TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Risto Põldsalu

KOORMA KOOSTAMISE VEEBIRAKENDUS

bakalaureusetöö

Juhendaja: Marko Kääramees
PhD

Tallinn 2017

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: Risto Põldsalu

22.05.2017

Annotatsioon

Töö eesmärgiks on uurida ettevõtte vajadusi ja nõudeid veoautode koormate koostamise rakenduse jaoks ning luua prototüüp, analüüsida rakendusega kaetavat protsessi ning leida võimalikud tehnoloogiad lahenduse loomiseks. Analüüsi põhjal luuakse veebirakendus, millega ettevõte saab graafilise kasutajaliidesega koormaid koostada. Samuti saab neid koormaid muuta ja hallata ning koorma koostamiseks vajalike andmete lisamist ja muutmist.

Töö tulemuseks on veebipõhine rakendus, mis on veebipõhine ja millega on võimalik planeerida erinevaid koormaid vastavalt koorma sisule.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 24 leheküljel, 5 peatükki, 10 joonist.

Abstract

Cargo packing web application

The aim of this thesis is to analyze a companies needs for cargo management and for the necessity of a program to put together plans for trucks and then develop an application with a graphical interface that can fufill the requirements found in the analyzis.

The solution is made as a web application that works in all computers without installing anything. It allows users to plan cargos by adding different kind of packages to the graph and allowing the user to manipulate all the pacakges comfortably. Already planned cargos are saved and can be modified if necessary. Also the packages that are added to the cargo are changeable and modifiable and allows the user to add different size of packages.

The graphical interface is done as a 2D planner. Because of the dimensions of the package the company mainly uses, both sides of the truck can be packed separately. That allows the program not to take into account the depth dimension which in turn is why it is possible to only use two dimensions.

The application is programmerd to be easy to use and not overwhelm the user with too many possibilities and paramateres.

The thesis is in Estonian and contains 24 pages of text, 5 chapters, 10 figures.

Lühendite ja mõistete sõnastik

ATI	TTÜ Arvutitehnika instituut
JSON	*JavaScript Object Notation* ehk Javascripti objekti esitlusviis [3]
WebGL	*Web Graphics Library* [5]
HTML	*HyperText Markup Language*
CAD	*Computer-aided design* ehk raalprojekteerimine
BaaS	*Backend-as-a-Service*
CLI	*Command Line Interface* ehk käsurea liides

Sisukord

Jooniste loetelu

1 Sissejuhatus

Teemaks on koorma koostamise rakendus veoautodele. See on planeeritud täitma tühimikku, kus praegu tehakse kõike käsitsi kasutades CAD tööriistu ja luues iga kord uuesti vajalikud komponendid ja pakid. Puudub arusaadav ja lihtne ülevaade koormast. Rakendus hakkab olema tööriistaks, mis laseb koormat koostada ja eemaldab tülikad kõrvaltegemised nagu iga paki eraldi joonistamine.

1.1 Taust ja probleem

Ettevõte peab koostama veoautode jaoks koormaid kauba laiali vedamiseks. Koormate paigutuse jaoks on vaja teha joonis, mille järgi pakid laaditakse autode peale. Ettevõttel puudub sobiv rakendus sellise joonise koostamiseks. Kasutusel on CAD joonestustarkvara, mida on võimelised kasutama ainult CAD programmide kasutusoskusega töötajad ning vajab aega erinevate pakkide joonestamiseks ja kõikide vajalike elementide tegemiseks ja kopeerimiseks. Ettevõte on aiamaju valmistav ettevõte, kelle toodangust enamus liigub Eestist väljapoole.

Joonist saab koostada kahemõõtmeliselt, sest pakkide paigutus muudab veoauto küljed eraldi aladeks. Parimaks paigutuseks on pakid tehtud selliselt, et neid mahuks auto peale sügavuse poolest kaks tükki. Kuna pakid on tavaliselt pikad ja piklikud sisu pärast, siis on modifitseeritud pakkide teisi mõõtmeid nii, et neid oleks mugav transportida ja sobiksid kõige paremini autode peale. Seetõttu on välja kujunenud külje pealt laadimine ja see kaotab ära vajaduse sügavuse mõõtme jälgimise koormate koostamisel. Samuti on sellepärast ühe koorma koostamise vaatel kaks autokasti, mis on tegelikult ühe auto kaks poolt.

Puudub sobiv rakendus, mis aitaks koostada lihtsalt ja kergelt koormaid veoautodele kauba laiali vedamiseks ja oleks sobiv sellele ettevõttele.

Käesoleva töö autor on olnud seotud ettevõttega ning töötanud seal. Nähes probleemi soovis ta leida sellele lahendust, et aidata ettevõtte töötajaid teha oma tööd lihtsamalt ja kiiremini. Samuti huvitab autorit optimeerimine ja arvuti võimekus teha arvutamisi

inimesest kiiremini ja tõhusamalt. Selle rakendusega saab ta aidata osa töökoormust kanda üle masinale.

1.2 Püstitus

Ülesandeks on uurida ettevõtte vajadusi ja vastavalt analüüsile prototüüpida lahendus veebirakendusena koorma koostamiseks graafilise liidesega. Leida sobivaim lahendus ettevõtte probleemi lahendamiseks kasutades tehnoloogiaid, mis annavad lõppkasutajale hea kasutajakogemuse. Lahendus peab olema tarkvara poolelt lihtne ja efektiivne.

1.3 Metoodika

Eesmärkide saavutamiseks uuritakse ettevõtte vajadusi ja rakenduse nõudeid ning võimalikke lahendusi. Võrreldakse olemasolevaid tehnoloogiaid ja lahendusi ning valitakse sobivaimad vahendid rakenduse loomiseks. Peale seda arendatakse prototüüp. Prototüüp luuakse eeldusega, et seda oleks võimalik täiendada vastavalt uute nõuete tekkimisele või olemasolevate nõuete muutustel. Tulemuste kontrolliks kasutatakse testimist ja ettevõtte vajaduste ja nõuete täitmise kontrollimist.

1.4 Ülevaade tööst

Esimeses osas uuritakse ettevõtte vajadusi ning olemasolevaid lahendusi ja koostatakse algne analüüs koos rakenduse eesmärkidega. Uuritakse võimalikke tehnoloogiaid ja valitakse neist sobivaimad ettevõtte eesmärke ja rakedused nõudeid arvestades. Siis arendatakse prototüüp, mis rahuldab kliendi soove ja mis on töötav variant rakendusest, mida saab kasutada ettevõttes. Lõpuks arutatakse edasistest võimalikest arenduse suunadest, mis tekitaksid rakendusele lisaväärtust ja lisafunktsionaalsust juurde, kuid ei mahtunud käesolevasse skoopi.

2 Analüüs

Käesolevas peatükis uuritakse olemasolevaid lahendusi ja räägitakse rakenduse analüüsist. Uuritakse mis on nõuded arendusele ning kuidas lahendada ettevõtte probleemi, et täita ettevõtte vajadusi.

2.1 Olemasolevad lahendused

Analüüsi raames uuriti internetist ettevõtete kodulehekülgedelt missuguseid lahendusi ja rakendusi pakutakse. Uuriti lahendusi, mis pakuvad pakkimise planeerimise toodet kas eraldiseisva rakendusena või teenusena internetis. Leiti kahte erinevat tüüpi rakendusi. Allalaetavad ja kohalikus arvutis jooksvad rakendused ning veebipõhised serveritel jooksad lahendused. Täpsemalt tuuakse siin mõlemast tüübist üks ja selgitatakse nende funktsionaalsust ja omadusi:

- EasyCargo3D [1] on veebipõhine rakendus, mis lubab koostada koormaid 3D graafikul. Pakke saab genereerida koormale korraga, lisades pakile parameetrid. Koorma koostamise arvutamine toimub brauseris. Printimiseks on võimalik teha pilti vabalt valitud nurga alt ja ka koormast seest, et oleks võimalik näha asetust. Hoiab koorma pakkide loendi ja kuva ühel ekraanil, et ei peaks vahetama erinevate akende vahel ja kogu info oleks ühes aknas kohe olemas. Igale pakile on võimalik lisada tingimusi, mis mõjutavad selle paigutamist koormale. Samuti on võimalik pakke paigutada sihtkoha järgi, mis annab tulemuse, kus ühe sihtkoha pakid on lähestikku. Koorma kuva on interaktiivne ja lubab pakke liigutada ja pöörata vastavalt vajadusele. On võimalik muuta veoauto ja koorma üldiseid mõõtmeid ja lisada virtuaalseid vaheseinu. Lubab lisada kaalu pakkidele, et jälgida telgede koormust.

