TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ra Koort 179193IADB

„Ehitustööde haldustarkvara prototüübi arendamine“

bakalaureusetöö

Juhendaja: Aleksei Talisainen
MSc

Tallinn 2020

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: Ra Koort

04.12.2020

Annotatsioon

Käesoleva töö eesmärgiks on luua erinevaid rasketehnikaga tegeleva ettevõtte tööprotsesse lahendava rakenduse prototüüp.

Arendusprotsessi käigus luuakse esirakendis korduvkasutatavad komponendid ning koostatakse nende abil erinevad vajalikud vaated. Tagarakendisse implementeeritakse süsteemis kasutatavad rollid ning vajadusel esirakendi komponente täitev äriloogika.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38-l leheküljel, 8 peatükki, 13 joonist, 4 tabelit.

Abstract

Development of a management system prototype for construction work

The aim of this thesis is to create a software prototype, that allows construction companies to better manage their internal processes and work orders.

During development different components and views will be created in the front-end and the necessary business logic for those components in the back-end.

The thesis is in estonian and contains 38 pages of text, 8 chapters, 13 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface,* Rakendusliides – määrab kuidas tagarakend esirakendiga suhtleb
Blob	Muutumatu andmetüüp
CSR	*Client-side rendering,* kliendipoolne renderdamine
CSS	*Cascading style sheets*, Veebilehtede stiilimiseks mõeldud standard
DBaaS	*Database as a service,* Andmebaasi haldamise teenus
Flexbox	Kaasaegne CSS-i paigutusmoodul
HTML	*Hypertext markup language,* Defineerib veebilehtede struktuuri
JSON	*Javascript Object Notation,* Andmevahetusvorming
JWT	*JSON Web Token,* Kahe osapoole vaheline turvaline andmevahetusstandard
MVP	*Minimum viable product*, minimaalne toimiv toode
NoSQL	Mitte-relatsiooniline andmebaas
OMS	*Order management* system, tellimuste haldussüsteem
Renderdamine	Teatud reeglite põhjal millegi kuvamine
REST	*Representational state transfer*, API arhitektuur
SEO	*Search engine optimisation,* Otsingumootorile optimeerimine
Snapshot	Testimiseks mõeldud hetkelise rakenduse oleku põhjal talletatav referentsfail
SPA	*Single Page Application*, üheleherakendus
SSG	*Static site generation*, staatiline lehe genereerimine
SSR	*Server-side rendering,* serveripoolne renderdamine
Teek	*Library,* Rakenduses kasutatav abistav valmislahendus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Ehitussektor on kahtlemata suur ja oluline osa majandusest ning seda digitaliseerida püüdvaid tehnoloogiafirmasid on palju. Vaatamata sellele ei ole digitaliseeritud lahendused ehitusfirmadel ja -töödel sugugi laialdases kasutuses. Kasutatakse peamiselt rakendusi lahendamaks spetsiifilisi probleeme. Näiteks võidakse kasutada töötunde haldavat teenust, kuid töökäsklused antakse ikka telefoni teel. Tööde ja töötajate haldamine käib siiamaani valdavalt suulisel teel või paberkandjal kuna ehitusfirmade personal ei kipu olema eriti vastuvõtlik uute tehnoloogiate suhtes. Levinud on hoiak, et asju ei tasu parandada, kui need pole katki.

Kindlasti oleks aga tööprotsesside digitaliseerimine suureks kasuks just ehitussektorile. Peamine ressurss ehitustöödel on aeg, mida kulub hetkel liialt palju käskude edastamisele, neis veendumisele, tehtud tööde kinnitamisele, kokkulepetele jõudmisele (nt. masinate rentimisel) jne. Digitaliseeritud andmete õigsuse kindlus ja edastamise kiirus ei ole võrreldav üksikute inimeste võimetega. Eemaldades ebavajaliku ajakao saavad ehitusettevõtted rohkem väärtust oma töötajate ajast. Samuti elavdavad kiiremad ja efektiivsemad ehitustööd kohalikku majandust, tekitades uusi äri- ja elamispindu või parandades piirkonna üldpilti ja elukvaliteeti.

Infrafly OÜ [1] eesmärk on luua terviklik veebiteenuste pakett rasketehnikaga tegelevatele ehitusfirmadele, mis mitte ainult ei asendaks valdava osa paberimajandusest, vaid parendaks ka firmasiseste tööprotsesside efektiivsust. Sealjuures peaks loodud lahendus olema intuitiivne, et ka näiteks tehnoloogiale võõrad ehitusmasinate operaatorid oleksid võimelised selle vähese vaevaga omaks võtma ning atraktiivne, et potentsiaalsete klientettevõtete korrespondendid sellele võimaluse annaks. Diplomitöös käsitletava toote nimi on Infrafly OMS, mille arendus algas samaaegselt autori palkamisega Infraflysse.

