TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jaan Erik Lepp 206191IABB

Laohaldustarkvara planeerimine ning arendamine Cargoson OÜ-le

Bakalaureusetöö

Juhendaja: Karl-Erik Karu
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: Jaan Erik Lepp

16.05.2023

Annotatsioon

Käesoleva lõputöö eesmärk on planeerida ning arendada laohaldustarkvara Cargoson OÜ-le. Kogemus ettevõtetega suhtlemisel on näidanud, et ettevõtetel puudub vajadustele vastav tarkvara lao töö haldamiseks. Laohaldustarkvara kasutamine võimaldab paremat informatsiooni liikumist, kõrgemat andmete kvaliteeti ning ajalist võitu.

Eesmärgi saavutamiseks analüüsis autor Cargosoni eksisteerivat prototüüpi ning turul pakutavaid alternatiivseid lahendusi. Analüüsi tulemusena saadi paika nõuded lõputöö käigus valmivale rakendusele. Valminud rakendus võimaldab kasutajatel registreerida ettevõttele konto, seadistada laadimisalad ning hallata kalendri vaates laadimisi. Laadimistega seonduv info liigub seejuures kõigile osapooltele ka meili teel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 4 peatükki, 10 joonist, 1 tabel.

Abstract

Designing and Development of Dock Scheduling Software for Cargoson OÜ

The aim of this bachelor's thesis was to design and create dock scheduling software for Cargoson OÜ. Experience from communication with companies has shown that companies do not have proper software for managing warehouse loadings. Dock scheduling software can help companies to improve information movement, data quality and help warehouses save time.

To design the application, the author first analyzed Cargoson's existing prototype and other alternatives on the market. Through this analysis a set of requirements were created for new application. In the developed application users can register their account, configure docks and manage their loadings in calendar view. All the relevant information will also be sent to parties via email.

The thesis is in Estonian and contains 25 pages of text, 4 chapters, 10 figures, 1 table.

Lühendite ja mõistete sõnastik

Convention over configuration	Kokkulepe enne konfiguratsiooni.
CSS	*Cascading Style Sheets* ehk veebilehe küljendamisel kasutatav märgistuskeel peamiselt kujunduse tarbeks.
HTML	*HyperText Markup Language*, Märgistuskeel veebilehe struktureerimiseks.
Javascript	Veebilehtede skriptimiseks loodud programmeerimiskeel.
Laohaldustarkvara	Tarkvara ladude töö optimeerimiseks.
MVC	*Model-View-Controller* ehk mudel-vaade-kontroller – rakendustes kasutatav arhitektuur, mis jaotab rakenduse kolmeks loogiliseks osaks.
SaaS	*Software as a Service* – pilvepõhine teenus, mille puhul on nii platvorm kui ka funktsionaalsused hallatud teenuse poolt.
SEO	*Search Engine Optimization* ehk veebilehe otsingumootoritele optimeerimine.
UI	*User interface* ehk kasutajaliides
UX	*User experience* ehk kasutajakogemus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Cargoson OÜ on veohaldustarkvara teenust osutav IT-ettevõte, mille missiooniks on aidata lihtsustada ettevõtete logistikat. Klientide logistilised vajadused on piisavalt keerulised, et need lahendatakse mitme vedajaga ning Cargosoni kodulehel on neil võimalik kogu logistika koondada ühte aknasse [1]. Seeläbi saavutatakse oluline ajaline võit ning parem ülevaade enda organisatsiooni logistika üle.

Kogemus ettevõtetega suhtlemisel on näidanud, et ettevõtetel puudub vajadustele vastav tarkvara ladude töö haldamiseks. Kui väljaminevate ja sissetulevate kaupade töövoo ajaline planeerimine on puudulik, siis tekivad ladudesse järjekorrad ning laotöö on ebaefektiivne. Samuti on olulisel kohal info liikumine ja andmete kvaliteet. Infot kauba saabumise ja välja minemise kohta peavad teadma nii müügiosakond, logistikaosakond kui ka laotöötajad. Lisaks on vaja aeg ja koht kauba liigutamiseks kooskõlastada veofirmaga. Paljud ettevõtted lahendavad hetkel neid probleeme Exceli tabelite ning Microsoft Outlook kalendri abil.

Antud lõputöö eesmärgiks on Cargoson OÜ-le luua rakendus laost väljaminevate ja sissetulevate kaupade haldamiseks. Rakendus on eraldiseisev hetkel olemasolevast veohaldustarkvarast. Hiljem peab olema võimalik need omavahel integreerida, kuna mõlemad lahendavad ettevõtete logistikaga seotud probleeme ning on üksteist täiendavad rakendused. Töö tulemusena valmib rakendus, mida saab hakata esimeste ettevõtete peal testima.

Antud töö esimeses peatükis analüüsib autor hetkel turul olevaid rakendusi ladude töö haldamiseks ning samuti Cargoson OÜ algelist prototüüpi antud tarkvarast. Analüüsi käigus pannakse paika vajalikud nõuded ning ideed, kuidas turul teistest sarnastest toodetest eristuda. Töö teises peatükis kirjeldatakse rakenduse arhitektuuri ning valminud vaateid. Samuti tutvustatakse arenduses kasutatud raamistikke ning meetodeid. Töö kolmandas peatükis analüüsib autor valminud rakendust ning põhjendab arenduse käigus tehtud otsuseid. Pannakse kirja võimalikud edasised arendused antud rakendusele.

2 Metoodika

Käesolev peatükk annab ülevaate lõputöö käigus arendatavast objektist ning kasutatud tööriistadest. Lisaks kirjeldab autor arendusprotsessi ning selles kasutatud põhimõtteid.

2.1 Objekti kirjeldus

Laohaldus on valdkond, mida digitaliseerimine saab oluliselt efektiivsemaks muuta. Kui varem toimus kogu protsess Exceli, emailide ja telefoni kõnede abil, siis nüüd on selle jaoks olemas IT lahendused ehk laohaldustarkvarad. Laohaldustarkvara aitab optimeerida protsesse laos ning seeläbi tuua ettevõttele ärilist kasu [2].

Laadimisalade töö optimeerimine aitab vähendada veoautode ooteaegu ning seeläbi ennetada järjekordi laadimisalades. Lisaks aitab laohaldustarkvara koondada kogu informatsiooni ühte kohta, mille abil paraneb info kvaliteet ning mille põhjal saab hiljem teha statistikat lao töö kohta [2].

2.2 Tööriistad

Antud peatükk annab ülevaate rakenduse arendamiseks kasutatud peamistest tööriistadest ning nendega seotud olulistest mõistetest.

2.2.1 Ruby on Rails

Ruby on Rails on veebirakenduste arendamiseks loodud raamistik, milles kirjutatakse koodi Ruby programmeerimiskeeles. Rails on avatud lähtekoodiga ning mõeldud veebirakenduste arendamiseks. Tänu suurele kogukonnale on see pidevalt uuenemas. Raamistiku suurimateks tugevusteks loetakse kiirust ja lihtsust. Seetõttu on see väga populaarne valik startup ettevõtete, kellel on vaja rakendus valmis saada võimalikult vähese ajaga. Railsi miinusena tuuakse välja kiirust võrreldes teiste veebirakenduste arendamiseks mõeldud raamistikega [3].

Ruby programmeerimiskeel on samuti avatud lähtekoodiga. Selle peamisteks kasutusaladeks on veebirakenduste arendamine ning andmeanalüüs. Ruby süntaks sarnaneb programmeerimiskeeltest kõige rohkem Pythoniga ning järgib minimalistlikku stiili [3].