- LoadXpert'i Load Planning [2] on allalaetav tarkvara mis pakub automaatset koorma koostamise planeerijat. Lubab erinevaid tüüpe ja erinevate kujudega pakke. Laseb salvestada andmebaasi erinevaid pakkide tüüpe, et neid koormates kasutada. Võimalik on valida erinevate koorma kujude ja transpordi vahendite

vahel. Koorma plaanil oskab tarkvara automaatselt lisada pehmendusi või täiteid ning lisada kinnitusi. Lubab mitme veoauto täitmist kui kogused on suuremad ja vaja on mitut koormat. Võimaldab erinevate suurustega pakkide lisamist koormale. Kuvab nimekirja praegu koormal olevate pakkide kohta. Lubab välja printimisel koostada nii 2D kui ka 3D graafikuid koos lisadetailide ja muu koorma kohta käiva infoga. Koostab koormaid automaatselt ja pakub ühe koorma paigutuse jaoks erinevaid variante. On võimalus panna lisaparameetreid, mis mõjutavad süsteemi planeerimist, lubades leida sobivaima koorma planeeringu.

Pakutavad lahendused on 3D graafilisel liidesel põhinevad programmid, mis on hästi välja arendatud. On võimalused koostada väga erinevaid koormaid koos suure hulga erinevate parameetritega. Muuta saab pakkide kui, asetust ja asukohta koormas.

Allalaaditavate rakenduste eeliseks on võimalus paremini kasutada ära süsteemi ressursse. Samuti lubavad juba hästi välja arendatud keskkonnad rohkemaid võimalusi automatiseerimiseks ja kasutada ära programmeerimiskeelte eeldusi, et süsteem oleks kiirem ja sujuvam.

Allalaaditavate rakenduste puuduseks on see, pakutavad lahendused peab installima arvutisse, mis piirab kasutatavust märgatavalt, sest teises arvutis kasutamiseks peab tarkvara jälle paigaldama. Tarkvara on ehitatud toetama kindlaid operatsioonisüsteeme ja vajab kindlat tarkvara ja draiverite olemasolu installitavas arvutis.

Veebirakenduste eeliseks on mobiilsus. Rakendust on võimalik kasutada palju rohkematel süsteemidel, sest vajavad ainult brauserit töötamiseks. Samas on ka võimelised kasutama ära süsteemi ressursse järjest enam nii kuidas brauserid arenevad. Internetis olevate andmete pluss on geograafilises mõttes kättesaadavuse kohta väga suur pluss kuna rakendust on võimalik kasutada igal pool kus on ühendus ning see lubab jälgida koormaid kõigil.

Veebirakenduste puuduseks on vajadus pideva interneti järele. Rakendused suhtlevad pidevalt serveriga andmete saamiseks ja ühenduse kadumisel hangub ka rakenduse töö ja rakendus muutub kasutuskõlbmatuks. Brauserite võimekuse puuduse pärast jäävad osad funktsionaalsused ja visuaalsed elemendid tegemata brauserite piiratuse tõttu ja brauseris olevate tehnoloogiate puuduste tõttu.

Käesoleva ülesande jaoks sobib veebirakendus ning selle puudused ei ole takistuseks rakendusele. Rakenduse peamine kasutuskoht on kontorites, kus on pidev ja hea ühendus internetiga ning rakenduse funktsionaalsuse vajadused katab brauseri võimekus täielikult. Rakenduse mõte on teha see kasutajale võimalikult lihtne ja et kasutajaliides ei sisaldaks üleliigset infot ja funktsionaalsust. Pole olemas sellist rakendust, mis kasutaks ainult 2D lahendust, mis paneb küll piirangud rakenduse võimekusele, aga teeb selle kasutamise palju lihtsamaks. Sellepärast tehakse uus rakendus, mille eesmärgiks on olla võimalikult lihtne kasutajale kasutada ja ei omaks liigseid mõõtmeid ja keerukust.

2.2 Rakenduse nõuded

Selles peatükis pannakse kirja rakendusele kehtivad nõuded, mis peavad olema realiseeritud prototüübis. Nõuded on jaotatud kaheks: funktsionaalsed nõuded ja mittefunktsionaalsed nõuded. Funktsionaalsed nõuded on seotud rakenduse äriloogika ja funktsionaalsuse poolega. Mittefunktsionaalsed nõuded on rakenduse arhitektuur ja käideldavus ning kasutatavus.

2.2.1 Ettevõtte vajadused

Ettevõtte vajadusteks on koormate koostamise võimalus. Selleks on vaja graafilise liidesega rakendust. Ettevõttel on vaja salvestada koostatud koormate kohta andmed ja võimalust neid hallata ja hilisemalt uuesti kasutada. Koorma koostamiseks on vajadus kasutada erinevaid pakitüüpe mille tõttu on vaja hoida meeles eraldi pakitüüpe ja sihtkohti pakkide jaoks. Neid peab saama hallata, kui pakid muutuvad või võetakse neid kasutusest ära.

Ettevõtte töötajatel peab olema võimalus andmetele ligi pääseda igal hetkel. Vajadus on näha koormaid ja nimekirju, et kõik töötajad oleksid teadlikud missugust koormat tuleb teha.

2.2.2 Rakenduse funktsionaalsed nõuded

Nõuded koostati vastavalt vajadustes kirja pandule.

Rakenduse funktsionaalsed nõuded:

1. Graafilisele liidesele genereeritakse veoautode koormate külgvaated 2D-s õigetes proportsioonides kastidena.

2. Kasutaja saab interaktiivselt liigutada ja manipuleerida graafilisel liidesel olevaid pakke.
3. Kasutaja saab lisada ja kustutada graafiliselt liideselt pakke.
4. Rakenduses on võimalik lisada koormale autosid.
5. Koormalt on võimalik kustutada autode vaateid.
6. Võimalus muuta graafilisel liidesel kuvatavat koorma autot.
7. Rakenduses on vaade pakitüüpide ja asukohtade nimekirjadega.
8. Süsteemis on võimalik lisada pakitüüpe koos mõõtmete ja värviga.
9. Sisestatud pakitüüpide andmeid on võimalik muuta.
10. Pakitüüpe on võimalik süsteemist kustutada.
11. Süsteemis on võimalik lisada asukohti koos koha nime, aadressi ja värviga.
12. Asukohti on võimalik süsteemist kustutada.
13. Sisestatud asukohtade andmeid on võimalik muuta.
14. Koormaid on võimalik süsteemist kustutada.
15. Rakenduses on vaade varasemalt koostatud koormate nimekirja kuvamisega.
16. Rakenduses on vaade koorma detailandmete vaatamiseks.
17. Varasemalt koostatud koormaid on võimalik uuesti laadida graafilisele liidesele muutmiseks.
18. Uuesti laetud koormaid on võimalik täiendada ja salvestada.
19. Uuesti laetud koormaid on võimalik salvestada uute koormatena.
20. Rakendus võimaldab koostada printimiseks graafilisest liidesest pildi.
21. Rakendus salvestab koorma autodest pildid.

