuse ja rohkete kasutusvõimaluste tõttu võetakse lahenduses kasutusele REST lähenemine koos JSON keelega.

2.3.1 Serveripoolse tehnoloogia valik

Järgnevalt tutvustatakse serveripoolseid programmeerimisraamistikke ja nende programmeerimiskeeli, mis on sobilikud planeeritava rakenduse ehitamiseks. Pärast võrreldakse neid omavahel ja valitakse sobivaim kasutamiseks.

Programmeerimiskeeled:

- Java – Objektorienteeritud ja klassipõhine programmeerimiskeel. Selle suurimateks eelisteks on kiire jõudlus, turvalisus ja vastupidavus. Samuti on sellel suur kasutajabaas ja võimalused testimiseks. Lisaks on selle rakendusi võimalik kasutada praktiliselt kõikidel seadmetel [9].

- C# - Microsofti poolt loodud objektorienteeritud keel. C# on vabavaraline ja tänapäeval üks kiiremini arenevaid keeli. On olemas tugi erinevat tüüpi rakenduste loomiseks ja kasutatav põhilistel operatsioonisüsteemidel [10].

- Php – Vabavaraline programmeerimiskeel, mis on laialt kasutatud. Eriti kasutatakse seda veebilehtedele ja -rakendustele funktsionaalsuse lisamiseks. Php üheks eeliseks on selle kasutamise lihtsus, mis võimaldab uutel kasutajatel kiiresti seda hakata kasutama [11].

- JavaScript – Programmeerimiskeel, mis võimaldab muuta veebilehed dünaamiliseks. Sellega on samuti võimalik luua serveripoolseid lahendusi [12]. Üheleherakenduse puhul on eeliseks see, et kasutajaliidese ja serveri jaoks saab kasutada sama keelt [13].

Raamistikud:

- Spring Framework - On populaarseim raamistik Java veebirakenduste ehitamiseks. Spring võimaldab lihtsamalt koostada APIsid, testida rakendust ja ühenduda andmebaasiga. Võimaldab jooksutada rakendusi erinevatel operatsioonisüsteemidel [14].

- .NET - Vabavaraline platvorm erinevat tüüpi rakenduste arendamiseks. Võimaldab luua ka veebirakendusi ASP.NET raamistikus ja jooksutada neid erinevatel

operatsioonisüsteemidel. Rakendusi valmistamiseks saab kasutada C#, F# ja Visual Basic programmeerimiskeeli [15].

- Laravel - Php programmeerimiskeele raamistik veebirakenduste loomiseks. Laravel on skaleeruv ja pakub võimalusi andmebaasiga ühendamiseks, ühik testide koostamiseks ning seda on samuti võimalik kasutada erinevatel operatsioonisüsteemidel [16].

- ExpressJS - Node keskkonna raamistik veebirakenduste ehitamiseks. Programmeerimiskeeleks on Javascript. Iseseisvalt pole sellel palju võimalusi, kuid sellele rohkelt kirjutatud vabavaralisi teeke, millega on võimalik lisada palju funktsionaalsust [17].

Valiku tegemisel ei ole otseselt tähtis, kui palju kasutajaid suudab rakendus ühel hetkel vastu võtta, sest potentsiaalne kasutajate arv ei ole väga suur. Selle tõttu puudub vajadus arvestada tehnoloogiate jõudlust. Arvestades autori isiklikku kogemust ja kõigi raamistike funktsionaalsust jäävad kõige rohkem esile Spring Framework ning .NET keskkond. Mõlemal head võimalused andmebaasiga ühendamiseks, head testimise võimalused ja rohkelt dokumenteeritud. Kahest valikust jääb siiski rohkem esile .NET, esiteks autori varasema kogemuse pärast ja Entity Frameworki Core'i pärast. Entity Framework võimaldab andmebaasi kaardistamist ja lisavõimalusena saab kasutada ka LINQ'i (*Language Integrated Query*) [18]. LINQ on C# programmeerimiskeele funktsionaalsus, mis võimaldab andmebaaside päringute loomist C# keeles [19]. Lisaks on autor varasemalt teinud suuremaid ja rohkem projekte kasutades .NET'i. Tulenevalt sellest osutub serveripoolse tehnoloogia valikuks .NET kasutades C# programmeerimiskeelt.

2.3.2 Klientrakenduse tehnoloogia valik

Arvestades seda, et rakenduse valmistamisel on plaanis kasutada REST API lahendust, tuleb valida lahendus, mis võimaldaks päringute tegemist serveripoolsele lahendusele ja päringutulemuste kuvamist sobilikus formaadis. Selle jaoks on sobilik lahendused JavaScripti kasutajaliidese raamistikud. Enamus JavaScripti kasutajaliideste raamistikke on mõeldud üheleherakenduste loomiseks mõeldud. Üheleherakendus ehk SPA (*Single Page Application*) on veebileht, kus funktsionaalsus toimub ainult ühel lehel. Kui vaates, midagi muutub, ei laeta serveri poolt uut vaadet ja vaate protsessimine toimub kasutaja poolel [20]. Järgnevalt tutvustatakse suuremaid ja tuntumaid JavaScripti raamistikke ja teeke.

- React.js - JavaScripti teek veebipõhiste kasutajaliideste loomiseks. React kasutab HTMLi kuvamisel *Virtual DOM*i, mille tõttu on teegi põhjal tehtud rakendused kiired [21].

- Angular - TypeScriptiga ehitatud kasutajaliidese raamistik. Raamistik on komponendipõhine. Sisseehitatud funktsionaalsust on palju. Enamik klientrakenduse jaoks vajalikku funktsionaalsust on juba olemas. Veel on põhjalik dokumentatsioon ja hea arendus tööriistade tugi [22].

- Vue.js - Raamistik kasutajaliideste loomiseks. Vue põhiteek on suunatud ainult vaadete kihile ning on lihtne lisada teistesse projektidesse. Kuid Vue rakendusele on võimalik lisada teeke, mis muudavad selle väga võimekaks raamistikuks üheleherakenduste jaoks [23].

Kõik eelpool väljatoodud raamistikud on sobilikud üheleherakenduste valmistamiseks. Võrreldes funktsionaalsust, mis raamistikega kaasa tuleb on Angularis kõige rohkem võimalusi, hea suuremahuliste projektide ehitamiseks. Õppimisraskuse poolest on Angular kõige nõudlikum, Vue.js ja React.js on lihtsamad [24]. Kuna töö raames valmiv rakendus ei ole kuigi suuremahuline ei ole Angulari suguse raamistiku valimine vajalik. Raamistiku valiku osas tuleb ka veel arvestada autori varasema kogemusega. Kõige paremini tunneb autor Vue'd. React'iga on autor väga vähe kokku puutunud. Kliendipoolseks tehnoloogiaks valitakse Vue raamistik, arvestades asjaolusid, et loodav lahendus ei ole liiga suure mahuga, autoril on kõige rohkem kogemust sellega ning rakenduse jaoks vajalikud funktsionaalsed on kättesaadavad välistes teekides.