2 Taust

Siin peatükis kirjeldatakse käesolevas diplomitöös käsitletava projekti tausta ja kavandamist. Autori roll on läbi viia taustauuring konkurentide kohta, tutvuda olemasoleva koodibaasiga ning anda oma panus uue projekti kavandamisesse, osa võtta rakenduse arendamisest ning hiljem anda hinnang tehtud tööle ja teenusele kui tervikule.

2.1 Projekti taust

Tööle asumise hetkel oli ettevõttel juba toimiv toode Infrafly Marketplace [2]. See on React Native [3] raamistikule ehitatud hübriidmobiilrakendus mis võimaldab erinevat rasketehnikat rentida ja rendile anda. Toote vastu tekkis Eesti turul mõnel määral huvi, kuid mitte piisavalt et see kasumlikuks teenuseks pöörata. Katsed minna välisturule ebaõnnestusid ning otsustati läbi viia pivot. Mobiilrakendus jäi alles, aga ettevõtte peamiseks fookuseks sai käesolevas töös käsitletav Infrafly OMS [1].

2.2 Probleem

Infrafly keskendub peamiselt kolme probleemi lahendamisele. Esiteks on tellimuste haldamine paberkandjal ebakindel. Informatsioon võib kommunikatsiooni käigus muutuda või kaduma minna. Masinajuhtidega suhtlemine on ajakulukas ja tihtipeale kaootiline. Teiseks probleemiks on paberimajandus, mida tekib üks hetk lihtsalt liiga palju, et firma toiminguid efektiivselt lahendada. Kolmandaks on töölehtede kogumine ja allkirjastamine, mis muidu nõuab füüsilist kohalolekut, kuid digitaalselt on seda võimalik ka kaugelt teha.

2.3 Eesmärk

Diplomitöö eesmärk on praktika jooksul välja arendada Infrafly OMS-i kasutatav prototüüp, mis lahendaks vähemalt eelmises peatükis välja toodud probleemid. Lisaks sellele antakse hinnang arendusele ja selle tulemusel valminud tootele.

2.4 Metoodika

Toote arendus toimub agiilse arenduse [9] põhimõtete järgi, kus iga iteratsiooni ehk sprindi pikkuseks on üks nädal. Plaan esialgseks funktsionaalsuseks pannakse paika taustauuringu ja partnerettevõtte töötajatega konsulteerimise põhjal. Toote MVP (ingl *minimum viable product*) valmimisel lähtutakse edaspidi eelkõige klientide soovidest ja tähelepanekutest.

Rakenduse prototüübi loomiseks valitakse planeeritud funktsionaalsuste põhjal välja tehnoloogiad, mis toetaksid seatud kolme probleemi lahendamist. Samuti peaksid tehnoloogiad olema kaasaegsed, arvestades sissejuhatuses mainitud toote intuitiivsust ja atraktiivsust. See tähendab, et lõpptulemusena valminud prototüüpi peaks olema võimalik müüa.

Kõikide eelnevate valikute puhul on lähtutud ka kitsatest ajalistest piirangutest. Otsiti paindlikke lahendusi, mis lubaksid kiiresti uusi funktsionaalsusi implementeerida ja testida, et võimalikult varakult esimesed kliendid saada. Kuna tegemist on vähemalt avalikult kättesaadavate toodete seas võrdlemisi katsetamata lahendusega, on oluline lähtuda edaspidises arenduses klientide soovidest mitte spekulatsioonist.

3 Analüüs

Selles osas on välja toodud taustauuring ja partnerettevõtteks oleva ehitusfirma töötajatega konsulteerimise abil paika pandud funktsionaalsused ning süsteemis eristatavad rollid MVP jaoks.

3.1 Taustauuring

Taustauuringu tulemusena leiti kolm olemasolevat teenust, mis olid Infrafly algse planeeritava tootega võrreldavad. Iga teenuse puhul on välja toodud mõned head ja halvad omadused. Omadustele on antud hinnangud kahes astmes: + on hea ja ++ on väga hea ning - on halb ja -- on väga halb.

3.1.1 Texada

Texada puhul on tegemist USA turul tegutseva platvormiga, kelle peamiseks kliendiks on masinapargi omanik. Uuringus leitud kolme teenuse seast on see kindlasti kõige sarnasem Infraflyle.