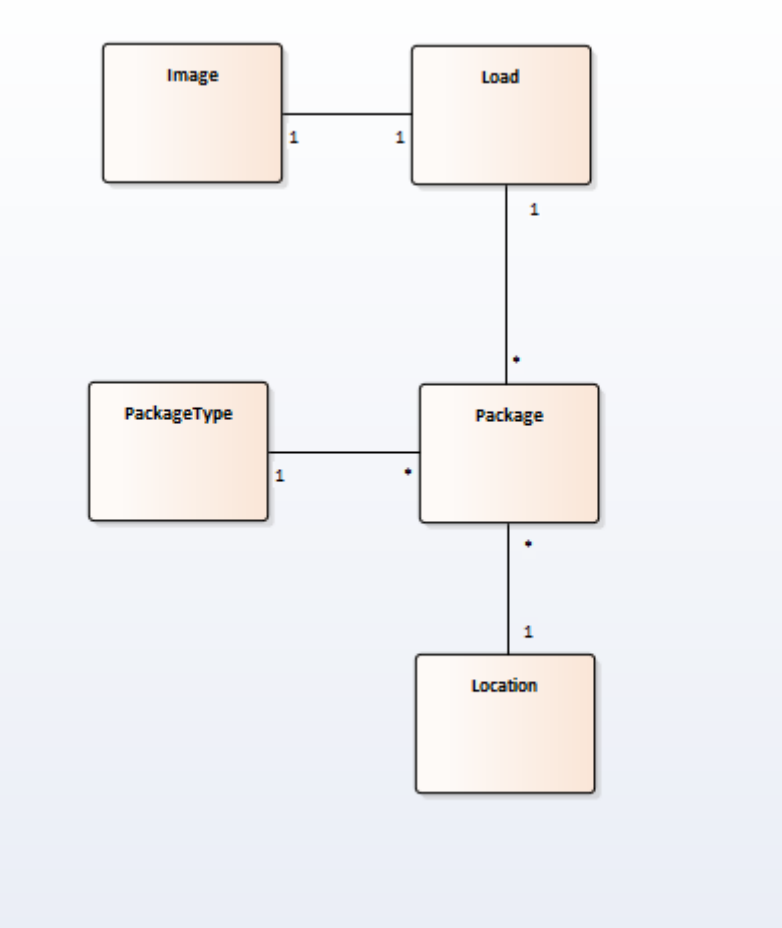
2.2.3 Rakenduse mittefunktsionaalsed nõuded

Rakenduse mittefunktsionaalsed nõuded:

1. Süsteemi kood on inglisekeelne.
2. Kasutajaliides on mugav ja intuitiivne.
3. Rakendust on võimalik kasutada brauserites, mis toetavad HTML5
4. Süsteem töötab veebilehitsejatega Google Chrome, Mozilla Firefox ja Microsoft Edge uusimate versioonidega.
5. Rakendus majutatakse veebis.

2.3 Süsteemi objektid

Põhilisteks objektideks süsteemis on koorem, pakk, pakitüüp, asukoht ja pilt. Joonisel nr.1 on näidatud ära nende seosed.



Joonis 1 Põhiolemid ja nende seosed

2.4 Rakenduse struktuur

Rakendus tehakse veebilehena, millel on kolm lehte. Osa funktsionaalsust on lahendatud modaalidga nendel kolmel lehel. Esimene leht sisaldab peamist rakenduse eesmärki ehk graafilist kuva. Eesmärk on lasta kasutajal kohe koormat koostama hakata ilma lisategevusi tegemata. Graafilise liidese jaoks on eraldi menüü, kus on nimekiri võimalikest tegevustest. Kaks teist lehte on andmete haldamiseks ja lubavad koorma ja pakkidega seotud andmeid muuta.

Rakenduse kolm vaadet:

- Esimene vaade on graafilise liidese vaade. See on rakenduse peavaade, kus on graafiline liides. Rakendust avades on kohe võimalik alustada koorma koostamist.
- Teine vaade on ajaloo vaade. See kuvab nimekirja loodud koormatest ja laseb neid hallata ja võtta uuesti muutmisesse esimesele vaatele. Ühe koorma detailide nägemiseks avaneb modaal.
- Kolmas vaade on pakkide ja asukohtade vaade. See on samuti abistav vaade, kus on nimekirjad erinevatest pakitüüpidest ja asukohtadest, mida kasutades on võimalik pakke koostada. Nagu ajaloo vaates, on ka kolmandas vaates võimalik kõiki pakke ja asukohti hallata. Selleks avaneb esemele nimekirjas peale vajutades modaal muudetavade väljadega.

3 Realiseerimine ja tulemused

Selles peatükis selgitab autor veebirakenduse ehitust, tehnilist teostust ja valitud tehnoloogiaid.

3.1 Kasutatavad vahendid ja tehnoloogia

Eesmärk oli luua rakendus võimalikult õhukesena. Sellepärast on rakenduse pakkumine veebis üks parimaid variante, sest kõik brauserid toetavad seda ja puudub vajadus rakendust eraldi installida.

Arenduseks kasutatakse WebStorm'i, millega saab teha veebirakendusi ja on optimiseeritud Javascripti kasutamiseks. Samuti on see autorile tuttav keskkond.

3.1.1 Javascript

Javascript [3] on skriptimiskeel, mida kasutatakse peamiselt veebilehtedele lihtsa funktsionaalsuse lisamiseks kuid on oma ehitus poolelt võimeline väga komplekse funktsionaalsuse loomisel. Seda kasutatakse väga tihti koos HTML ja CSS-ga veebilehtedes. Javascript on prototüübipõhine objektorienteeritud keel, mis on süntaksilt sarnane Java-le kuid erinedes omaduste poolest päris palju. Samuti on ta teiste keeltega võrreldes omapärane. Ühe asjana puuduvad tal standardsed vahendid sisendiks ja väljundiks.

Vaadati erinevaid võimalusi graafilise liidese lahendamiseks:

3.1.2 HTML Canvas

HTML Canvas [4] on lisatud HTML-i koos HTML5-ga. Kasutatakse skripti teel veebilehele joonistamiseks Javascripti läbi. Canvas on element mis on osa veebilehest ja mille peale on võimalik joonestada. Canvas omab kõrguse ja laiuse parameetreid. See on bitmap element, mis ei sisalda omaette objekte ja pildil olevad elemendid ei ole otseselt aktiivsed ja interaktiivsed. Seda juhitakse läbi Javascripti. Sellega on võimalik joonistada veebilehele jooniseid, pilte, graafikuid ja kasutada seda rakendustes osana

kasutajaliidesest. Puuduseks jääb Canvasel 3D toe puudumine. Samas on seda võimalik implementeerida Canvase peale ehitatud teiste teekidega nagu WebGL.

WebGL [5] on HTMLi ja Javascripti põhjal 3D graafika renderdamiseks Javascripti API. Kasutatav veebis ilma igasuguste lisa pluginateta ja juhitav Javascripti poolt. Esimene versioon tuli välja 2011. See on toetatud enamustes brauserites, aga vajab ka graafikakaardi toetust, mis piirab kasutatavust. WebGL kasutab HTML Canvast ning on võimeline lahendama väga keerulisi graafilisi kuvasid 3Ds. Selles projektis ei läinud see kasutusse, sest toetuse vähesuse tõttu ei ole see veel nii laialt igale poole levinud ning rakenduse nõuetest tulenevalt ei ole praegu selle funktsionaalsuse vajadust.

Vaadates rakenduse nõudeid on aru saada, et praeguste nõuete juures sobib HTML Canvase kasutamine 2D kontekstis ja see täidab graafilise liidese vajadused. Samuti on Canvast lihtne juhtida läbi Javascripti, mis lubab terves rakenduses kasutada ühist keelt ning läbi selle hoida andmete liikumine rakenduses võimalikult lihtne ja arusaadav.