2.3.3 Andmebaasi valik

Klient-server mudelis on andmete salvestamiseks vaja andmebaasi. Tänapäeval jagunevad andmebaasid kahte peamisse kategooriasse, SQL (*Structured Query Language*) ja NoSQL (*Not only SQL*) andmebaasideks. SQL andmebaasid on tabelite põhised. NoSQL andmebaasid on dokumendi, võti-väärtus paaride, graafide või laiade tulpade põhised, kus andmed on struktureerimata. Samuti on oluline vahe selles, kuidas andmeid päritakse. SQL baaside puhul kasutatakse andmebaasi põhist versiooni SQL päringukeelest. [25]. Kuna autoril puudub kogemus NoSQL andmebaasidega vaadeltakse edasipidiselt ainult SQL andmebaase. Järgnevalt on kirjeldatakse tuntumaid SQL andmebaase.

- MySQL - Üks kõige populaarsemaid andmebaasisüsteeme, mis on laialt kasutatud suurte firmade poolt. 2010. aastast kuulub see Oracle'ile, mis põhjustas kartusi, et MySQL muutub läbinisti tasulikseks, kuid siiski säilitati ka vabavaraline versioon. MySQLi puhul kiidetakse head dokumentatsiooni. Samuti on see hästi skaleeruv, mis tagab selle kasutatavuse suurte koguste andmete puhul.
- Oracle Database - Samuti Oracle'ile kuuluv andmebaasisüsteem. Võrreldes MySQLiga on sellel rohkem andmetüüpe, andmetalletus võimalusi, täiustatud turvavõimalused ja toetab XML (*Extensible Markup Language*) formaati. Kõige suuremaks miinuseks selle puhul maksumus.
- Microsoft SQL Server - Microsoftile kuuluv andmebaasisüsteem. Positiivseteks omadusteks on selle puhul lihtne kasutatavus, automaatne uuenduste kontrollimine ja nende allalaadimine. Täisversiooni puhul on takistuseks eelkõige suur hind.

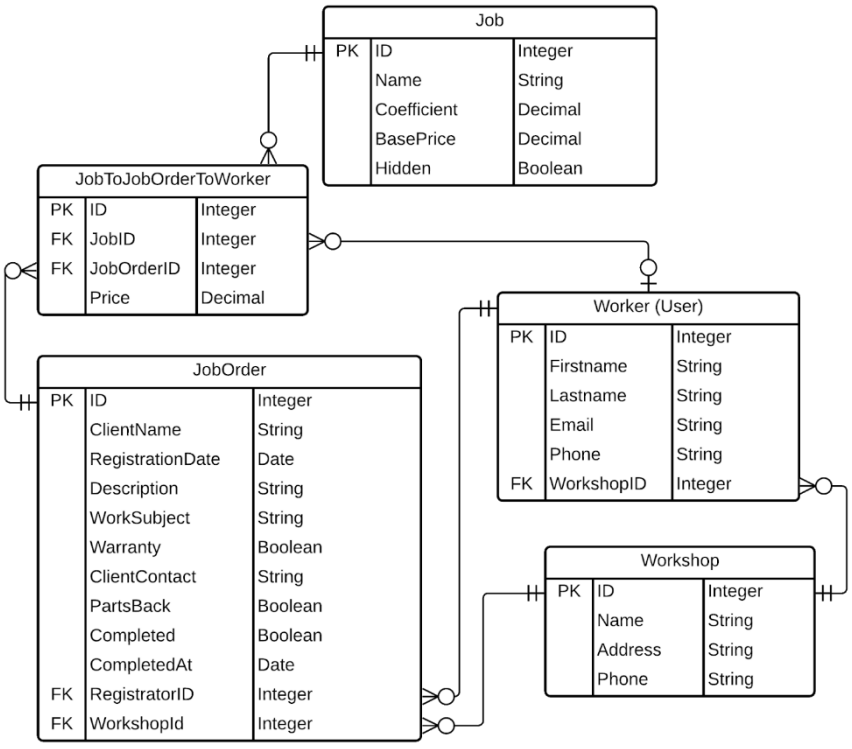
- PostgreSQL - Võrreldes teiste süsteemidega on sellel kõige rohkem võimalusi. Seda saab kasutada kõigi suuremate operatsioonisüsteemide peal ja paljudel programmeerimiskeeltel on hea tugi PostgreSQLi jaoks. Kasutamise eest tasu ei pea maksma [26].

Valides kõikide variantide vahel arvestatakse eelkõige maksumust. Selle tõttu langevad valikust välja Oracle Database ja Microsoft SQL Server. MySQLi ja PostgreSQLi vahelisest valikust valitakse viimane, sest autoril on eelnev igapäevase töötamise kogemus sellega.

3 Rakenduse kirjeldus

3.1 Andmebaasi struktuur

Andmete talletamise jaoks on loodud andmebaasi skeem, mis koosneb tabelitest ja nende vahelistest seostest. Käesolevas lahenduses kasutatakse 5 tabelit. Skeem on koostatud tulenevalt probleemist ja funktsionaalsetest rakenduse nõuetest. Igal tabelil on unikaalne võti ehk ID. Tabelites kirjeldatud on väljad, mida oleks oluline teada. *User* tabel on .NET raamistiku poolt loodud tabel, mis antud lahenduse kontekstis tähistab töötajat või administreerivat kasutajat. Serveri poolses lahenduses on *AppUser* nimetusega.



Joonis 1. Ekraanitõmmis andmebaasi skeemist

JobOrder tabel esindab ühte töökäsku, selles on viited *User* ja *Workshop* tabelitele, millest viimane talletab ühe töökoja andmeid. *User* tabeli viide kirjeldab töö registeerijat, kuna töökäsul saab olla ainult üks registeerija ei ole vahetabelit vaja. *Job* tabel on ühe

konkreetse töö kirjeldus, mida saab töökäsule lisada. Kuna tööde nimekiri on eeldefineeritud ja üks *Job* tabeli kirje saab olla mitme *JobOrder* tabeli kirjega seotud, on nende vahelise seose loomiseks vaja vahetabelit, milleks on *JobToJobOrder*. *JobToJobOrder* tabelis on väli *Price*, mis on töölt saadud hind, mis lisatakse igale kirjele. See on oluline, siis kui on soov muuta töö hinda konkreetses töökäsus. Töö koefitsienti üksik olukordade puhul ei muudeta.

3.2 Serveri poole ülesehitus

Serveri poolne rakenduse osa on valmistatud C# programmeerimiskeelega, kasutades .NET raamistikku. Andmete salvestamiseks kasutatakse EF Core'i, mis on vabavaraline raamistik andmebaasi olemi suhte kaardistamiseks [27]. Iga andmebaasi tabeli kohta on loodud C# klass, milles on tabeli tulpadele vastavad väljad.

Järgmisena on loodud *Repository layer* ehk repositooriumi kiht. Repositoorium on klass, milles on realiseeritud andmete pärimis loogika [28]. Repositooriumite ühendamiseks kasutatakse UOW (*Unit Of Work*). Juhul, kui soovitakse kasutada mitut erinevat tabelit ühe päringu ajal, võib tekkida olukord, kus repositooriumite kasutamiseks tuleb kirjutada palju korduvat koodi. Selle olukorra lahendamiseks saab kasutada UOW lahendust. Repositooriumite ühendamiseks saab teha ühe üldise repositooriumi, mida UOW kasutab [29].

Äriloogika teostamiseks on kasutatud *Service Layer*'it ehk teenuskihti. Repositooriumid tegelevad andmebaasi suhtlusega, siis teenused teenuskihis tegelevad nende valideerimisega ja muutmisega. Teenuskiht on vahelüliks repositooriumite ja kontrollerite vahel [30].