Tabel 1. Texada rakenduse omadused

Omadus	**Kirjeldus**	**Hinnang**
Terviklik süsteem	Pakub masinaid rentivale ettevõttele äriprotsesside lahendamiseks tervikut paketti (masina jälgimisest arve koostamiseni).	++
Keeruline kasutajaliides/-kogemus	Kasutajaliides on üsna algeline. Keerulise süsteemi puhul on tihti raske leida mida otsid. Kliendid vajavad väljaõpet.	--
Töölehed edastatakse automaatselt	Täidetud ja allkirjastatud töölehed edastab süsteem kontorile automaaselt.	+

3.1.2 Navirec

Navirec on Eestis populaarsuselt teine ning ülemaailmselt kasutusel olev veokipargi haldussüsteem. Kuigi peamiseks kliendiks on transpordiga tegelev ettevõte, sai siiski tõmmata paralleele planeeritud rakenduse talituse kaudu.

Tabel 2. Navireci rakenduse omadused

Omadus	**Kirjeldus**	**Hinnang**
Reaalajas jälgitav	Tarkvaras hallatavaid protsesse on suuresti võimalik reaalajas jälgida. Veoste puhul on see eriti oluline.	+
Võrdlemisi vähene andmetöötlus	Teenus on mõeldud rohkem visuaalse ülevaate saamiseks parasjagu käivatest töödest.	-
Dünaamiline graafikute koostamine	Graafikuid on võimalik detailideni konfigureerida ja kolmes erinevas formaadis eksportida.	++

3.1.3 Rose Rocket

USA ja Kanada turul tegutseva Rose Rocketi puhul on samuti tegemist tootega, mis on mõeldud veostele. Nagu ka Navireci puhul sai siin tõmmata paralleele sarnaste planeeritavate vaadete kaudu.

Tabel 3. Rose Rocketi rakenduse omadused

Omadus	**Kirjeldus**	**Hinnang**
Eraldi keskkonnad juhile ja kontorile	Masinajuhtidele on loodud eraldi mobiilrakendus.	+
Dünaamiline teavituste süsteem	Teavitustele on võimalik määrata tähtsuse aste ning neid selle järgi konfigureerida.	+

3.1.4 Järeldused

Uuringu käigus leiti, et Eesti turul ei ole veel toodet, mis pakuks terviklikku lahendust masinapargiga ehitusettevõttele. Muude turgude toodete seast leiti üks lähedalt võrreldava funktsionaalsuse- ja suunitlusega teenus. Kõigi eelnevas sektsioonis välja toodud omadustega arvestati Infrafly OMS-i esialgsete nõuete koostamisel.

3.2 Rollid

Uuringute põhjal jõuti järeldusele, et erineva talitusega ameteid on ehitusfirmades üldjoontes neli: omanik, logistik, raamatupidaja ja operaator. Seega peaks ka arendatav rakendus neid nelja ametit rollidena implementeerima. Teatud vaated ja toimingud peaksid olema keelatud või lubatud vastavalt rollile.

Operaatori puhul peaks ligipääsetav olema vaid mobiilvaade, kus kuvatakse kasutajale ainult temaga seotud tellimused ja masinad. Kuna tegemist on tunnitöölisega peab operaator olema võimeline looma tellimuse alla töölehti, kirjeldamaks tehtud tööd logistikutele ja raamatupidajatele. Lisaks sellele peavad nad olema võimelised ka enda parooli muutma, sest kasutaja loob neile logistik. Sealjuures esimesel sisselogimisel peaks süsteem nõudma paroolivahetust, et vähendada kommunikatsioonis ringi hõljuvaid salasõnu, mis on suur turvarisk.

Paljud funktsionaalsused vähendavad just logistikupoolset dokumenteerimist. Seega otsustati, et logistikud pääsevad vähemalt algselt ligi kõikidele vaadetele. Samuti on lubatud neile kõik toimingud peale teiste paroolide muutmine.

Kuna raamatupidaja töötab eelkõige tehtud tööde põhjal kogutud informatsiooniga peaks ta ligi pääsema kõikidele vaadetele nagu logistik, kuid ilma andmete muutmise õigusteta. Küll aga peaks olema lubatud tellimuste ja töölehtede eksportimine mis tahes valitud formaati. Rakenduse suurim kasutegur raamatupidajale realiseerub tegelikult andmete automaatses kogumises ja töötlemises.