3.1.3 Majutus ja andmebaas

Andmebaase uurides võeti aluseks baasi koodiga ühendamise lihtsust. Tahtmine oli leida lahendus, mis ei vajaks serveri poolse koodi kirjutamist rakenduses endas. Selleks vaadati platvorme, mis võimaldavad andmebaasiga ühendust frontendist kasutaja seadmest. Üritades püsida lihtsuse juures ja otsides võimalusi lasta valmis süsteemidel võimalikult palju tööd ära teha, jäid valikusse Backend-as-a-Service majutajad, kes pakuvad ühtset komplekti nii andmebaasist kui ka rakenduse majutusest. Valikutes olid Stamplay, Firebase, Syncano ja Backand, kes kõik pakuvad sarnast toodet. Valikuks langes Firebase, sest see on madalate kuludega ja eksisteerib eelnev kogemus platvormiga. Samuti on see laialt kasutatud, mis tähendab head kommuuni probleemide lahenduste leidmiseks ning see katab ära käesoleva rakenduse vajaduses.

Firebase Database [6] on andmebaasina kasutatud NoSql andmebaas. See on reaalaja andmebaas, mis on pilvepõhine ja kus andmed salvestatakse JSON formaadis. Baasi muutuste korral sünkroniseeritakse andmed iga ühendunud kliendiga. Tavaliste HTTP päringute asemel kasutab Firebase andmete sünkroniseerimist, kus iga kord kui andmed muutuvad, saavad selle uuenduse kohe kõik seadmed. Kui rakendus peaks kaotama võrguühenduse, saab jätkata tööd, sest Firebase Realtime SDK salvesta andmed kõvakettale, hoides nii rakenduse töötavana. Ühenduse taastumisel saadetakse kõik

uuendused ja andmed sünkroniseeritakse. Andmebaasiga ühendumiseks ei ole vajalik teha seda läbi serveri vaid on võimalik otse kliendi seadmest. Seetõttu puudub vajadus eraldi andmete töötluseks serveri poolt. Samuti on selle kaudu võimalik ka turve ja andmete valideerimine. Andmebaasile pääseb ligi läbi Firebase Console veebilehe, mis lubab ülevaadet selle kasutaja loodud andmebaasidest ning laseb muuta andmebaasi seadeid. Samuti on võimalik näha andmebaasi andmeid, millest tuuakse siia ka näide Joonis 2, kus on näha baasi puust ühte osa.

Joonis 2 Andmebaasi puu struktuur näiteandmetega

Firebase hosting [7] on BaaS lahendus ehk kõik ühes server ja andmebaas, kus serveri poole eest hoolitseb Firebase. Firebase majutus pakub staatiliste veebilehtede majutamist, kus veebileht on HTML, CSS ja Javascripti ning teistest failidest koosnev rakendus, mille

failid ei muutu dünaamiliselt. Firebase on turvaline ja ilma mingisuguse konfigureerimiseta saavad kõik lehed külge SSL sertifikaadi, mis tagab turvalise andmete edastuse. Kogu info ja leht majutatakse SSD-del, mis võimaldab kiiret vastust päringutele. Väga lihtne ja kiire on platvormile deploymine kasutades Firebase CLI-d. Käsurea tööriistad lubavad rakenduse üles laadida ühe käsuga. Samuti pakub leht versioniseerimist ja on võimalik teha roll-backe ühe klõpsuga.

3.1.4 Frontend

Frontendi raamistikuks valiti Angular.

Angular [8] on struktuurne raamistik, mis on loodud dünaamiliste veebirakenduste jaoks. Antud raamistik võimaldab kasutada HTML malli ja laiendada HTML-i süntaksit, mis väljendab rakenduse komponente selgelt ja lühidalt. Kõik toimub veebilehitseja sees, mis teeb Angularist ideaalse partneri iga võimaliku serveritehnoloogiaga. Angular aitab struktueerida rakendust, pakkudes kontrollerite ja teenuste ehitamise võimalust ja lastes neid kergelt omavahel ühendada ja kasutada. Angulariga on võimalik siduda kasutajale kuvatavad elemendid otse Javascripti mudeli külge. See lubab muuta andmeid lehel dünaamiliselt ja lihtsalt. Uuendades mudelis andmeid, kuvatakse need ka kohe kasutajale.

Rakenduses kasutatakse iga lehe jaoks oma kontrollerit ning andmetüüpidel on omad mudelid ja teenused andmete käsitlemiseks.

3.2 Rakenduse arhitektuur

Süsteemi arhitektuur on mõeldud olema võimalikult lihtne ja kergelt hallatav. Seda silmas pidades tehakse rakendus veebirakendusena. Kasutades Firebase'i on võimalik teha rakendus ilma serveripoolse lisakoodita ning ilma andmebaasi üles sättimise ja täpsustamiseta.

Rakendus on ehitatud kihilise arhitektuuriga, kus on kolm kihti. Firebase Hosting ja Firebase Database lubavad rakendust teha nii, et kirjutatakse ainult Front-endi ja võimaldab andmebaasi ühenduse lahendada kasutaja seadmes:

- Controller ehk kontrollerite kiht: see on seotud vaadetega ja omavad skoopi, kus seotakse andmed veebilehel kujutatavaga ning vahendab kasutaja tegevusi.

- Service ehk teenusekiht: teenused tegelevad andmete käsitlemise ja töötlemisega ja baasi suhtlusega.

- Server: Majutus on Node serveri peal ja kuna andmebaasiga ühendumine toimub front-endis, siis puudub ka kood serveri poolsest back-endist, aga see on ikkagi osa rakendusest ja sellepärast ka siin välja toodud.

Klassid on jaotatud package-by-layer meetodiga, kus ühes paketis on vastavalt ühe kihi klassid. Firebase Hostingu tõttu puudub rakendusel praeguse funktsionaalsuse juures vajadus serveripoolseks äriloogikaks, kuigi Firebase võimaldab luua Firebase funktsioone, mis on sisuliselt back-end kood rakendusele.

3.3 Tehniliste probleemide lahendamine

Selles peatükis seletab autor lahti rakenduse arendusega seotud põhilisemad kohad ja üldised kasutused ning nende lahendused.

3.3.1 HTML Canvas interaktiivseks muutmine

Kuna HTML Canvas ise on ainult bitmap, siis tuleb seda juhtida Javascripti abil. Luuakse selline funktsionaalsus, võttes põhjaks olemasoleva näite [10], kus on loodud funktsionaalsus elementide liigutamiseks objektidena.

Interaktiivseks muutmiseks on hiire alla vajutamisel vaja aru saada kas hiire alla jääb mõni objekt. Kõiki joonistatud elemente Canvasel hoitakse nimekirjas ja hiire vajutusel käiakse see nimekiri läbi ja leitakse element millele vajutati. Vastavalt edasisele tegevusele hakatakse objekti liigutama koos hiirega või kustutatakse objekt Objekti kustutamiseks on tema nurgas eraldatud ala, millele on joonistatud rist. Seda vajutades kustutatakse objekt.

Tuvastamiseks, kas hiir vajutas objektile, omavad kõik objektid X ja Y koordinaati, mille järgi on nad joonistatud graafikule. See punkt on objekti keskpunkt ja objekt ise on joonistatud selle ümber vastavalt paki tüübi mõõtmetele. Et märkida liigutatav objekt teistest ettepoole, et see graafikul oleks kõige peal peale selle liigutamis, muudetakse selle asukohta Javascripti nimekirjas ja pannakse see kõige lõppu. Siis joonistatakse kõik objektid uuesti joonisele. Objektid joonistatakse järjekorras. Sellepärast on nimekirjas

tagapool olevad objektid joonisel pealpool. Objekti kustutamisel kustutatakse ta ka nimekirjast.