Selleks, et rakendus saaks välise maailmaga suhelda on vaja kontrollereid. Kontroller vastutab selle eest, et kasutaja HTTP päringule tuleks vastus. Vastuse sisu oleneb sellest, milliselt on kontroller programmeeritud [31].

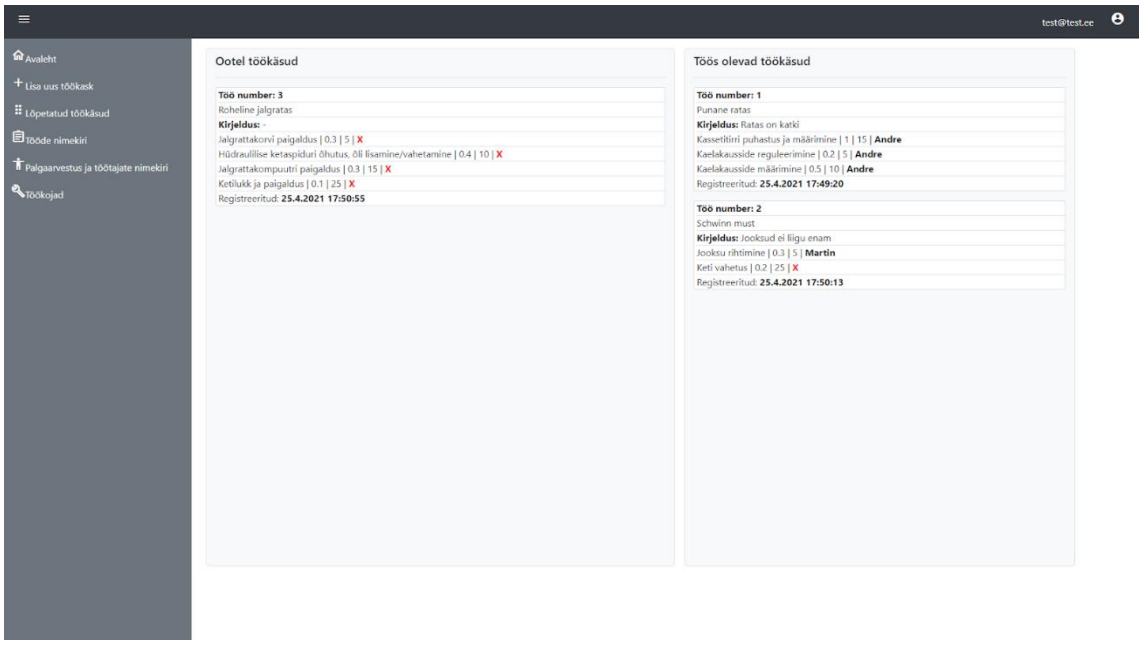
Välja arvatud sisselogimise eest vastutav kontrolleri sihtkoht, nõuavad antud rakenduse puhul kõikide kontrollerite sihtkohad autoriseerimist. Autoriseerimiseks on kasutatud JWT'd (*Json Web Token*), mis on standard turvaliseks andmete vahetamiseks JSON objekti kujul [32]. Kui kasutaja sisselogimise andmed on õiged luuakse serveris tema

andmete põhjal krüpteeritud võti, mis saadetakse talle tagasi. Kui kasutaja soovib teha rakenduses, midagi sellist, mis vajab konkreetseid kasutajaõiguseid, siis ta peab HTTP päringuga kaasa andma tema jaoks loodud võtme.

3.3 Kasutajaliidese kirjeldus

Rakenduse kasutamiseks on vaja kasutajaliidest, mis antud lahenduse puhul on loodud Vue.js raamistikuga. Antud lahenduse puhul pole kasutatud JavaScripti programmeerimiskeelt vaid hoopis TypeScripti, mis on vabavaraline programmeerimiskeel, mis lisab JavaScriptile tüüpimise, et vältida vigade tekkimist koodis [33]. Vue kasutab visuaalide loomiseks HTMLi (*Hypertext Markup Language*) ja CSSi (*Cascading Style Sheets*). HTML on kirjeldab veebilehe struktuuri ja elemente, mis selles peaks olema [34]. CSS kirjeldab, milliselt HTML elemente kuvada [35]. Lisaks Vue'le on kasutatud ka Bootstrap kasutajaliidese raamistikku, mis muudab lihtsamaks rakenduse disani loomise HTML ja CSS põhjal [36]. Koodistruktuur koosneb Vue klassi komponentidest. Neile on juurde tehtud liidesed, mis vastavad serveri rakendusest saadavatele andmetele. Liideste põhjal on loodud andmepäringu klassid, mis kõik rakendavad ühte baasteenust, kus on olemas funktsionaalsus päringute tegemiseks, mis serveri rakenduse kõikides kontrollerites kordub, nagu näiteks ühe domeeni tüübi kõikide kirjete pärimine. Enamus Vue komponentidest kujutavad ühte eraldi vaadet, milles on vastavalt programmeeritud loogika funktsionaalsuse saavutamiseks. Osad komponendid on ainult mõeldud kasutamiseks teiste sees.

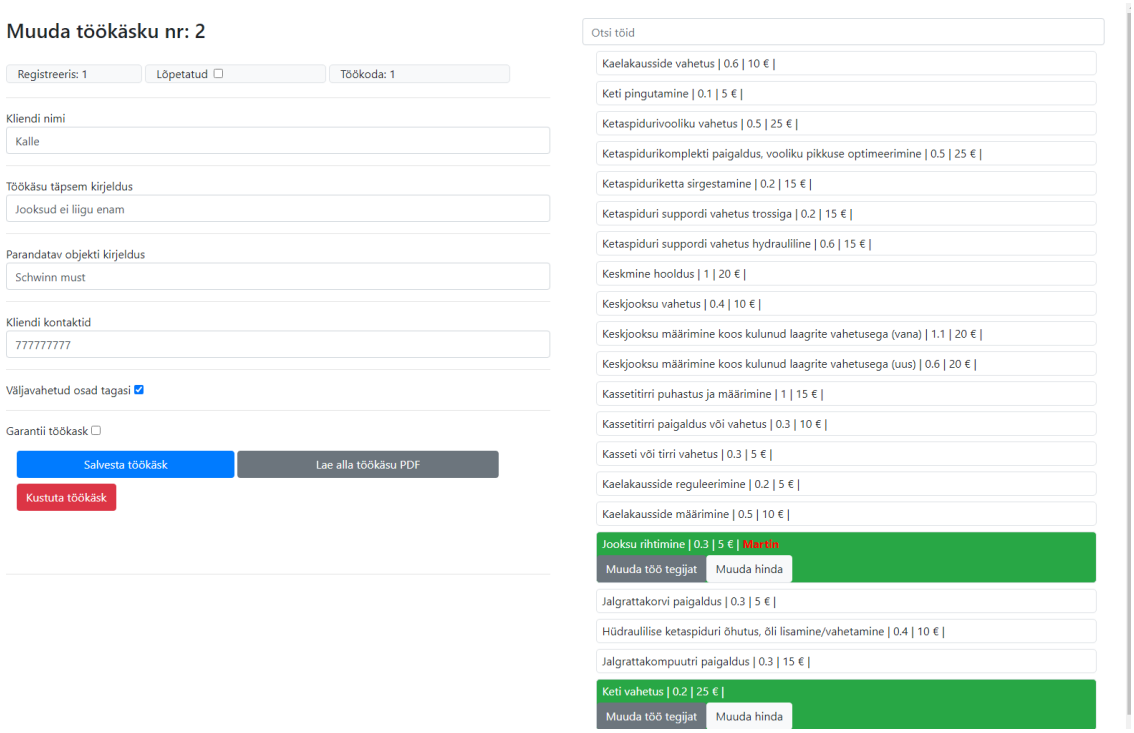
Rakenduse kasutamiseks tuleb kasutajal esmalt sisse logida. Kui kasutaja ei ole sisse logitud suunatakse ta alati tagasi sisselogimise lehele. Pärast sisse logimist suunatakse kasutaja rakenduse põhivaatesse. Põhivaates kuvatakse töökäske kahes tulbas. Ühes on töösse võetud ja teises töösse võtmata töökäsud. Töösse võetud käskudel on töö kirjed, millel on töö tegija lisatud.