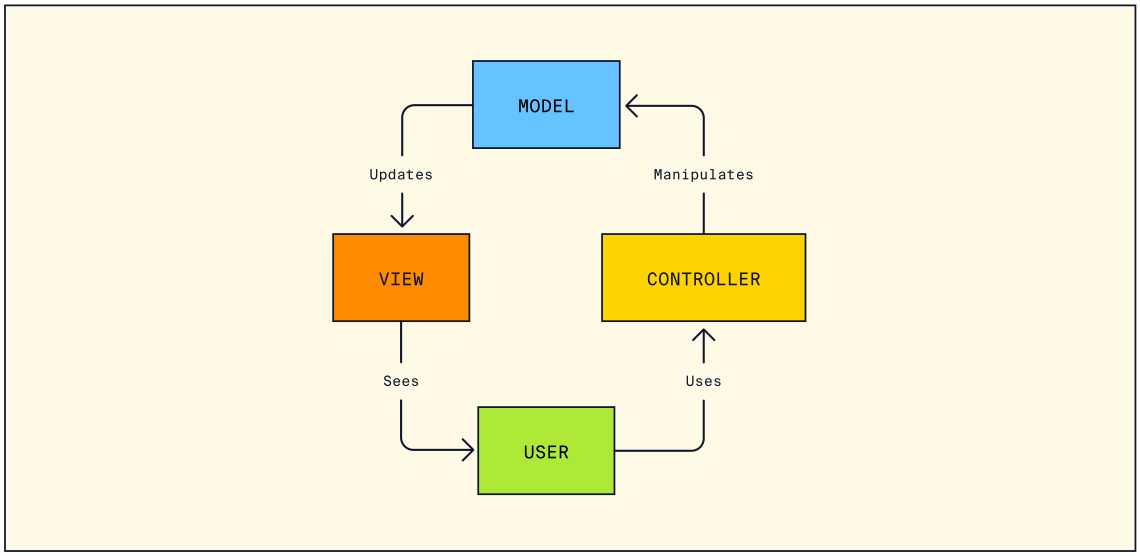
2.2.2 JavaScript

JavaScript on interpreteeritav objektorienteeritud programmeerimiskeel, mida kasutatakse veebilehtede interaktiivseks muutmiseks. Selle abil saab veebilehe andmeid uuendada ilma lehte uuesti laadimata ning kuvada kasutajale erinevaid animatsioone, mis parandavad kasutajakogemust [4].

Antud lõputöö raames kasutab autor kliendipoolset JavaScripti, mis tähendab, et kood töötab otse kasutaja brauseris. Kood laetakse kasutaja brauserisse lehekülje laadimisel koos HTML ja CSS koodiga. Kliendipoolne kood on enamasti seotud erinevate sündmustega, mida kasutaja leheküljel teha saab. Näiteks saab koodijuppe siduda klaviatuuri klahvide vajutamisega või hiireklikiga konkreetsetel elementidel leheküljel [4].

2.2.3 MVC

Model-View-Controller (MVC) on tarkvara arhitektuuri muster, mis koosneb kolmest kihist – mudel, vaade ja kontroller. Mudeli kihis on kirjeldatud andmeobjektid ning millised need objektid andmebaasis on. Samuti valideerib mudeli kiht andmebaasis tehtavaid muudatusi ja loodavaid objekte. Kontrolleri kiht reguleerib kogu rakenduse tööd. Selle kihi ülesandeks on töödelda kasutaja antud käsku ning tagastada vaade. Vaate kiht sisaldab endas peamiselt HTML, CSS ja Javascripti koodi, mis määravad kuidas kasutajale kuvatud andmed välja näevad [5].



Joonis 1 MVC muster [5]

Joonisel 1 on ülevaatlikult näha veebirakenduse töövoog läbi MVC mustri. Lisaks mudelile, kontrollerile ja vaatele, on välja toodud ka kasutaja roll.

2.2.4 Puhas kood

Puhas kood on mõiste, mida kasutatakse kindlate printsiipide, põhimõtete ja praktikate järgi kirjutatud koodi kohta. Puhast koodi iseloomustab kerge loetavus ning arusaadavus. Seda on hiljem lihtne muuta [6].

Põhimõtted, mida puhas kood jälgib [6]:

- Meetodid peavad olema lühikesed ning tegema ainult üht asja. Ideaalis võiks meetodid jääda 20 koodirea piiridesse. Kui mõni meetod on sellest pikem, tuleks see jagada laiali väiksemateks osadeks.

- Tuleb vältida kommentaaride kirjutamist. Tihti kasutatakse kommentaare halvasti kirjutatud koodi parandamiseks, kuid tegelikult tuleb keskenduda sellisel juhul koodi enda loetavamaks muutmisele. Kommentaaride kirjutamine on õigustatud juhul, kui koodijupi eesmärki pole võimalik kirjeldavate nimedega edasi anda.

- DRY ehk *Don't repeat yourself* põhimõtte kohaselt tuleb vältida koodi dubleerimist. Kui sama koodi kasutatakse mitmes kohas, siis hiljem muudatusi tehes tuleb muudatused teha samuti mitmes kohas. Seeläbi on suurem oht ka vigu teha.

- KISS ehk *Keep it simple stupid* põhimõte käsib koodi hoida võimalikult lihtsana. Arendamisel tuleks vältida ebavajalikku keerukust, mis võib teha koodi lugemise teiste arendajate jaoks raskemaks. Samuti on keerukat koodi hiljem raske muuta või refaktoorida.

- Koodi tuleb pidevalt refaktoorida. Lihtne reegel on, et peale muudatuste tegemist peab kood olema alati puhtam kui enne. Nii toimub refaktoorimine pidevalt ning kood püsib puhas.

- Meetodite ning muutujate nimed peavad olema kirjeldavad. Tuleb vältida nimede andmist, millel puudub tähendus. Koodi loetavuse parandamiseks peavad nimed kirjeldama, mida antud meetod teeb või mis väärtust antud muutujas hoitakse. Samuti võimaldavad kirjeldavad nimed neid lihtsasti koodist otsida.

- Testid peavad olema lihtsasti loetavad, ei tohi üksteisest sõltuda ning nende jooksutamine peab olema kiire. Kui testid võtavad kaua aega, siis arendajatel on vähem motivatsiooni neid tihti käima panna.

2.3 Protsess

Autor lähtus rakenduse arendamisel agiilse arenduse põhimõtetest. Agiilne arendus on tarkvaraarenduse metoodika, mis võimaldab luua väärtust väiksemate osade kaupa ning annab projektile juurde paindlikkust. Erinevalt koskmudelist võimaldab agiilne arendus saada pidevat tagasisidet ning reageerida muutustele kohe. Tänu sellele saab vältida liigse aja kulutamist töödele, mis lõpuks kliendi poolt tagasi lükatakse [7].

Agiilse arenduse üks olulisi põhimõtteid on hinnata inimestevahelisi suhteid kõrgemalt kui kirjalikku dokumentatsiooni ning kindlaid protseduure. Pidev suhtlus kliendi ning arendustiimi vahel on olulisem kui omavahelised kindlad kokkulepped. Samuti on probleemide lahendamine ning väärtuse loomine kliendile olulisem rangelt ainult dokumentatsiooni järgimisest [7].

Kuna autoril endal puudub kogemus logistika valdkonnas töötamisel, siis pidev tagasiside oli arenduse jooksul oluline. Agiilse arenduse käigus sai autor pidevalt tagasisidet Cargosoni töötajatelt, kellel on logistika valdkonnas pikaaegne kogemus ning teadmised seal esinevatest probleemidest. Tagasiside põhjal oli võimalik teha pidevalt muudatusi, et

luua võimalikult palju väärtust klientidele raiskamata liigselt aega arendustööde peale, mis lõpuks oleks tulnud ümber teha.

3 Loodava rakenduse analüüs

Käesolev peatükk annab ülevaate Cargosoni enda laadimiskalendri prototüübist ning turul olevatest alternatiivsetest lahendustest. Analüüsi tulemusena paneb autor paika nõuded uuele rakendusele ning põhjendab, kuidas erineb uus rakendus hetkel eksisteerivast prototüübist.