Tuuakse välja koodinäide Package klassis olevast funtksioonist, mis kontrollib kas etteantud koordinaadid on objekti alas. Tõese vastuse puhul teab rakendus, et see objekt on hiire all.

```
Package.prototype.hitTest = function(hitX,hitY, proportions) {
    var pack = this.packageType;
    var mFactor = 2*1000;
    return((hitX > this.x - pack.width/mFactor*proportions)
        &&(hitX < this.x + pack.width/mFactor*proportions)
        &&(hitY > this.y - pack.height/mFactor*proportions)
        &&(hitY < this.y + pack.height/mFactor*proportions));
};
```

Edasi peab aru saama, kas kasutaja soovib teha lihtsalt hiire vajutust või liigutada hiire all olevat pakki. Vajutades hiire alla, pannakse käima intervalliga funktsioon, mis liigutab, uuendades vajutatud objekti positisiooni ja joonistab pilti nii kaua kuni hiir all on. Liigutamise ajal uuendatakse pilti piisavalt tihti, et paki liikumine oleks inimsilmale sujuv.

3.3.2 HTML Canvas pildi tegemine

Kuna üheks nõudeks on ka koorma koostamisel sellest pildi koostamine, siis kasutades Canvase enda funktsionaalsust on võimalik luua png formaadis pilt, mida on võimalik kohe välja printida. Koorma pilt salvestatakse baasi tekstistringina. Pilti on võimalik kohe peale koostamist sellisena nagu ta luuakse kuvada veebilehele.

3.3.3 Ühendamine Firebase andmebaasiga

Firebase Databse NoSql on JSON formaadis puu, kust saab andmeid kätte küsides mingi puu haru lapsi. Firebase lubab lisada nendele harudele kuulajaid, mis laste uuenemisel saadab andmebaasi poolt uued andmed ühendatud kuulajatega rakendustele. See laseb uuendada andmeid reaalajas kõikidele kasutajatele. Seetõttu pole vaja uuendada lehte kasutaja poole pealt.

Järgnevalt on koodinäide angulari koodist, mis on koormate harule lisatud kuulaja mis andmete uuenemisel uuendab globaalset koormate nimekirja.

```
rootRefLoca.on('value', function(snapshot) {
	loads.length = 0;
	angular.foreach(snapshot.val(), function (load,key) {
		loads.push(Load.build(load,key));
	});
});
```

Planeerides baasi, peab arvestama puu meetodi puudusi. Kui panna koorma alla kõik sellega seonduv nii, et selle haru sügavus järjest kasvab, siis soovides saada koorma andmeid, laetakse ära kõik selle koorma küljes olevad harud ja lapsed. See aeglustab süsteemi tööd ja on halb disain. Parem on jagada andmed laiali eraldi harude alla samamoodi nagu SQL andmebaasis tabelitena, kus info kätte saamiseks ei ole vaja minna puul sügavale. See annab väga palju tunda kui tehakse tingimustega päringuid ja otsingus on vaja minna mitu astet sügavake, et saada kätte andmed. See vajab kõikide nende andmete laadimist ja alles peale seda vaadata kas laps üldse vastab parameetritele.

Otse veebist andmebaasiga suhtlus on tehtud lihtsaks ja kompaktseks ning suunab keerulise ühenduse loogika Firebase Database osale ning aitab hoida rakenduse koodi kompaktse ja selgena. Lehe enda laadimine ei kannata andmete laadimise all, sest neid laetakse eraldi ja leht täidetakse andmetega kohe kui need baasist kohale on jõudnud tänu algularile. Lehe põhi on valmis ja funktsionaalne ning andmed lisatakse siis kui need on ära kohale jõudnud.

Andmete salvestamiseks võetakse jälle viide koormate harule ning lisatakse sinna uus laps või uuendatakse olemasolevat.

Järgnevalt näidatakse koodinäidet, kus funktsioon võtab sisse koorma ja kontrollib kas see on olemasolev või uus koorem ja salvestab selle baasi.

```
this.saveLoad = function (load) {
    var ref;
    if (load.key === null) {
        ref = loadRef.push();
        load.key = ref.key;
    } else {
        ref = loadRef.child(load.key);
    }
    var loadSave = angular.copy(load);
    ref.set(loadSave);
    return true;
};
```

Varem salvestatud koormatel on küljes *key* ehk identifikaator, mis on nende kirje kohanäitaja baasis. *Key* puudumisel lisatakse koormate harusse uus tühi koorma objekt. Selle objekti *key* lisatakse ka salvestatavale koormale ning salvestatakse koorem tühja objekti peale.

3.4 Teostus

Arendus toimus iteratiivsel meetodil. Et valideerida funktsionaalsuse võimalikkust valitud tehnoloogiatega, arendati esmalt rakenduse selgroog ja põhifunktsionaalsused. Seejärel arendati lisad ja ülejäänud nõuded.

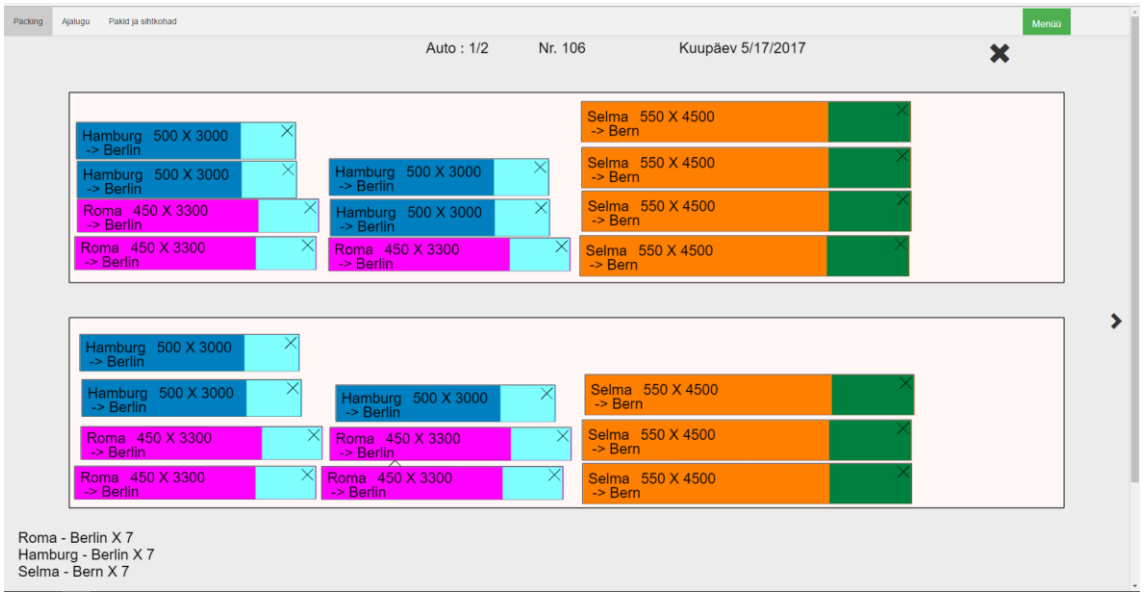
Funktsionaalsetest nõuetest lähtuvalt pandi kirja kasutusjuhud ja hakati neid realiseerima tähtsuse järjekorras, hinnates olulisemaks vajaduspõhiselt need funktsionaalsused, milleta rakendus ei saa töötada. Funktsionaalsust testiti ja sobivuse korral võeti ette järgmine osa.

Iga iteratsiooni valiti mingi osa funktsionaalsusest ja arendati see. Kui arenduse käigus selgus, et midagi ei ole võimalik teha nende vahenditega, siis muudeti ka vastavalt analüüsi. Kui see avaldas ka mõju olemasolevale koodile ja funktsionaalsusele, viidi need muudatused sisse igale poole enne edasi minemist. Sellega üritati hoida koodi ühtsust ja selgust, et kood oleks käsitletav.

Kõik kirja pandud nõuded said täidetud. Nõuetest jäeti välja need osad, mis on pandud kirja peatükis „4 Edasised arengusuunad“. Järgnevalt seletatakse lahti rakenduse vaated ja nendega seotud nõuete järgi seletatakse lahti rakenduse töötamine.