Joonis 2. Kasutajaliidese avaleht

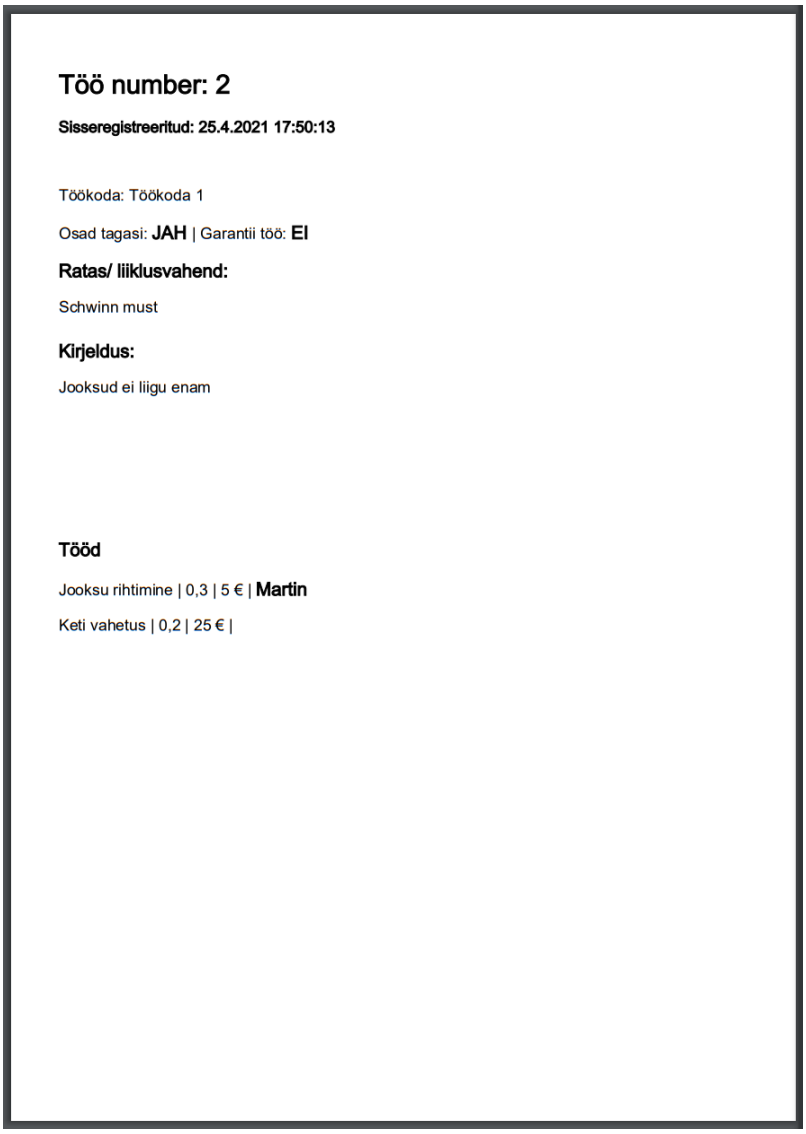
Lehekülje päise paremas nurgas on kuvatud tööaja e-mail ja ikoon, mille peale vajutades avaneb rippmenüü valikutega kasutaja seadetesse suundumiseks ja välja logimiseks. Vaate vasakus servas on menüü navigeerimiseks erinevate vaadete vahel. Pilt 2 menüüs on kuvatud kaks lisa valikut, mis on nähtavad ainult administraatorile, milleks on “Palgaarvestus ja töötajate nimekiri” ning “Töökojad”.

Töökäskude loomiseks ja olemasolevate detailvaate kuvamiseks kasutatakse sama vaadet. Kuid olenevalt sellest, mida teha soovitakse muudetakse vaate osasid vastavalt.



Joonis 3. Töökäsu detailvaade

Pildil 3 on kujutatud olemasolevat töökäsku. Vaate vasakus pooles saab muuta töökäsu tööde nimekirja. Vajutades töö peale, mis ei ole roheliseks värvitud, lisatakse töö töökäsu tööde nimekirja. Vajutades nuppu “Muuda töö tegijat” avaneb aken, milles saab valida tööle töötaja, kes seda tegema hakkab. Ühel tööl saab olla korraga ainult üks töötaja. Kui muuta töö hinda, siis muutub hind ainult konkreetse töökäsu piires. Muudatuste salvestamiseks tuleb vajutada “Salvesta töökäsk” nuppu. Töökäsu vaates on ka valik PDF faili alla laadimiseks, mida oleks võimalik parandatava eseme külge panna. Vajutades roheliseks värvitud tööd arvestatakse see töökäsust välja. Töökäsu kustutamisel avatakse väike aken valiku kinnitamiseks. Loomisel ei kuvata registeerijat, kas töökäsk on lõpetatud ja töökoja numbrit. Lisaks on loomisvaates vasakus pooles ainult üks nupp, töökäsu salvestamiseks.



Joonis 4. Töökäsu printimisvorm

Lõpetatud töökäskude vaates kuvatakse töökäske, mis on märgitud lõpetatuks. Seal on lisaks otsinguriba, millega saab otsida töökäsu selles sisalduva informatsiooni põhiselt.

Tööde vaates kuvatakse ka töid, mis ei ole nähtavad töökäsu loomisel ja muutmisel. Põhjuseks on see, et need võivad olla juba töökäsule lisatud ja neid võib vaja olla töötasude arvutamisel. Tööde vaates on nupp, mis suunab kasutaja uue töö loomise vaatesse. Lisaks on tööde nimekirjas igal tööl nupp, mis suunab kasutaja selle muutmise vaatesse.

Palgaarvestuse ja töötajate nimekirja vaates kuvatakse igat töötajat, tema tehtud tööde pealt saadud tasusid ja kui ta registreeris töökäske, siis ka nende pealt arvestatud registreerimistasusid. Vaikeseade tasude ajavahemiku jaoks on jooksva kuu esimene ja viimane

päev, kuid vahemikku on võimalik muuta. Valides töötaja saab liikuda töötaja detailide vaatesse.