3.1 Cargosoni eksisteeriv prototüüp

Cargosoni SaaS mudelil põhinevas veohaldustarkvaras saavad ettevõtted sisestada ja hallata enda transpordi tellimusi ning saata need otse enda veofirmade süsteemi. Klientideks on keskmise või suure suurusega ettevõtted, kellel on vaja paremat ülevaadet enda logistikast. Sellistel ettevõtetel on tihti palju transpordi saadetisi ning seeläbi ka vajadus hallata efektiivselt enda laost sisse- ja väljaminevaid kaupu. Selleks, et ka seda poolt ettevõtete logistikas toetada, on loodud Cargosoni poolt laadimiskalendri prototüüp.

Cargoson OÜ laadimiskalendri prototüüp on arendatud pidades silmas klientide nõudeid tootele ning valdkonnas pikalt tegutsenud meeskonnaliikmete isiklikku kogemust. Prototüüp on Cargosoni rakenduse osa ning tehtud lisafunktsionaalsusena, mida kliendid kindla kuutasu eest kasutada saavad.

Funktsionaalsed nõuded tootele on järgmised:

- Klient peab saama seadistada enda lao laadimisalad koos lahtiolekuaegadega.
- Klient peab saama kalendris lisada, muuta ja kustutada laadimisi.
- Kalendrile peab olema ligipääs kõigil Cargosoni veahaldustarkvara kasutajatel.

Toode täidab peamised funktsionaalsed nõuded täies ulatuses. Lisaks on veel oluline, et laadimistega seotud osapooled oleksid infoga õigeks ajaks kursis. Selle jaoks saadetakse kõigile osapooltele (veofirma kontaktisik ning laadimise autor) peale laadimise lisamist või muutmist email kogu infoga.

3.1.1 Prototüübi tugevused

Probleemi valideerimine on esimene samm, mida uue toote turule toomiseks on vaja teha [8]. Cargoson on enda laadimiskalendri prototüübiga ära testinud klientide vajaduse toote järgi. Samuti on ära valideeritud klientide tagasiside põhjal peamiste funktsionaalsuste vajadus, mis on kirjeldatud nõuetena peatükis 3.1.

Prototüübi teiseks tugevuseks on ühilduvus Cargosoni veohaldustarkvaraga. Kuna need on üksteist toetavad tooted, siis kasutaja jaoks on mugav neid kasutada ühes aknas. Hetkel saab laadimiskalendrisse aja broneerida kohe peale transpordi tellimuse tegemist. Laadimise juures täidetakse seejuures info vedaja, kauba koguse ja kuupäeva kohta automaatselt.

3.1.2 Prototüübi nõrkused

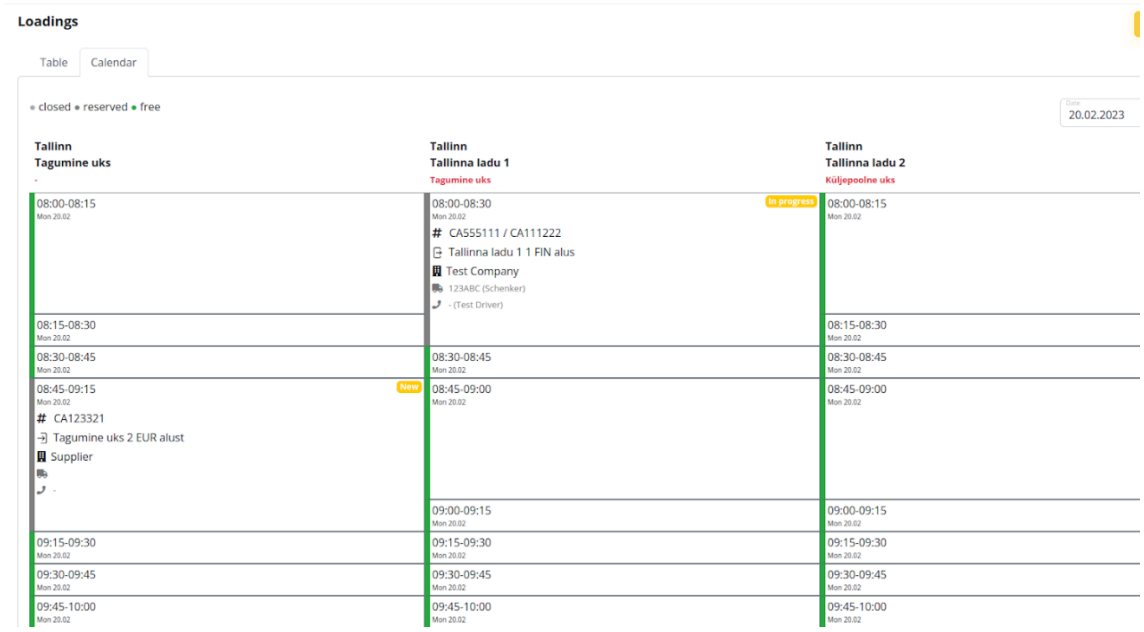
Vajadus täiesti uue laadimiskalendri rakenduse järgi tekkis ideest, et seda saaks kasutada ka ilma Cargosoni veohaldustarkvarata. On kliente, kes näevad ühes neist toodetest väärtust, aga teises mitte. Selleks on oluline, et neid saaks kasutada nii koos, kui ka eraldi. Hetkel on laadimiskalendri prototüüp veohaldustarkvara osa ning neid eraldi kasutada ei saa. Seda peab autor ka selle prototüübi suurimaks nõrkuseks.

Teiseks on tagasisidest klientidelt selgunud, et laadimiskalendri haldamiseks on vaja erinevaid rolle, mida praegune prototüüp ei võimalda. Näiteks soovivad ettevõtted anda ligipääsu oma laadimiskalendrile ka veofirmale. Põhjus peitub selles, et tihti klient ise ei tea, millal kaup saabub lattu ning veofirma peab sellest klienti teavitama. Vältimaks liigset manuaalset tööd, on veofirmal vaja võimalust tekitada kliendi kalendrisse laadimisi ise ning neid ka vajadusel muuta.

SaaS tüüpi tarkvara kasutajatele on oluline efektiivsus ning kasutajamugavus. Paljud tarkvara ettevõtte jaoks kriitilised näitajad nagu kasutajate rahulolu ning maksvate klientide teenuse kasutamise lõpetamise protsent on seotud kasutusmugavusega [9]. Kasutajamugavus on koht, mida uue rakenduse arendamisel saab parandada.