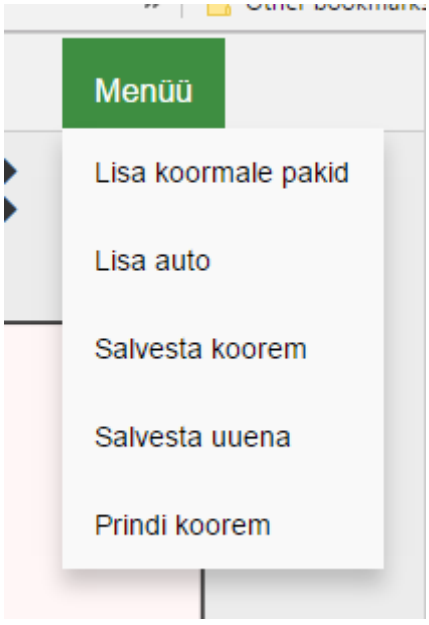
3.4.1 Graafilise liidese vaade

Rakenduse peamisel kuval on graafiline liides koormaga. Samuti on seal menüü koormaga seotud tegevusteks, kust saab lisada pakke ja autosid ning salvestada koormat.



Joonis 3 Graafilise liidese vaade

Graafilisel liidesel on peamised andmed koorma enda ja selle auto pakkide kohta, mis tulevad ka koorma printimisel, et saaks ülevaadet kiiresti ning jälgida missugune koorem on pildil ja missugused pakid seal täpsemalt on. Järgnevalt on lisatud kuva graafilisest liidesest.

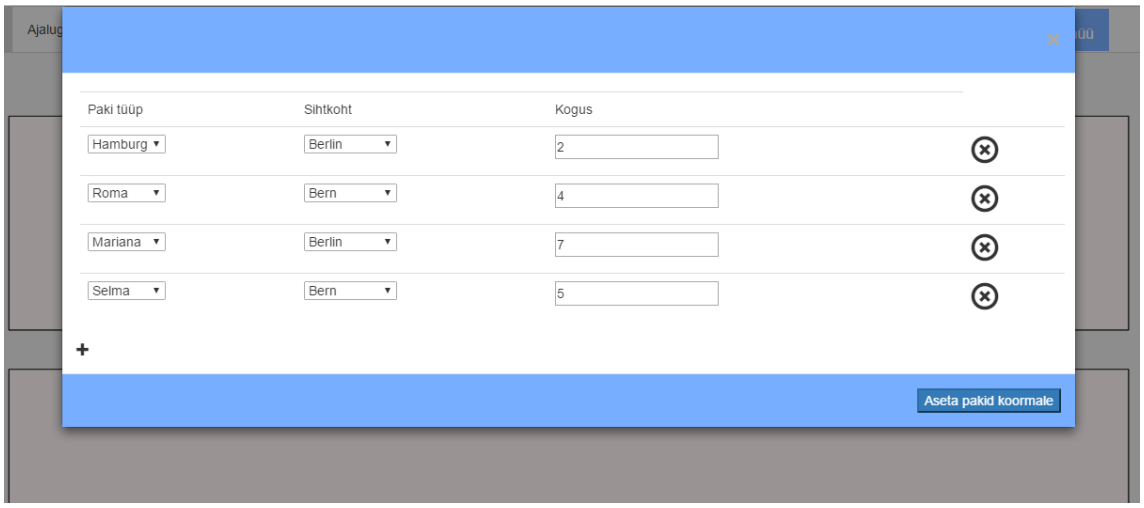


Joonis 4 Dropdown menüü graafilise liidese kasutamise jaoks

Graafiline liides on hoitud võimalikult lihtsa ja müravabana, et jääks võimalikult palju ruumi koormate jaoks ja rakendusele peale vaadates oleks kohe selge, mis rakenduses toimub ja mis on võimalik teha. Autode suurused on võetud euroopa standardite järgi [9] ning on pakkidega võrreldes reaalsete proportsioonide ja mõõtudega. Praegu kasutab ettevõtte ainult ühte sorti autosid ja puudus vajadus lisada koorma suuruse muutmine rakendusse.

Lehe avamisel luuakse uus koorma objekt, mis võtab andmebaasist viimase lisatud koorma järgi omale järgmise numbri. Samal ajal laetakse baasist pakkide tüübid, asukohad ja koormate ajalugu. Kui lisada koormale pakke, siis tüüpide ja asukohtade valikus olevad nimekirjad on samad, mis on pakkide ja asukohtade vaates nimekirjades.

Lisades koormale pakke, kasutatakse pakkide genereerimiseks abistavat andmeobjekti, mis on pakkide lisamisel vaatel iga rea kohta eraldi objekt. Lisades uue rea, tehakse uus objekt, mis kuvatakse lisamise listi. See objekt koosneb kasutaja valikutest ja soovitud pakkide kogusest ning selle põhjal genereeritakse pakid.



Joonis 5 Pakkide lisamise modaalaken

Pakke hoitakse Canvase teenuses eraldi nimekirjas. Pakkide genereerimisel pannakse pakkidele külge auto number, millele see genereeriti. Graafilisele liidesele kuvatakse pakk vastavalt pakitüübi ning asukoha värvile. Domineeriv värv on paki tüübi värv ja väiksem osa on asukoha värv. See teeb kergeks ja silmaga nähtavaks ühe sihtkoha pakid ja samade mõõtmetega pakid. Paki peal on veel pakitüübi nimi, mõõtmed millimeeter ühikutes ja sihtkoha nimi. Samuti on pakil selle kustutamiseks paremas nurgas rist, mis eemaldab paki vaatelt.

Joonis 6 Paki kujutis graafilisel liidesel

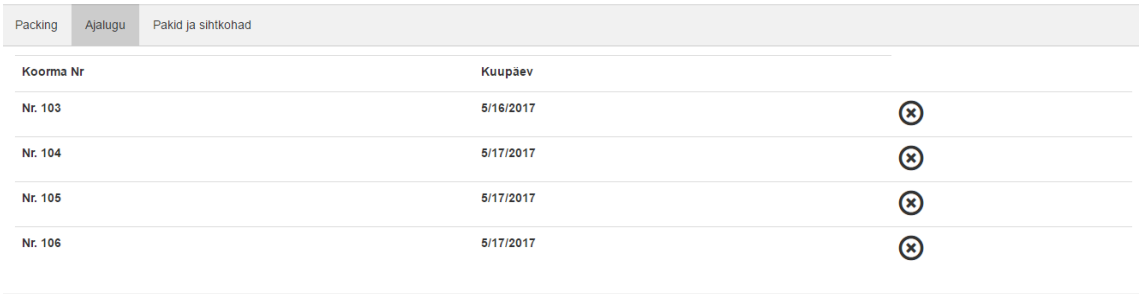
Navigeerides autode vahel ei vahetata pilti vaid vastavalt valitud autole joonistatakse Canvasele ainult selle auto pakid. Autot on võimalik kustutada, kui neid on rohkem kui üks. Selleks tekib graafilisele liidesele üles äärde rist, mida vajutades saab auto kustutada. Auto kustutamisel kustutatakse ka sellele autole lisatud pakid nimekirjast ja kuvatakse järgmist autot. Autode vahel navigeerimiseks tekivad liidese äärtesse nooled, kui on võimalik vahetada autot.

Koorma salvestamisel salvestatakse eraldi koorma objekt, koorma küljes olevad pakid ja genereeritakse koorma autodest pildid ning salvestatakse need kõik andmebaasi. Seost koormaga hoiavad pakid ja pildid ise, omades koorma numbrit. Kui koorem on laetud varasemast koormast, on võimalik see peale muutmist salvestada uue koormana, andes sellele uue numbri ja kuupäeva ning pannes pakkidele külge uue koorma numbri ja tehes tühjaks nende key väärtuse, mis on seos olemasoleva pakiga baasis. Siis saavad nad salvestamisel uue unikaalse key väärtuse.