Valitud töötaja detailide vaadet saavad näha ainult administraator ja töötaja, kelle kasutaja kuulub. Lisaks kasutaja andmete muutmise võimaluse on seal ka töötaja töötasud kuvatud. Sarnaselt administraatorile saadaval olevale töötajate nimekirjale, kasutatakse töötaja detailvaates ajavahemiku vaikeseadena jooksva kuu esimest ja viimast päeva, millele lisaks on funktsionaalsus vahemiku valimiseks. Veel kuvatakse kõiki töökäske, millel töötaja on registreerija või teeb töid.

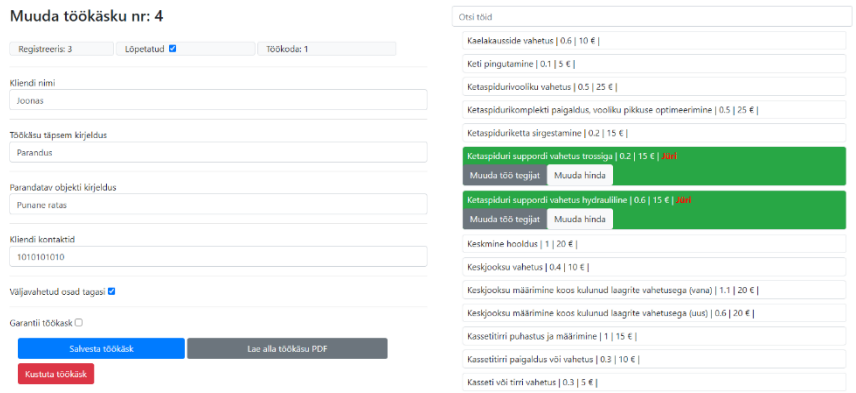
Töökodade kuvas on nimekiri töökodadest ja nende andmetest. On võimalused suunduma ks loomise ja muutmise vaadetesse.

4 Testimine ja tulemused

4.1 Testimine

Põhiline funktsionaalsus, mis rakenduses testimist vajab on töötasude arvestamine. See on selle rakenduse jaoks kõige suurem probleem, mida lahendama peaks. Järgnevalt on välja toodud kolm olukorda. Testide eel on töötasude nimekirjas iga töötaja tööde pealt teenitud ja registreerimistasude summad 0 eurot.

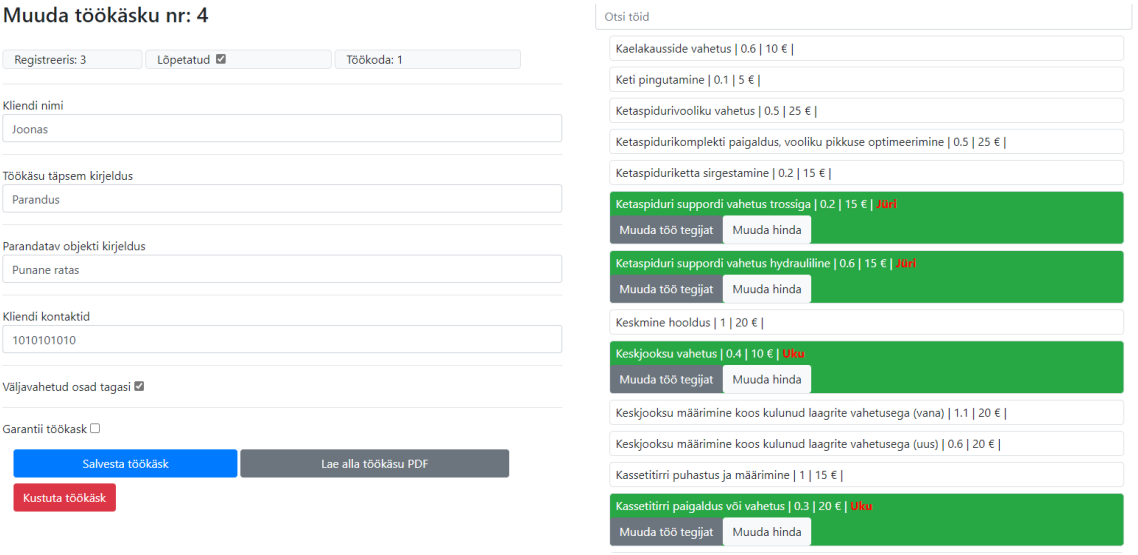
Pildil 5 on kujutatud lõpetatud töökäsku. Töökäsu registreeris Andre. Mõlemad tööd sooritas Jüri. Esimese töö hinnaks oli 15 eurot ja koefitsient oli 0.2, mis töö tasuna oleks 0.2 * 15 ehk 3 eurot. Teise töö eest saab Jüri 0.6 * 15 eurot ehk 9 eurot. Kogu summa Jürile töökäsu pealt on 12 eurot. Registreerimistasu töö kohta on 10%, mis antud töökäsu kohta Andrele oleks 15 * 0.1 + 15 * 0.1 = 3 eurot. Tulemused on kuvatud töötasude nimekirjas Pildil 6.



Joonis 5. Töökäsk nr.4, mille registreeris Andre, töid teeb Jüri.

Joonis 6. Töökäsu nr. 4 tasud.

Järgnevalt lisatakse samale töökäsule veel kaks tööd. Eesmärgiks on näidata, kuidas muutuvad registreerimistasud ja kuidas toimib süsteem, kui töö maksumus on töökäsus muudetud. Tööde sooritajaks on määratud Uku. Töö, mille nimetus on “Kassetitirri paigaldus või vahetus”, hinda on konkreetse töökäsu puhul muudetud 10 eurolt 20 eurole. Antud seisu tõttu suureneb Andre registreerimistasude summa 6 euro peale. Uku teenitav summa kahe töö pealt on 10 eurot.

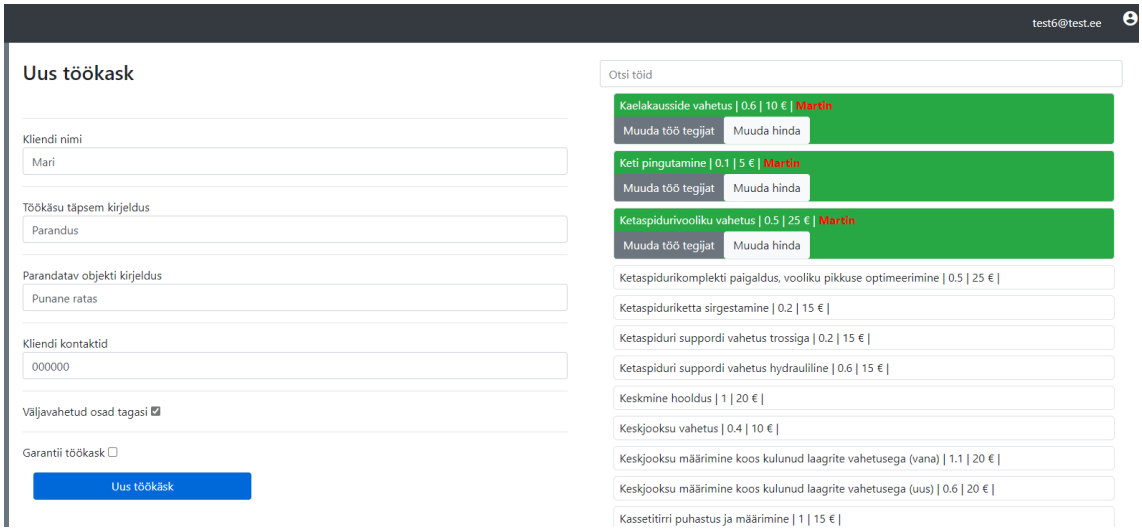


Joonis 7. Töökäsk nr.4, millele on juurde lisatud tööd, mida teeb Uku.