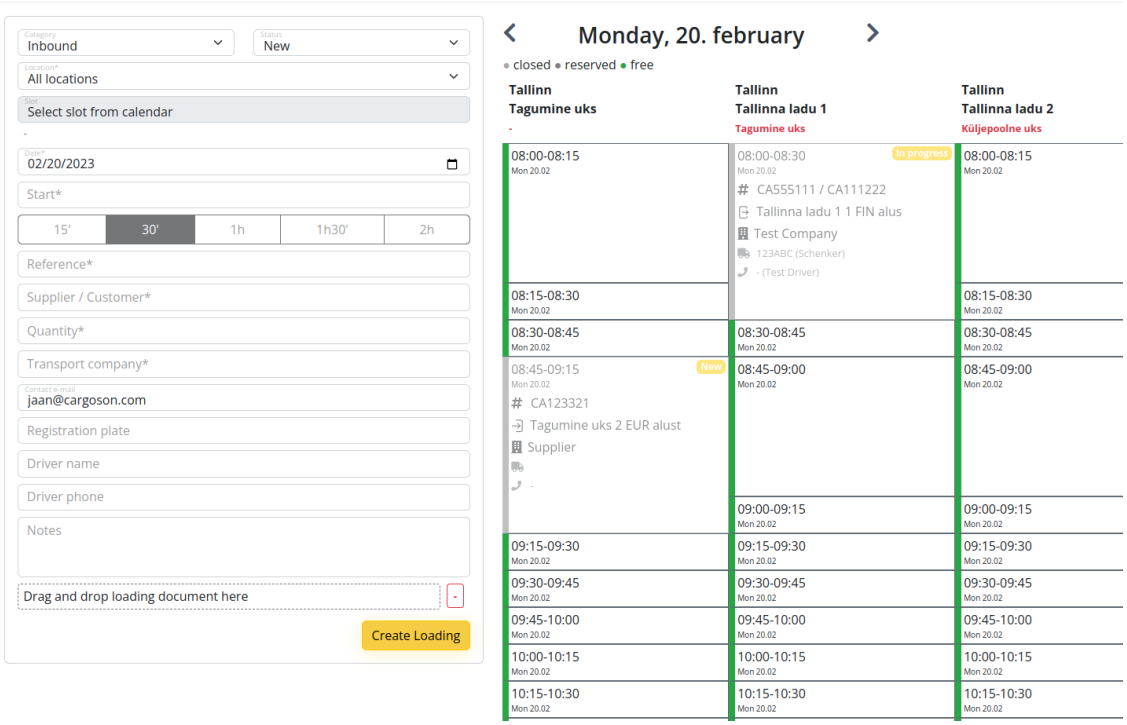
3.1.3 Prototüübi analüüs

Järgnevalt analüüsib autor prototüübi kasutajaliidese kaht peamist vaadet, mida kliendid iga päev kasutavad, ning nende kasutamise mugavust.



Joonis 2 Cargosoni laadimiskalendri prototüübi kalendri vaade

Kalendri vaade on antud tarkvaras kliendi jaoks tema peamine töölaud. Siin on näha lao tänase päeva kõik sisse- ja väljaminevad kaubad. Antud vaates on kalendri struktuurist raske aru saada ning kellaaja järgi otsimine nõuab süvenemist. Kasutajaliides peab olema kliendi jaoks intuitiivne ning ei tohi panna teda mõtlema, kuidas midagi teha saab [10]. Samuti on kogu laadimisega seotud info ühesuguse teksti suuruse ja paksusega, mille tõttu on konkreetse otsitava info leidmine raskem. Vabade aegade blokkidel on igaühel suure tekstiga kellaajad, mis tegelikult on teisejärguline informatsioon kasutaja jaoks. Teksti suurus ja silmapaistvus peaksid väljendama info olulisust [10].



Joonis 3 Cargosoni laadimiskalendri prototüübi laadimise lisamise vaade

Uue laadimise lisamisel suunab Cargosoni prototüüp uuele lehele, kus saab täita info laadimiseks vajalike andmetega (Joonis 3). Kõrval on näha ka selle päeva kalendri vaade, et aega parem planeerida oleks. Siin tekitab segadust laadimisala valik. Kui kõik muud andmed saab täita vasakul olevas jaotises, siis laadimisala ja aeg tuleb valida klikates paremal vabale aja blokile. Autori arvates saab seda protsessi lihtsustada. Klient võiks saada juba algses kalendri vaates (Joonis 2) klikata vabale ajale ning selle põhjal saab uue laadimise lehel ära täita info laadimisala ja kellaja kohta. Kalendri kuvamine kahes eraldi vaates on autori arvates ebavajalik ja tekitab infomüra.

3.2 Alternatiivsed lahendused

Tänu logistika valdkonna digitaliseerimisele on turul mitu laadimiskalendri SaaS tüüpi rakendust. Need on otsesteks konkurentideks antud töö tulemusena valmivale rakendusele. Autor analüüsib selles peatükis kahte alternatiivset lahendust - GoRamp [11] ning Opendock [12]. Analüüsi eesmärgiks on kaardistada nende funktsionaalsused ja tuua välja antud töö käigus valmiva laadimiskalendri erinevus konkurentidest.

Järgnevas tabelis (Tabel 1) on välja toodud peamised nõrkused ja tugevused mõlema rakenduse kohta.

Tabel 1 Alternatiivsed rakendused ja nende funktsionaalsused

	GoRamp	**Opendock**
Iseseisev registreerumine	Ei	Ei
Laadimiste lisamine	Jah	Jah
Laadimisalade seadistamine	Jah	Jah
Erinevad rollid	Jah	Jah
Emaili teavitused	Jah	Jah
Võimalus integreerida veohaldustarkvaraga	Ei	Jah

Autori hinnangul on mõlemal rakendusel olemas kõik klientide jaoks vajalikud funktsionaalsused. On võimalik seadistada laadimisaladel vabasid aegu, hallata laadimisi, anda ligipääs broneeringute tegemiseks ettevõtte välistele kasutajatele ning saata välja emaili teel teavitusi. Lisaks on Opendockil tugevaks plussiks integratsiooni võimalus Loadsmart nimelise veohaldustarkvaraga.

Mõlema rakenduse nõrkusena näeb autor iseseisvalt registreerumise võimalust. Kuna iga kliendiga tegeletakse personaalselt ning pannakse konto püsti, siis peab ka toote maksumus selle lisa ressursi võrra kallim olema. Siin näeb autor kohta, kuidas antud töö käigus valmiv laadimiskalender saab erineda turul olevatest konkurentidest. Ise registreerumise ja konto seadistamise jaoks peab olema rakenduse kasutamine intuitiivne ning piisavalt lihtne. Lihtsus ja kasutajamugavus on teine koht, millega uus rakendus saab erineda turul olevatest konkurentidest.

3.3 Rakenduse nõuded

Rakenduse funktsionaalsete ja mittefunktsionaalsete nõuete väljatöötamiseks viis autor läbi kohtumise Cargoson OÜ töötajatega, kellel on pikaajaline kogemus logistika valdkonnas. Lisaks võttis autor arvesse tehtud analüüsi olemasolevale prototüübile ning turul olevatele alternatiividele.

Uue rakenduse peamiseks erinevuseks Cargosoni prototüübiga saab olema lisa roll kasutajatele nimega “kolmas osapool”. Antud rolliga kasutajad saavad laadimiskalendrisse broneerida aegu, kuid ei näe teiste laadimiste kohta infot. Sellega lahendatakse ära olukord kui veofirma soovib ise laadimiseks aja määrata. Tal peavad olema näha vabad ajad, kuid ta ei tohi saada näha muud infot lao töö kohta. Lisaks paneb autor rõhku parema kasutajaliidese loomisele, mis teeb kliendi jaoks igapäevast tööprotsessi mugavamaks.

Kuna turul juba saadaolevate lahenduste funktsionaalsused on head ning klientide vajadusi rahuldavad, siis eelise loomiseks peab töö käigus valmiv rakendus võimaldama ise registreerumise võimalust. See teeb kogu protsessi potentsiaalse kliendi jaoks läbipaistvamaks, kiiremaks ja lihtsamaks. Järgnevalt loetleb autor ette kõik rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded.

Funktsionaalsed nõuded rakendusele:

- Klient peab saama ise registreerida enda ettevõtte konto.
    - Ettevõtte konto juurde peab saama emaili teel kutsuda teisi kasutajaid.
    - Iga kasutaja võib kuuluda mitme ettevõtte juurde.
- Kasutaja peab saama näha, lisada, muuta ja kustutada laadimisalasid.
    - Laadimisaladel peab olema võimalik määrata lahtiolekuaegu.
- Kasutaja peab saama lisada, muuta ja kustutada laadimisi.
    - Laadimisi peab saama näha kalendri vaates.
    - Laadimised peavad olema visuaalselt kergesti eristuvad olenevalt nende staatusest (uus, pooleli või valmis).
- Laadimise lisamisel peab välja minema emaili kinnitus laadimise autorile ning vedajale.
- Kasutaja peab saama ettevõttel sisse-välja lülitada emaili teavitusi.
- Kasutajatel peab olema võimalik rolliks määrata “kolmas osapool”.
    - “Kolmas osapool” rolliga kasutajad tohivad näha ainult kalendrivaadet.
    - “Kolmas osapool” rolliga kasutajad peavad saama lisada laadimisi ning näha/muuta/kustutada ainult enda tehtud laadimisi.
    - “Kolmas osapool” rolliga kasutajad peavad nägema laole laadimiseks reserveeritud aegasid, kuid mitte infot nende kohta.