Koorma printimise vajutamisel genereeritakse koormast pilt ning avatakse see uues aknas koos pildiga, kust on kerge seda kas salvestada saatmiseks või printida.

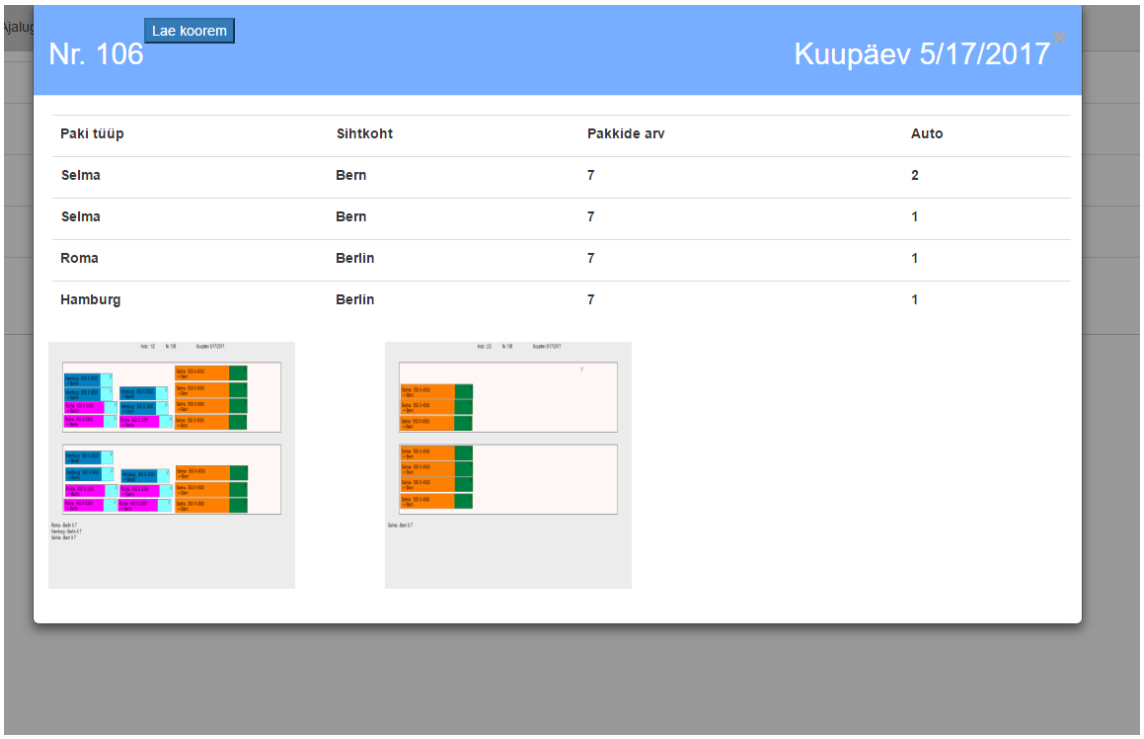
3.4.2 Ajaloo vaade

Ajaloo kuvas on nimekiri loodud koormatest.



Joonis 7 Koormate ajaloo vaade

Koormaid on sealt võimalik kustutada ja koormale peale vajutades avaneb modaal koorma detailandmetega, mis sisaldab koorma enda detaile, koorma pakkide nimekirja, kus näeb pakkide sihtkohti ja tüüpi ning missuguse auto peal nad on.

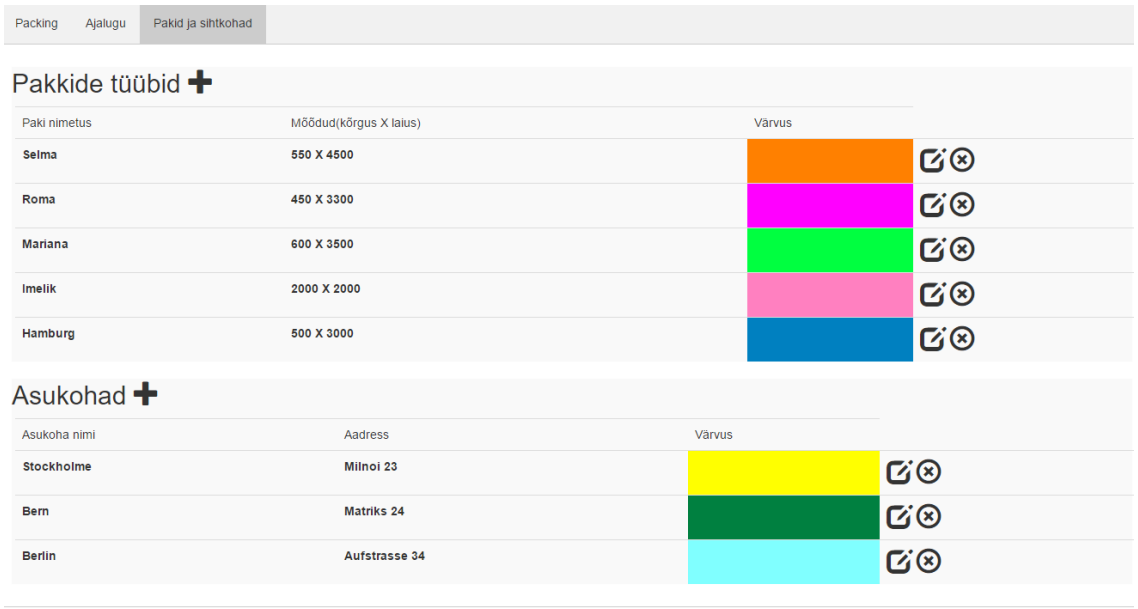


Joonis 8 Koorma detailvaade

Lisaks kuvatakse modaalis ka pildid koormast. Vastavalt igale autole on eraldi pilt ning pildile vajutades avaneb see uues aknas, kust seda saab välja printida. Modaalis on ka nupp „Lae koorem“. See asendab praegu graafilisel liidesel oleva koorma selle koormaga, mis valiti. Selleks laetakse baasist vajalikud andmed ja sisestatakse need liidesesse. Edasi on võimalik seda sama koormat muuta liidesel, et jätkata pooleli jäänud tööd või kasutada seda uue koorma põhjana. Hiljem saab seda salvestada uue koormana. Kui kustutatakse koorem, leitakse ka sellega seotud pakid ja pildid ning kustutatakse ka need.

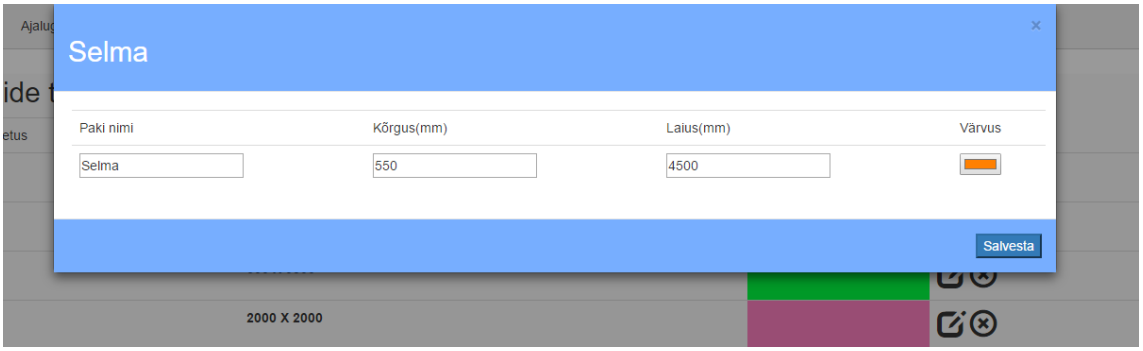
3.4.3 Pakid ja sihtkohad vaade

Pakkide ja sihtkohtade vaates on kaks nimekirja. Mõlemad nimekirjad laetakse rakenduse avamisel ja neid hoitakse globaalsetes muutujates. Mõnda pakki või asukohta muutes muutuvad nad ka mujal valikutes. Graafilist liidest pakkide ja asukohtade muutmine ei muuda. Peale muutmist peab graafilisele liidesele pakid uuesti genereerima.