Joonis 8. Töökäsu nr. 4 tasud.

Viimasena luuakse Martini kasutaja alt uus töökäsk, millele lisatakse kolm tööd. Eesmärk on näidata olukorda, kui töö registreerija ja tööde sooritaja on sama isik. Pärast uue loo-

mist märgitakse töökäsk lõpetatuks. Tööde pealt saadud summa arvutatakse järgneva tehtega 0.6 * 10 + 0.1 * 5 + 0.5 * 25 = 19 eurot. Registreerimistasude summa on 25 * 0.1 + 5 * 0.1 + 10 * 0.1 = 4 eurot. Vastavad tulemused on kuvatud pildil 10.



Joonis 9. Töökäsu, mille registreerib ja millel töid teeb Martin.

Joonis 10. Tasud pärast Martini tehtud töökäsu lõpetamist.

4.2 Tulemused

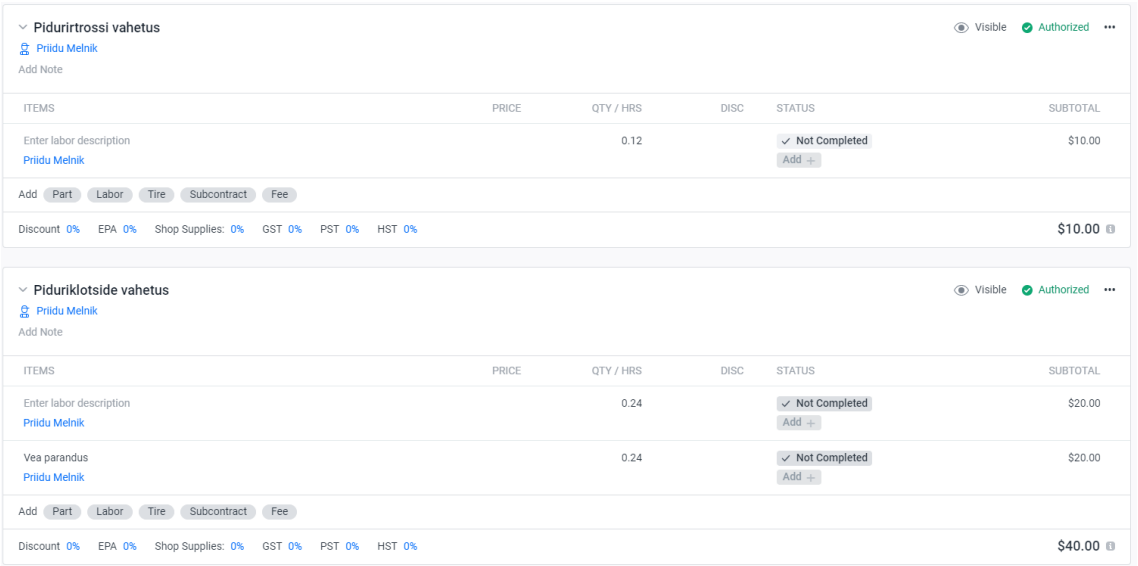
Kõik funktsionaalsed nõuded said täidetud. Rakendus on saavutanud eesmärgi muuta töötajate töötasude arvutamine lihtsamaks. Puudub vajadus käsitsi arvutada töötajate tasusid iga töökäsu kohta. Protsess on nüüd automatiseeritud. Töökäskude registreerija saab iga märgitud töö pealt 10% maksumusest. Tööde sooritajate tasud arvestatakse õigesti vastavalt tööde maksumuse ja koefitsientide korrutiste summadele. Töökäskude loomise protsess on muutunud lihtsamaks. Kasutajatel on lihtsasti võimalik otsida töid ja töökäske.

Võimalik on luua uusi töökodasid ja siduda kasutajaid nendega. Administraatoril ja töötajatel on kiirelt võimalik näha töötasusid määratud ajavahemikus. Võimalik on luua uusi töid ja neid vajadusel töökäsu vaatest peita.

Samuti said täidetud mittefunktsionaalsed nõudmised. Rakendus on Bootstrapi raamistikku kasutades muudetud selliseks, et rakendust on ka võimalik kasutada mobiilibrauseris ja arvuti brauseris. Lahendus kasutab eesti keelt.

4.3 Olemasolevate rakenduste võrdlus

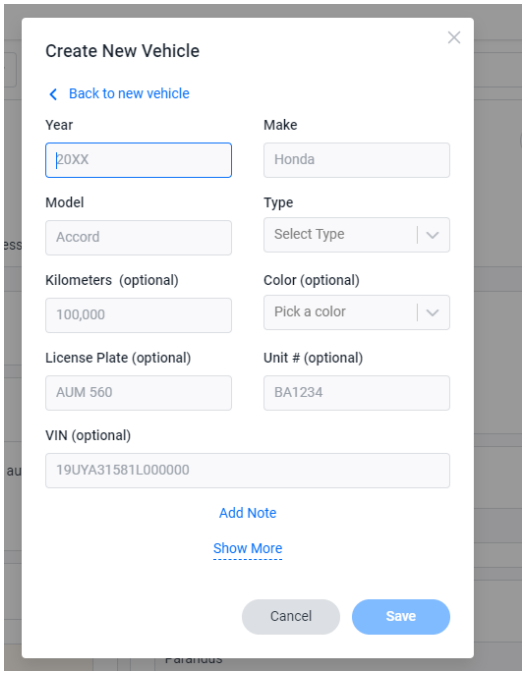
Valminud rakendust võrreldakse Repro ja Shopmonkey'iga, nende lihtsa ligipääsetavuse tõttu. Vaadatakse võimalusi uue töökäsu loomiseks. Shopmonkey puhul eksisteerib enamus vajalikku funktsionaalsust. Saab luua töökäsu ja sellele töid lisada, võimalik on tookäsu vormi printimine. Varasemalt salvestatud töid saab uuesti valida. Suureks puuduseks on see, kuidas töötasude lisamine töödele käib. Töötasu saab ära märkida, kuid sellele ei saa lisada töö spetsiifilist koefitsienti ning see on omavahelises sõltuvuses töötatud tundidega. Üheks probleemiks on veel see, et töökäsk lõpetada tuleb läbida protsess, kus tuleb kinnitada kliendi makse sooritamine.



Joonis 11. Shopmonkey töökäsk.