Mittefunktsionaalsed nõuded rakendusele:

- Kood peab jälgima puhta koodi põhimõtteid.
- Rakendus peab olema ehitatud selliselt, et seda saaks hiljem lihtsalt integreerida Cargosoni veohaldustarkvaraga.
- Äriliselt olulised funktsionaalsused peavad olema testitud.

4 Tulemus

Selles peatükis antakse ülevaade valminud rakenduse arhitektuurist ning valminud kasutajaliidesest. Rakenduse valmimisel on lähtutud eelnevalt määratud nõuetest ning ka Cargoson OÜ töötajate tagasisidest. Kasutajaliidese puhul on lisaks vaadetele kirjeldatud ka kasutajakogemuse parandamiseks tehtud otsuseid.

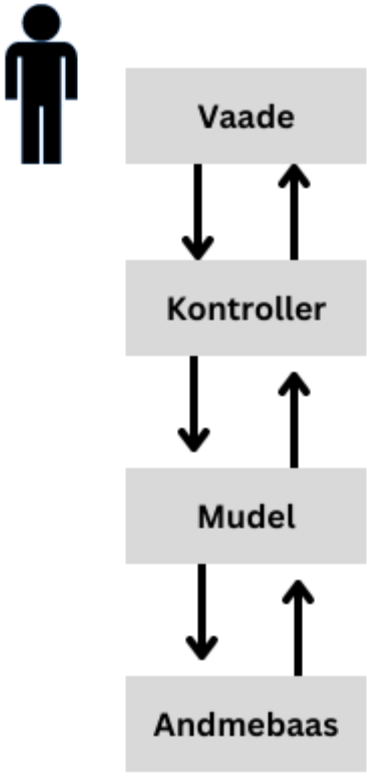
4.1 Arhitektuur

Arenduseks kasutatud Ruby on Rails raamistik jälgib põhimõtet *convention over configuration* [13]. See tähendab, et arenduse lihtsustamiseks on paljud tehnilised detailid raamistiku poolt paika pandud. Kiireks ja efektiivseks arendamiseks on mõistlik järgida raamistiku kasutamise parimaid praktikaid. Rakenduse arhitektuur on seetõttu valitud Ruby on Railsi sisse ehitatud arhitektuuri järgi, milleks on MVC muster.

Mudeli kihis on ära kirjeldatud enamus äriloogikast ning paika pandud validatsioonid andmeobjektidele. Näiteks kontrollib mudeli kiht, et kaks laadimist ei kattuks omavahel ajaliselt. Samuti valideerib mudeli kiht, et kaks kasutajat ei saaks sama emailiga olla.

Kontrolleri kihis küsitakse andmebaasist andmed ning valmistatakse need ette vaate jaoks. Lisaks kontrollitakse, et kasutaja oleks sisse logitud ning tal oleks õigus küsituid andmeid näha, muuta või kustutada. Kui kasutaja üritab mõne teise ettevõtte andmetele ligi pääseda, siis kontrolleri ülesanne on kasutaja ümber suunata ja teda teavitada ligipääsu õiguste puudumisest.

Vaates äriloogika puudub ning kasutajale kuvatakse kontrolleris ette valmistatud andmed. Kasutajakogemuse parandamiseks on osadel lehtedel kasutatud JavaScripti. Näiteks on kalendri vaates JavaScriptiga töötav filter, mis võimaldab kasutajal kiirelt liikuda erinevate kuupäevade vahel ilma, et peaks terve lehekülje uuesti laadima.



Joonis 4 MVC muster

Joonisel 4 on välja toodud, kuidas toimub rakenduses suhtlus kasutaja ja andmebaasi vahel jälgides MVC mustrit.

4.2 Kasutajaliides

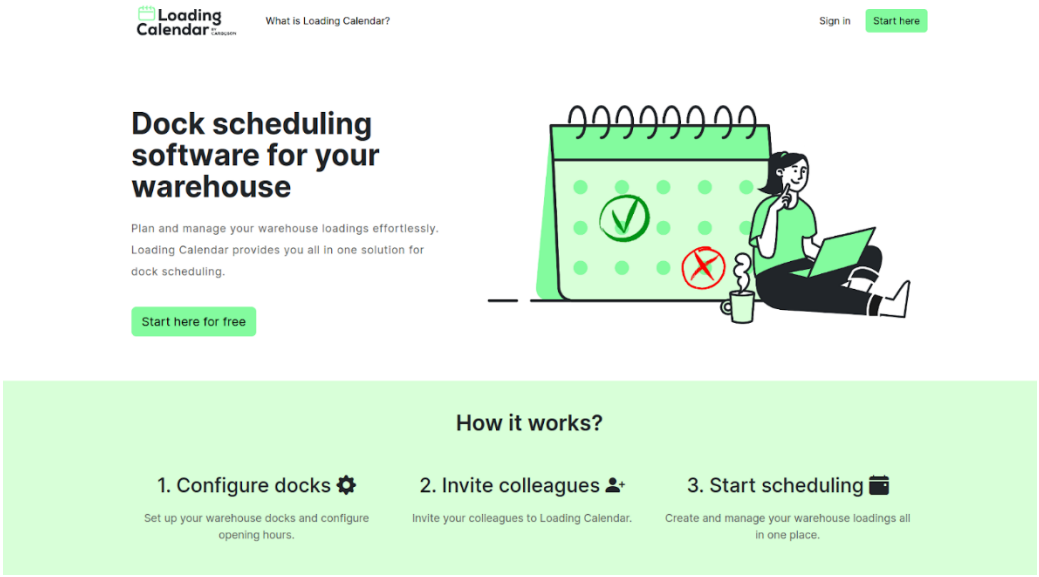
Rakenduse kasutajaliidese arendamisel lähtus autor rakendusele seatud nõuetest ning kasutajate vajadustest. Selles peatükis on välja toodud tähtsamad vaated ning selgitatud autori tehtud disainilahendusi nende vaadete puhul. Vaated on jagatud kahte kategooriasse olenevalt sellest, kas kasutaja on sisselogitud või mitte.

Kasutajaliidese prototüübi loomisel lähtuti UX ja UI disaini põhimõtetest, et süsteem oleks kasutajale võimalikult mugav ja lihtne. UX disain keskendub kasutajakogemusele: disaini loomisel püütakse vaadelda kõiki toote või teenusega seotud aspekte läbi kasutaja erinevate tajumismeelte ning luua disain selliselt, et see oleks eesmärgipärane, kättesaadav, usaldusväärne ja käepärane. UI disain keskendub kasutajaliidesele ning määrab süsteemi struktuuri, visuaalse ilme, vormid ja visuaalse stiili. UI disaini peamised

komponendid on visuaalne disain ja selle süsteemid korduvkasutatavate elementide loomiseks ning prototüüp. Kasutajaliidese prototüüp on loodud viisil, et selle kasutamine oleks piisavalt intuitiivne ning loogiline ka ilma täiendava juhendamiseta [14].