Joonis 9 Pakkide ja asukohtade vaade

Mõlemal nimekirjal on võimalik pluss märgist lisada kumbagi tüüpi juurde. Sinna vajutades avaneb modaal, kuhu peab sisestama vastavalt kas paki või asukoha andmed ja valima sellel värvuse. Igat pakki ja asukohta on võimalik muuta, vajutades rea lõpus olevale pliiatsiga kastile, mis avab samasuguse modaali nagu lisamisel ja kus on võimalik muuta andmeid. Salvestades uuendatakse andmed baasis ja rakenduses. Kõiki pakke ja asukohti on võimalik ka kustutada, vajutades rea lõpus olevale ristile, mis kustutab selle rea täielikult.



Joonis 10 Pakitüüpide ja asukohtade lisamise modaal

4 Võimalikud arengusuunad

Valmis saanud rakendusele on võimalik juurde lisada funktsionaalsust ja arendada seda edasi. Selles peatükis räägib autor osadest võimalikest lahendustest mida oleks võimalik edaspidi teha.

4.1 Automatiseerimine ja optimiseerimine

Nagu olemasolevate lahenduste puhul on ka selles rakenduses võimalik juurde panna koorma paigutuse automaatne arvutamine, andes ette vajalike pakkide nimistu. See annab võimaluse optimeerida paigutust ja lasta rakendusel ära teha koorma koostamine algusest lõpuni ehk kuni jooniseni. See võib anda parema tulemuse kui kasutaja enda poolt kokku pandud koormad, mis annaks edasipidi ka säästu aja- ja kütusekulus ja lihtsust pakkide maha võtmisel sihtkohtades. Optimeerimist saaks kasutada teekonna arvutamise lisamisel. Selleks on vaja ühendada rakendus mõne kaardisüsteemiga nagu Google Maps ja arvutada sealt võetud andmetega lühim ja parim teekond. Mitme auto puhul saaks ära jagada missugustesse sihtkohtadesse mis autod lähevad ja vastavalt sellele jagada pakid autode peale.

Arvutamise loogikat on võimalik teha läbi Javascripti kasutaja arvutis või läbi Firebase serveri. Firebase Hosting toetab funktsioonide [11] hoidmist pilves, kuhu on võimalik panna keerulisemat äriloogikat ja arvutamist. Neid käivitatakse kas läbi koodi kasutades Firebase funtksionaalsust või läbi läbi HTTPS päringute. Sinna on võimalik lisada algoritm, mis tagastab sobiva pakkide asetuse etteantud nimekirjaga, arvestades pakkide suurust, sihtkohti ja vajadust pakkide kätte saamiseks sihtkohas neid võimalikult vähe ümber tõsta.

4.2 3D Joonis

Kuigi praegu vajadus puudub, on võimalus ehitada joonis üles 3D pilti näitama, mis võib aidata ülevaate saamisel koormast ja lubaks erisuurustega pakke, mis poleks standardmõõdus ning võivad ära rikkuda praegu kasutusel oleva eraldatud poolte

süsteemi. Seda on võimalik teha kasutades olemasolevaid lahendusi nagu WebGL, mis kasutab HTML Canvast WebGLi poolt välja arendatud funktsionaalsusega.

4.3 Liides andmete saamiseks väljastpoolt

Võimalik on teha liides joonise koostamiseks väliste andmefailide või tarkvara poolt antava info põhjal (Excel, teised andmebaasid), mis võivad ettevõttes kasutusel olla. Selle läbi väheneb eksimisvõimalus ja puudub kasutajal vajadus uuesti sisestada andmed iga koorma koostamise jaoks eraldi. See töötaks hästi koos automatiseerimisega, kus kasutajal on vaja anda ette ainult andmed ning süsteem teeb ise kõik vajaliku ning tagastab koorma pildi. Kindlasti on ka siis võimalik veel manuaalselt muuta koostatud koormaid.

4.4 Autoriseerimine

Võimalik on lisada kasutajad koos audentimisega. Iga kasutaja saaks näha ainult tema enda sisestatud koormaid. See on vajalik funktsionaalsus, kui rakendust hakkab kasutama rohkem kasutajaid ning keegi ei soovi, et nende koormaid kustutatakse.

4.5 Autode muudetavus

Selle arenduse skoobist jäi välja veoautode kastide suuruse muutmise võimalus. See polnud praeguse funktsionaalsuse juures kohe vajalik ning jäi väikese prioriteedi tõttu skoobist välja.

5 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli uurida ettevõtte vajadusi ning vastavalt vajadustele luua veebirakendus, millega oleks võimalik koostada koorma joonist veoautodele. Nõuete ja eemärkide analüüsi tulemusena saadud nimekijra järgi saab öelda, et need said kõik täidetud selles arenduses. Valmis prototüüp, mida on lihtne kasutada ja täidab püstitatud nõuded ning mis asendab ja muudab kergemaks senise töövoo. Lahenduse poolest on see ettevõttele mugav ja sisaldab neid funktsionaalsusi, mida ettevõttel vaja on. Kuna koorma koostamiseks praegu selles ettevõttes ei ole ühtegi lahendust peale CAD tööriistade kasutamise, siis see leiaks kindlasti kasutust ettevõttes.

Isiklikust küljest vaadates oli projekt väga kasulik ja õpetlik, sest sain kokku puutuda rohkem tehnoloogiatega, millega tavapäraselt kokku ei puutu. Veebirakenduste külg on mul tuntavalt nõrgem ja vajas süvenemist ja õppimist. Kaua kulus erinevate võimalike lahenduste uurimine ja neist sobivaima leidmine, kuid tänu sellele tunnen ennast mugavamalt järgmiseks projektiks õiget lahendust otsides. Samuti vajas õppimist rakenduste korrektne ülesehitus ja kasutajaliidese disain.

Kasutatud kirjandus

[1] EasyCargo3D “EasyCargo3D” http://www.easycargo3d.com/ [Võrgumaterjal] [Kasutatud 4 2017]

[2] LoadXpert’i Load Planning “LoadXpert”http://www.loadxpert.com/ [Kasutatud 4 2017]

[3] Andris Reinman, „JavaScript edasijõudnutele” [Võrgumaterjal] Available: http://tahvel.info/books/javascript.html [Kasutatud 5 2017]

[4] “HTML Canvas” [Võrgumaterjal] Available: https://en.wikipedia.org/wiki/Canvas_element [Kasutatud 4 2017]

[5] “WebGL” [Võrgumaterjal] Available: https://en.wikipedia.org/wiki/WebGL [Kasutatud 4 2017]

[6] “Firebase Database” [Võrgumaterjal] Available: https://firebase.google.com/docs/database/ [Kasutatud 4 2017]

[7] “Firebase Hosting” [Võrgumaterjal] Available: https://firebase.google.com/docs/hosting/ [Kasutatud 4 2017]

[8] Sander Leetus “AngularJS raamistiku õppematerjal” 2015 [Võrgumaterjal] Available: www.cs.tlu.ee/teemad/get_file.php?id=400 [Kasutatud 3 2017]

[9] Veoautode mõõdud [Võrgumaterjal] Available: http://www.etslogistika.ee/teadmiseks/veokite-ja-haagiste-mahutavus/ [Kasutatud 4 2017]

[10] Dan Gries “A simple HTML5 Canvas dragging example using object oriented programming” [Võrgumaterjal] Available: http://rectangleworld.com/blog/archives/129 [Kasutatud 3 2017]

[11] “Firebase Functions” [Võrgumaterjal] Available: https://firebase.google.com/docs/functions/ [Kasutatud 3 2017]