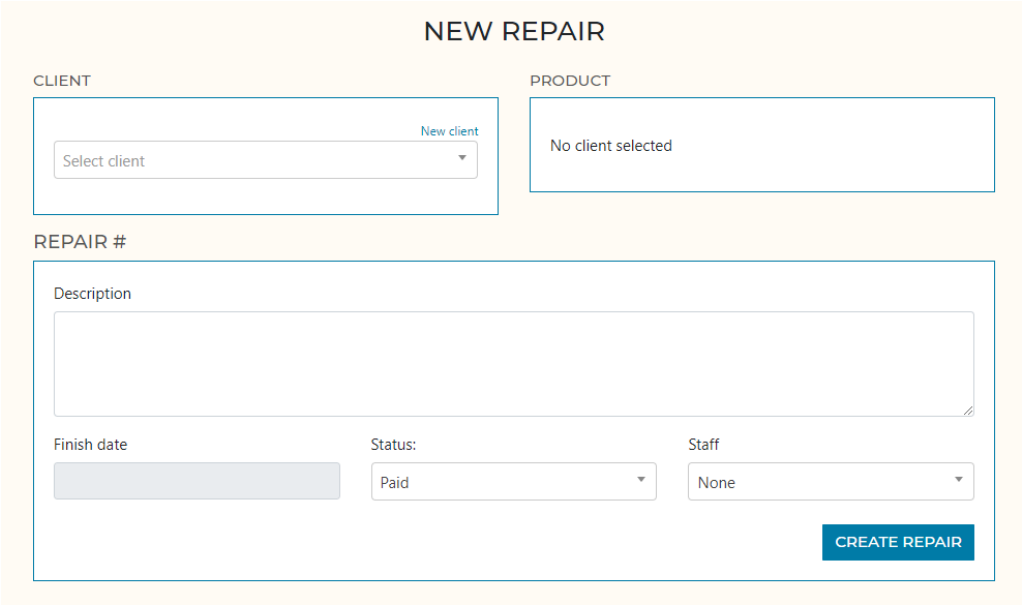
Rakenduse puhul on kliendi andmete ja parandatava liiklusvahendi lisamine küllaltki tülikas, kui need varasemalt juba ei ole loodud. Nende loomiseks avanevad uued aknad,

kus on mitmeid sisendvälju andmete sisestamiseks. Teatud väljade täitmine on kohustuslik. Keeruline ja segadust tekitav võib olla eriti just liiklusvahendi loomine. Kuna Shopmonkey on suunatud ka autodele on seal mitmeid välja, mille täitmine on vajalik ainult autode puhul, kuid nende olemasolu on võib kasutajas segadust tekitada ja aeglustada lisamist.



Joonis 12. Shopmonkey liiklusvahendi lisamine.

Repero puhul on esmalt võimalik märgata seda, et sellel on puudu tööde lisamise võimalus uuele töökäsule. Hiljem töökäsu vaates pole isegi võimalik valida varasemalt salvestatud töid ning nendele lisaks koefitsiente. Puudub sootuks võimalus lisada eraldi töid. Reperos käib töötasu arvestamine tunnitasu aluses, kus kõigil on sama tasu. Mehaanik peab märkima töötatud tundide mahu, mis korrutatakse tunnitasuga. Otseselt ei ole võimalik printida töökäsku, kuid saab printida arve, mida saaks kasutada töökäsuna. Kuna puudub varasemalt salvestatud tööde nimekiri, mida saaks töökäsule lisada, ei ole rakendus piisavalt sobilik kasutamiseks Velomarketi probleemi puhul, kus on oluline kiire tööde lisamine töökäsule.



Joonis 13. Repero uue paranduse (töökäsu) loomise vaade.

Mõlema rakenduse puhul on sarnaseks probleemiks see, et puudub sobilik lahendus Velomarketi koefitsiendi põhise töötasude arvestamiseks. Kuid mõlemal oli veel omadusi, mis vähendasid nende sobilikust kasutamiseks. Käesoleva lõputöö käigus valminud lahenduse kõige suuremaks eeliseks on võimalus arvestada ja näha sooritatavate tööde koefitsiente. Samuti võrreldes teiste rakendustega on töökäsu vaade loomisel ja muutmisel on kompaktsem, mis võib kiirendada töökäskudega tegelemist. Uue lahenduse eeliseks konkreetse probleemi puhul saaks ka veel lugeda üldisema lihtsuse, sest lisafunktsionaalsust on vähem. Eriti on see märgatav võrreldes Shopmonkey lahendusega, kus on väga palju lisasid ja valikuid, mis ei ole ainult töödega seotud. See võib aeglustada lahenduse kasutamise õppimist ja tekitada töötajates segadust ning vigu kasutamisel.

4.4 Edasiarenduse võimalused

Rakendusel on rohkelt võimalusi edasiarenduseks. Olemasolevate lahenduste võrdluses välja toodud variantidel, kõigil on võimalused muude toimingute jaoks, kui ainult tööde haldamine.

Võimalustest, mida juurde saaks lisada, kõige olulisem oleks ilmselt integratsioon arvete loomise ja kassasüsteemiga. See tähendaks muidugi seda, et tuleks arvestada ka töös kasutatavate osadega. Sellest tulenevalt saaks rakendusele veel omakorda juurde ehitada inventuuri haldamise süsteemi.

Arvestades seda, et süsteemis juba arvutatakse töötajate töötasusid, oleks hea võimalus juurde ehitada süsteem töötajate töö planeerimiseks ja graafikute loomiseks, sest töökojas töötatakse graafiku aluses ning lisaks tööde pealt saadud tasudele on neil ka tunnitasu.

Turg taoliste lahenduste jaoks on küllaltki suur, sest väga palju firma, mis tegelevad asjade remontimisega vajada oma igapäevaste tööprotsesside haldamiseks mingisugust lahendust. Antud lahenduse puhul kõige suurimaks miinuseks laiemal turul võib olla selle kitsas suunitlus, mis arvestab ainult konkreetse firma tööprotsessidega. Kui pakkuda lahendust teistele ettevõtetele, tuleks tõenäoliselt suuri muudatusi teha.

Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua lahendus, mis muudaks Velomarket OÜ jalgrataste ja muude kergliiklusvahendite parandamistööde haldamise ning nende põhjal töötasude arvestamise lihtsamaks. Põhiliseks probleemiks oli liigne sõltuvus paberformaadist ja sellest tulenev keeruline palgaarvestus. Eesmärgiks oli seda sõltuvust vähendada.

Probleemi lahendamiseks uuris töö autor firma olemasolevat tööprotsessi rattaparandus tööde korraldamisel ja töötasude arvutamisel. Uuriti olemasolevaid lahendusi, mis võiks sobida probleemi lahendamiseks. Olemasolevatel rakendustel kõigil oli teatavad puudused, miks need ei sobinud firma probleemi lahendamiseks. Uurimistulemuste põhjal koostas autor funktsionaalsed ja mittefunktsionaalsed nõuded. Pärast nõuete koostamist valiti sobilikud tehnoloogiad nende täitmiseks.

Rakendus valmis hajusa veebirakendusena. Lahenduse serveripoolse tehnoloogia valikuks osutus C# programmeerimiskeel koos .NET raamistikuga. Andmete salvestamiseks kasutati PostgreSQL andmebaasi. Kasutajaliides loodi Vue JavaScript raamistikuga kasutades TypeScript programmeerimiskeelt.

Rakenduse valmides kirjeldati, milline see on ja kuidas see toimib. Pärast seda katsetati sellel põhineva töötasude arvutamise süsteemi. Katsetati olukorda, kus töökäsuga on seotud üks registreerija ja üks töötaja. Järgmisena prooviti olukorda, kus töökäsule lisatakse juurde teine töötaja. Viimasena vaadati olukorda, millal töötaja ja registreerija on sama kasutaja. Katsetuste põhjal oli võimalik kinnitada, et töötasude arvestus toimub plaanipäraselt.