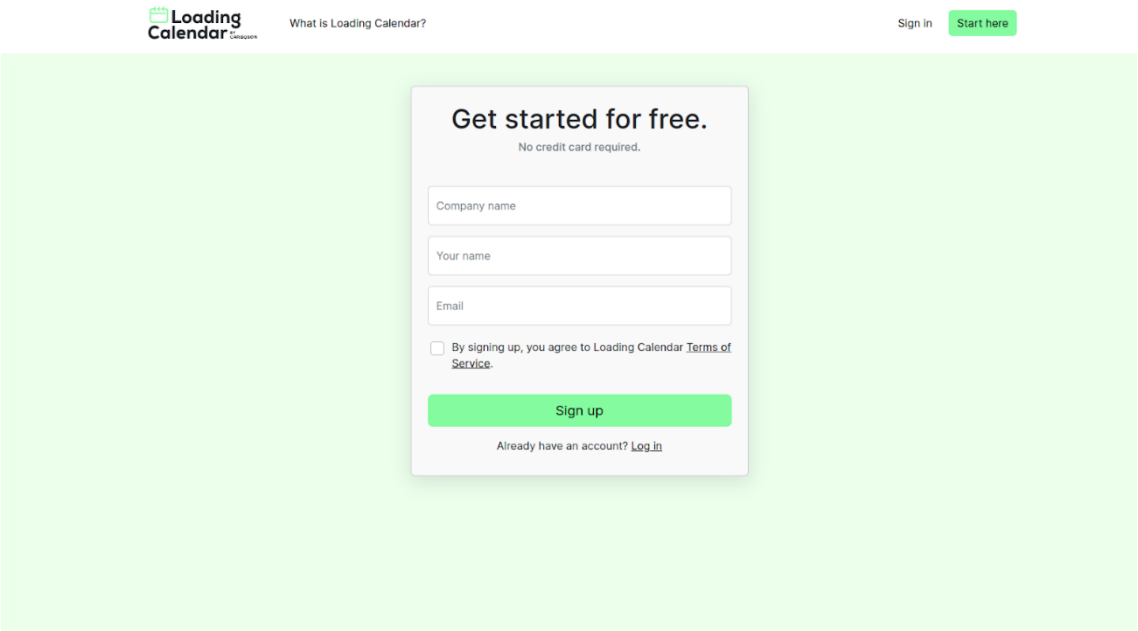
4.2.1 Sisse logimata kasutaja vaated

Valminud rakendus on toode, mis on mõeldud kasutamiseks kõikidele potentsiaalsetele klientidele, mitte vaid ühele ettevõttele. Ettevõteteni jõudmiseks on oluline informatiivne maandumisleht (ingl k "*landing page*") ehk leht, kuhu uued potentsiaalsed kasutajad esimesena jõuavad veebilehte külastades [15].



Joonis 5 Maandumisleht

Joonisel 5 on näha valminud rakenduse maandumisleht. Lehel on olemas suur pealkiri, kus on selgelt sõnastatud, mis rakendusega on tegu. SEO koha pealt on tähtis, et rakenduse ühelauseline tutvustus oleks HTML koodis h1 elemendi sees ning h1 elemente oleks maandumislehel ainult üks [16]. Selleks, et lehe külastajaid kutsuda registreerima ning rakendust kasutama, on nii lehe keskosas kui ka üleval navigatsiooniribal silmatorkavad nupud, mis viivad otse registreerimise lehele.

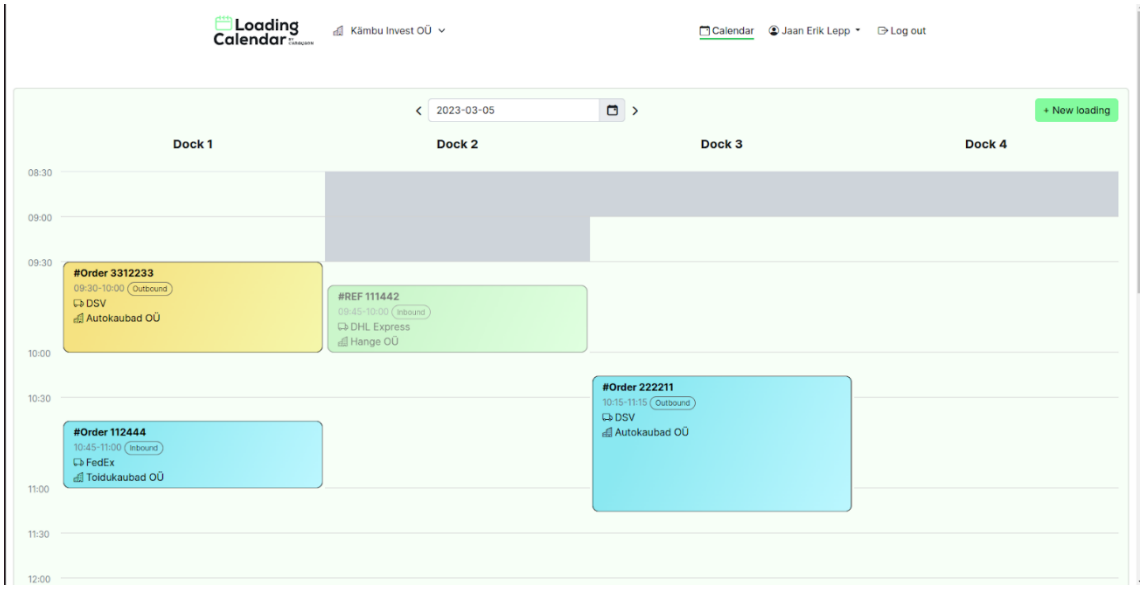


Joonis 6 Konto registreerimisleht

Joonisel 6 on näha kasutajate registreerimisleht. Seal tuleb sisestada ettevõtte nimi, enda nimi ning email. Peale registreerumist saadetakse kasutajale emaili valideerimiseks kinnituskiri. Vajutades kinnituskirjas olevale lingile, saab kasutaja valida endale parooli ning logida sisse. Emaili kinnitamine on vajalik, et vältida petturitel registreerimast emailiga, mis ei kuulu neile.

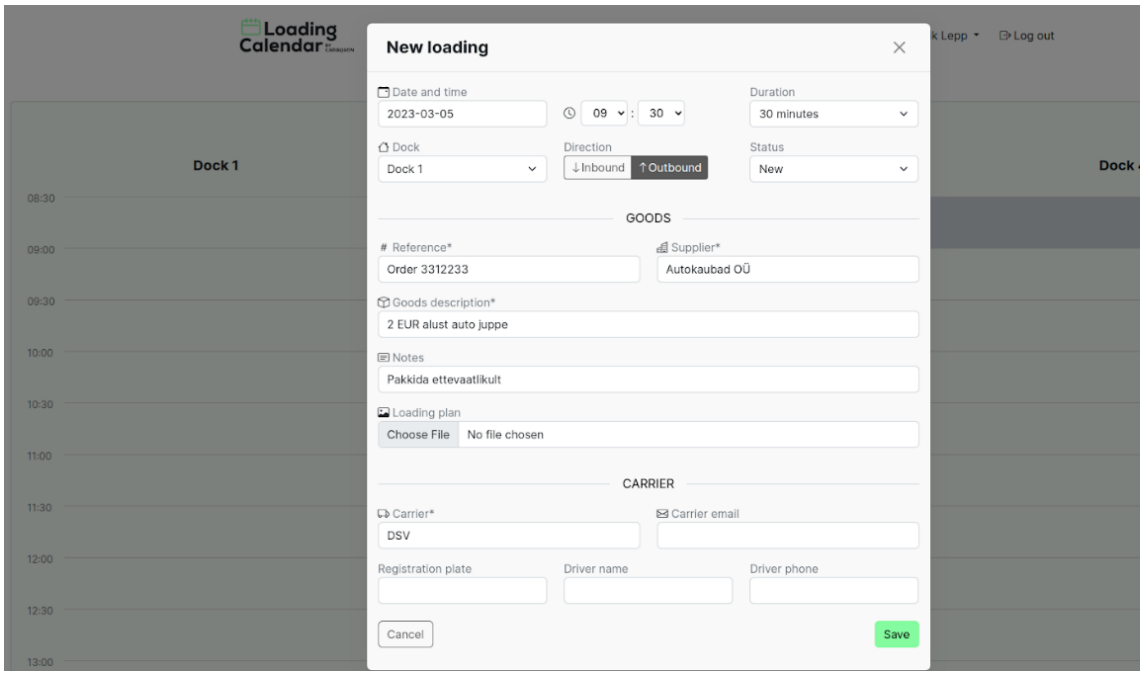
4.2.2 Sisselogitud kasutaja vaated

Sisselogitud kasutaja vaadete disainimisel on autor lähtunud sellest, kui palju mingit funktsionaalsust kasutatakse. Antud rakenduse puhul on kahtlemata kõige olulisem laadimiste kalendrivaade, mis on kasutajate igapäevane töölaud. Laadimisalade seadistamine ning uute kasutajate kutsumine on pigem teisejärgulised ja harva külastatavad vaated, mida on vaja peamiselt konto esialgsel seadistamisel.



Joonis 7 Kalendrivaade

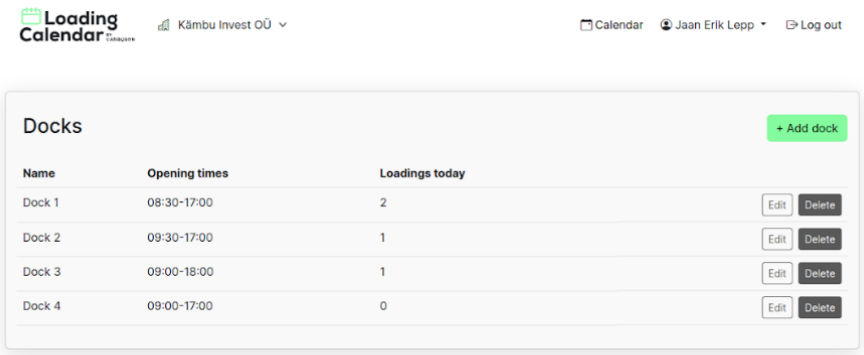
Joonisel 7 on näha rakenduse kalendrivaade, mis avaneb kasutajale kohe peale sisselogimist. Laadimised on näha ühe päeva kohta ning kalendri ülaosast saab vahetada kuupäeva. Parema ülevaate tagamiseks on laadimised staatuse järgi värvikodeeritud. Sinine tähendab alustamata, kollane pooleliolevat ning roheline lõpetatud staatuses laadimist. Laadimiste puhul on olulise infona veel esiletõstetud kauba viide või number, mille järgi seda laos eristatakse. Lisaks on näha iga laadimise kohta, kes on vedaja ning kellele see kaup kuulub ehk kellelt kaup tuleb või kellele läheb. Liigse infomüra vältimiseks on kalendris välja toodud ainult kõige olulisem info ning lisainfot näeb vajutades laadimise peale.



Joonis 8 Laadimise lisamise vaade

Joonisel 8 on näha uue laadimise lisamise vorm. Sama vormi kasutatakse ka laadimise muutmisel. Kasutajal on uue laadimise lisamiseks kaks võimalust. Esimene võimalus on kalendrivaates vajutada nupule “New loading” ning siis avaneb kasutajale tühi vorm dialoogiaknas. Teine võimalus on kasutajal valida kalendris vaba aeg ning vajutada selle peale. Seejärel avaneb dialoogiaknas vorm, kus kuupäev, kellaaeg ja laadimisala on vastavalt ette täidetud kasutaja tehtud kliki põhjal.

Kuna uute laadimiste lisamine on üks peamiseid funktsionaalsuseid, siis on väljade automaatne ette täitmine oluline samm hea kasutajakogemuse loomiseks. Samuti on kalendrivaates sellisel moel laadimiste lisamine kasutajate jaoks intuitiivne.

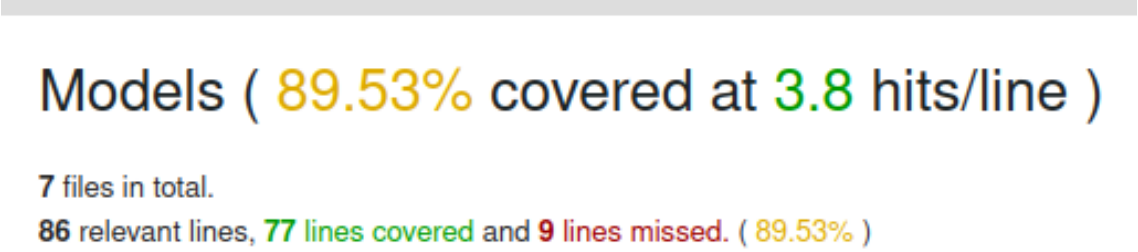


Joonis 9 Laadimisalade loetelu vaade

Joonisel 9 on laadimisalade koondvaade. Tabelis on näha kõik ettevõtte seadistatud laadimisalad, nende lahtiolekuajad ning laadimiste arv igaühes tänase päeva jooksul. Laadimiste arv aitab kasutajal omada ülevaadet erinevate laadimisalade töökoormuse kohta ning vajadusel teha korrektuure, näiteks tõsta laadimisi ümber ühelt alalt teisele. Laadimisalasid on võimalik lisada, muuta ning kustutada. Lisamise vormil tuleb määrata laadimisala nimetus ning lahtioleku kellaajad. Lahtioleku kellaaeg on vaikimisi ette määratud kella 9-st kella 5-ni.

4.3 Testid

Rakenduse üheks mittefunktsionalseks nõudeks oli äriliselt oluliste funktsionaalsuste testimine. Selle jaoks kirjutas autor mudelite kihile ühiktestid, mis katavad ära 89% mudelite koodist. Kuna mudeli kihis asuvad objektidele seatud piirangud, validatsioonid ning suur osa äriloogikast, siis oli oluline teste eelkõige just sellele koodi osale kirjutada. Mudelitele kirjutati kokku 30 ühiktesti.



Joonis 10 Mudelite testidega kattuvus

Joonisel 10 on näha, et mudelite kihis on 86 rida koodi ning testidega on sellest kaetud 77 rida. Samuti on joonisel välja toodud, et iga rida mudelite kihi koodi on keskmiselt testitud 3,8 korda.

5 Analüüs

Antud peatükis kirjeldab autor rakenduse tehnilist teostust ning põhjendab tehtud valikuid, analüüsib vastavust nõuetele ning toob välja edasised plaanid ja võimalikud edasiarendused. Analüüsi käigus selgub, kas rakendus vastab ootustele ning milline on eelis turul olevate alternatiivide ja Cargosoni prototüübi ees.

5.1 Tehniline teostus

Autor valis arenduskeskkonnaks just Ruby on Railsi, sest tal on seal kõige pikaajalisem kogemus. Juba tuttavas raamistikus on võimalik kogemuse omamisel kirjutada kvaliteetsemat koodi ning rakenduse arendus toimub kiiremini kui täiesti uue raamistikuga. Samuti mängis raamistiku valimisel rolli see, et Ruby on Rails on olnud esimene valik paljude tuntud veebirakenduste puhul nagu näiteks Github, Shopify, Airbnb ja Spotify [17].

Rakenduse arhitektuuri loomisel kasutas autor MVC mudelit. Ruby on Rails raamistik on loodud selle mustri kasutamiseks ning paljud seadistused tehakse arendaja eest seeläbi automaatselt ära. Seetõttu järgis autor arendamise lihtsustamiseks ning kiirendamiseks MVC mustrit.

5.2 Rakenduse vastavus nõuetele

Valminud rakendus täidab kõik sellele seatud funktsionaalsed ning mittefunktsionaalsed nõuded. Olemas on peamised funktsionaalsused, et saaks rakenduse reaalselt kasutusele võtta. Selleks, et kliendid oleks valmis ka maksma antud tarkvara eest, tuleb veel arendada sellele lisandväärtust.