Töö eesmärk sai täidetud. Loodi rakendus, mis võimaldab luua kiiremini töökäske, neid hallata ja nende põhjal töötajate palgaarvestust koostada. Kõik funktsionaalses ja mittefunktsionaalsed nõuded said täidetud. Autori jätkab süsteemi arendamist, sest firmal soov selle põhjal enda tööprotsesse paremaks muuta.

Viited

[1] Repero, “Features,” [Võrgumaterjal]. https://repero.me/en/features. [Kasutatud 10.04.2021]

[2] Repero, “Tutorial,” [Võrgumaterjal]. https://repero.me/en/tutorial. [Kasutatud 10.04.2021]

[3] Shopmonkey, “Bicycle Repair Shop Management Software,” [Võrgumaterjal]. https://www.shopmonkey.io/bicycle-shop-software. [Kasutatud 10.04.2021]

[4] AutoFutur, “Koduleheküg,” [Võrgumaterjal]. https://www.autofutur.net/. [Kasutatud 10.04.2021]

[5] L. Wittgenstein, “Distributed Application Architecture,” Wayback Machine [Võrgumaterjal]. https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf. [Kasutatud 10.04.2021]

[6] D.Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, “Web Services Architecture,” W3C, 11.02.2004. [Võrgumaterjal]. https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest. [Kasutatud 10.04.2021]

[7] Red Hat, “What is REST API?,” [Võrgumaterjal]. https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Kasutatud 10.04.2021]

[8] Json.org, “Introducing JSON,” [Võrgumaterjal]. https://www.json.org/json-en.html. [Kasutatud 10.04.2021]

[9] Codeinstitute, “What is Java and is it important?,” [Võrgumaterjal]. https://codeinstitute.net/blog/what-is-java/. [Kasutatud 16.05.2021]

[10] Mahesh Chand, “What is C#?,” C#Corner, 07.03.2020. [Võrgumaterjal]. https://www.c-sharpcorner.com/article/what-is-c-sharp/. [Kasutatud 16.05.2021]

[11] Php, “What is PHP,” [Võrgumaterjal]. https://www.php.net/manual/en/intro-whatis.php. [Kasutatud 16.05.2021]

[12] MDN Web Docs, “What is JavaScript?,” [Võrgumaterjal]. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 16.05.2021]

[13] Clever Solution, “When and Why to Use Node.js for Backend Development,” [Võrgumaterjal]. https://clever-solution.com/when-and-why-to-use-node-js-for-backend-development/. [Kasutatud 16.05.2021]

[14] Spring, “Why Spring?,” [Võrgumaterjal]. https://spring.io/why-spring. [Kasutatud 10.04.2021]

[15] Microsoft, “What is .NET?,” [Võrgumaterjal]. https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet. [Kasutatud 10.04.2021]

[16] Laravel, “Installation,” [Võrgumaterjal]. https://laravel.com/docs/8.x#why-laravel. [Kasutatud 10.04.2021]

[17] MDN Web Docs, “Express/Node introduction,” [Võrgumaterjal]. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. [Kasutatud 11.04.2021]

[18] Microsoft, “Compare EF Core & EF6,” 23.01.2019 [Võrgumaterjal]. https://docs.microsoft.com/en-us/ef/efcore-and-ef6/. [Kasutatud 11.04.2021]

[19] Microsoft, “Language Integrated Query (LINQ),” 02.02.2017, [Võrgumaterjal]. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/. [Kasutatud 11.04.2021]

[20] K. Lawson, “What Is a Single Page Application and Why Do People Like Them so Much?,” Bloomreach [Võrgumaterjal]. https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html. [Kasutatud 11.04.2021]

[21] S. Morris, “Tech 101: What Is React JS?,” Skillcrush, [Võrgumaterjal]. https://skillcrush.com/blog/what-is-react-js/#what. [Kasutatud 11.04.2021]

[22] Angular, “What is Angular?,” [Võrgumaterjal]. https://angular.io/guide/what-is-angular. [Kasutatud 11.04.2021]

[23] Vue.js, “Introduction,” [Võrgumaterjal]. https://vuejs.org/v2/guide/. [Kasutatud 11.04.2021]

[24] Shaumik Daityari, “Angular vs React vs Vue: Which Framework to Choose in 2021,” Codeinwp, 15.03.2021, [Võrgumaterjal]. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/. [Kasutatud 11.04.2021]

[25] M. Smallcombe, “SQL vs NoSQL: 5 Critical Differences,” Xplenty, 19.05.2020 [Võrgumaterjal]. https://www.xplenty.com/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20vertically%20scalable%2C%20NoSQL%20databases%20are,better%20for%20unstructured%20data%20like%20documents%20or%20JSON. [Kasutatud 11.04.2021]

[26] Laura. M, “SQL Database Management System: Which One Should You Choose?,” [Võrgumaterjal]. BitDegree, 05.01.2021, https://www.bitdegree.org/tutorials/sql-database-management-system/. [25.04.2021]

[27] Microsoft, “Entity Framework Core,” 20.09.2020 [Võrgumaterjal]. https://docs.microsoft.com/en-us/ef/core/. [25.04.2021]

[28] L. Nguyen, “How to implement Repository & Unit of Work design patterns in .NET Core with practical examples [Part 1],”, Enlab, 07.01.2021 [Võrgumaterjal]. https://enlabsoftware.com/development/how-to-implement-repository-unit-of-work-design-patterns-in-dot-net-core-practical-examples-part-one.html#:~:text=Repositories%20are%20just%20classes%20that%20implement%20data%20access,data%20access%20logic%20%28CRUD%29%20and%20other%20special%20logic.[25.05.2021]

[29] T.Dykstra. “Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (9 of 10),” Microsoft, 30.07.2013 [Võrgumaterjal]. https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application. [25.04.2021]

[30] S. Walther, “Validating with a Service Layer (C#),” Microsoft, 02.03.2009 [Võrgumaterjal]. https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/models-data/validating-with-a-service-layer-cs. [25.04.2021]

[31] R. Andreson, “Part 2, add a controller to an ASP.NET Core MVC app,” [Võrgumaterjal]. Microsoft, 23.01.2021, https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-controller?view=aspnetcore-5.0&tabs=visual-studio. [25.04.2021]

[32] JWT, “Introduction to JSON Web Tokens,” [Võrgumaterjal]. https://jwt.io/introduction. [25.04.2021]

[33] TypeScript, “What is TypeScript?,” [Võrgumaterjal]. https://www.typescriptlang.org/. [25.04.2021]

[34] W3schools, “HTML Introduction,” [Võrgumaterjal]. https://www.w3schools.com/html/html_intro.asp. [25.04.2021]

[35] W3schools, “CSS Introduction,” [Võrgumaterjal]. https://www.w3schools.com/css/css_intro.asp. [25.04.2021]

[36] T. Bacinger “What is Bootstrap? A Short Bootstrap Tutorial on the What, Why, and How,” Toptal, [Võrgumaterjal]. https://www.toptal.com/front-end/what-is-bootstrap-a-short-tutorial-on-the-what-why-and-how. [25.04.2021]

Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Priidu Melnik

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Velomarket OÜ parandus- ja hooldustööde haldusrakenduse loomine", mille juhendaja on Meelis Antoi

   1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

29.04.2021