Tagasiside Cargoson OÜ meeskonnaliikmetelt oli positiivne. Rakenduses on võimalik seadistada enda laadimisalad, hallata laadimisi ning info edastatakse automaatselt osapooltele emaili kaudu. Lisaks on olemas „kolmanda osapoole“ roll, tänu millele saab lahendada ka keerulisemad klientide vajadused. Tarkvara müümise osas annab see eelise, kui on võimalik lahendada rohkemate klientide probleeme. Esialgsed nõudmised

rakendusele on täidetud ning edasised nõudmised tekivad juba testides tarkvara klientide peal.

Lisaks Cargosoni meeskonnale sai autor tagasisidet ka potentsiaalselt kliendilt, kes tulevikus laadimiskalendri tarkvara sooviks kasutama hakata. Suurimaks probleemiks hetkel on integratsiooni puudumine Cargosoni veohaldustarkvaraga. Kliendil on ebamugav omada kasutajaid nii veohaldustarkvaras kui ka laadimiskalendri tarkvaras. Lisaks tuleb igapäevaseid tööülesandeid täita kahes eraldi aknas ja seeläbi dubleerida osa informatsioonist. Ettepanekuid oli ka kalendri vaate kohta. Esiteks peaks laadimisi saama filtreerida laadimisalade põhjal. Teiseks võiks olla võimalik vaadata tänaseid laadimisi ülevalt alla jooksva tulbana, mitte kalendrina. Seeläbi saab omada paremat ülevaadet, mis on juba tehtud ja mis tegemata.

5.3 Autori edasised plaanid

Autor näeb valdkonnas suurt potentsiaali tehnoloogiliseks arenguks ning kavatseb jätkata rakenduse edasiarendamist. Rakendusel on piisavalt funktsionaalsuseid, et see saaks aidata ettevõtteid laohalduse paremal korraldamisel. Järgmise sammuna on eesmärk leida esimesed ettevõtted, kes oleks nõus hakkama antud rakendust testima. Reaalses elus kasutamine ning tagasiside saamine on kõige parem viis ka tarkvara edasi arendada ning lahendada probleeme, mis ettevõtetel reaalselt olemas on.

Rakenduse arendamisel on järgmine oluline samm integratsiooni loomine Cargosoni veohaldustarkvaraga. Klientide jaoks annab suurt lisandväärtust, kui peale transporditellimuse tegemist saab samas keskkonnas broneerida vedajale kaubale lattu järgi tulemiseks aja. Kui info liigub kahe tarkvara vahel automaatselt, on info kvaliteet parem ning kasutajatel lihtsam omada täielikku ülevaadet oma ettevõtte logistikast. Samuti parandab info automaatne liikumine kasutajakogemust mõlemas tarkvaras.

Lisaks integratsioonile veohaldustarkvaraga, on autoril veel ideid rakenduse edasiarendamiseks. Näiteks oleks kasutajatel vajalik näha logi päeva jooksul tehtud muudatustest, mida praegu võimalik näha pole. Kui päeva jooksul on broneeritud laadimiseks uus aeg, siis üllatuste vältimiseks peaks see logis kajastuma. Kasutajamugavuse parandamiseks peaks saama kalendrivaates laadimisi filtreerida

vastavalt staatusele, laadimisalale ning autorile. Mõlemad muudatused pole kriitilise tähtsusega rakenduse kasutuselevõtmisel.

6 Kokkuvõte

Tootmisettevõtetel puudub tihti efektiivne viis enda lao sissetulevate ja väljaminevate kaupade haldamiseks. Bakalaureusetöö eesmärk oli arendada Cargoson OÜ-le rakendus, mis aitaks seda probleemi lahendada.

Rakenduse nõuete väljaselgitamiseks analüüsis autor Cargosoni eksisteerivat laohaldustarkvara prototüüpi ning turul pakutavaid alternatiive. Arenduse käigus arvestas autor lisaks seatud nõuetele ka Cargoson OÜ töötajate tagasisidet valminud funktsionaalsustele.

Rakendus valmis Ruby on Rails raamistikus ning täitis kõik sellele seatud funktsionaalsed ja mittefunktsionaalsed nõuded. Kasutajal on võimalik rakenduses seadistada enda laadimisalad ning hallata laadimisi. Lisaks liigub laadimistega seonduv info kõigile osapooltele meili teel.

Antud rakendusega täideti bakalaureusetöö käigus püstitatud eesmärk. Tagasiside Cargoson OÜ töötajatelt oli positiivne ning rakendusel on olemas minimaalsed funktsionaalsused selle kasutamisele võtmiseks. Autor jätkab rakenduse arendamist ning järgmisena on plaan integreerida laohaldustarkvara Cargosoni veohaldustarkvaraga. Lisaks sellele on plaanis leida rakenduse testimiseks esimene ettevõte, kellega koostöös välja selgitada edasised vajalikud arendused.

Kasutatud kirjandus

[1] Cargoson [Võrgumaterjal]. Available: www.cargoson.com. [Kasutatud 25.02.2023].

[2] GoRamp, “What is dock scheduling?”, GoRamp [Võrgumaterjal]. Available: https://www.goramp.com/blog/what-is-dock-scheduling. [Kasutatud 01.05.2023].

[3] S. Miller, “What is Ruby on Rails?”, Codecademy [Võrgumaterjal]. Available: https://www.codecademy.com/resources/blog/what-is-ruby-on-rails. [Kasutatud 01.05.2023].

[4] AWS, “What Is Javascript (JS)?”, AWS [Võrgumaterjal]. Available: https://aws.amazon.com/what-is/javascript. [Kasutatud 01.05.2023].

[5] Codecademy, “MVC: Model, View, Controller”, Codecademy [Võrgumaterjal]. Available: https://www.codecademy.com/article/mvc. [Kasutatud 01.05.2023].

[6] R. C. Martin, C. M. Feathers, T. R. Ottinger ja J. J. Langr, Clean code A handbook of agile software craftsmanship. Boston: Pearson Education, 2016.

[7] Atlassian, “What is agile?”, Atlassian [Võrgumaterjal]. Available: https://www.atlassian.com/agile. [Kasutatud 01.05.2023].

[8] E.Ries, The Lean Startup. Great Britaion: Penguin Business, 2011.

[9] Ramotion, “How to Improve SaaS UX Design? Examples and Best Practices”, Ramotion [Võrgumaterjal]. Available: https://www.ramotion.com/blog/saas-ux-design. [Kasutatud 19.02.2023]

[10] S.Krug, Don’t Make Me Think!: A Common Sense Approach to Web Usability, Second Edition. Berkeley: New Riders Publishing, 2005.

[11] GoRamp [Võrgumaterjal]. Available: www. goramp.eu. [Kasutatud 25.02.2023].

[12] Opendock [Võrgumaterjal]. Available: www.opendock.com. [Kasutatud 25.02.2023].

[13] S. Cynixit, “Overview of Ruby on Rails Architecture” [Võrgumatterjal]. Available: https://medium.com/@SravanCynixit/overview-of-ruby-on-rails-architecture-9902de7c93f9 [Kasutatud 19.04.2023].

[14] E. Canziba, Hands-On UX Design for Developers, Birminham: Packt, 2018

[15] Mailchimp, “Landing page”, Mailchimp [Võrgumaterjal]. Available: https://mailchimp.com/marketing-glossary/landing-pages. [Kasutatud 07.03.2023].

[16] A. Hardwick “What is an H1 Tag? SEO Best Practices” [Võrgumaterjal]. Available: https://ahrefs.com/blog/h1-tag. [Kasutatud 07.03.2023].

[17] Rails [Võrgumaterjal]. Available: https://rubyonrails.org. [Kasutatud 18.03.2023].

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

Mina, Jaan Erik Lepp

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Laohaldustarkvara planeerimine ning arendamine Cargoson OÜ-le“, mille juhendaja on Karl-Erik Karu.
   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.

16.05.2023

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.