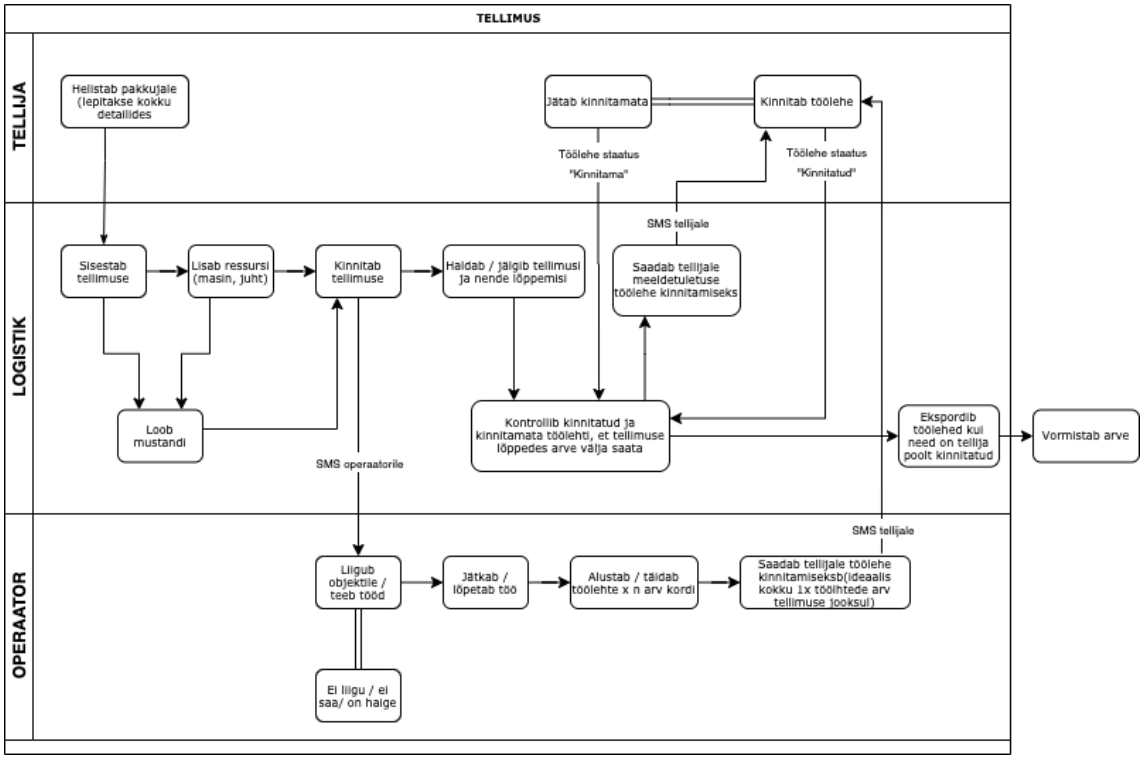
Omanikule peaksid ülevaatlikus korras kõik vaated ja funktsionaalsused lubatud olema, kuid eraldi organisatoorseid funktsionaalsusi algselt plaanis lisada ei ole. Samas on kindel, et tulevikus võetakse see teema käsile. Seega leiti, et oleks mõistlik roll juba süsteemi rajamisel luua.

Kõige tähtsam roll on logistik, kellele on suunatud enamik teenuse funktsionaalsustest. Kõikide vaadete (peale sisse logimise) ja rollide puhul peaks toimuma kasutaja autentimine ja rollidepõhine autoriseerimine.

3.3 Funktsionaalsused

Siin on esitatud erinevate toimingute kaudu planeeritud funktsionaalsused. Kõik toimingud lähtuvad esmalt kasutaja rollist. Kõikide tabelite puhul on kasutajal võimalik kuvatavaid andmeid kas otsingu või määratud filtrite abil täpsustada. Lisaks sellele on valdav osa tabelite veergudest erinevat moodi järjestatavad.

Järgneval joonisel on välja toodud ootuspärane rakenduse kasutusvoog.



Joonis 1. Ootuspärane süsteemi kasutusvoog

Järgnevad toimingud peavad olema võimaldatud kõikidele rolliga kasutajatele.

Üldised toimingud:

- Logib sisse
- Logib välja, hävitab sessiooni
- Vahetab kuvatava teksti keelt
- Vahetab parooli

Raamatupidajale lubatud toimingud peaksid olema valdavalt ülevaatlikud. Andmete muutmisõigust neil olla ei tohiks. Küll aga peab olema lubatud andmete eksportimine.

Raamatupidaja toimingud:

- Vaatab tellimuste tabelit
- Vaatab töölehtede tabelit
- Vaatab tehnika tabelit
- Vaatab töötajate tabelit
- Vaatab klientide tabelit
- Ekspordib töölehtede väljavõtte PDF formaadis
- Ekspordib töölehtede väljavõtte XLSX formaadis

Operaatorile peaks olema võimaldatud oma töödega seonduvate andmete vaatamine, loomine ja muutmine. Rakendust peaks saama kasutada vaid neile loodud spetsiaalsete vaadete kaudu.

Operaatori toimingud:

- Saab SMS-iga teate uue lisatud tellimuse kohta
- Vaatab oma aktiivseid tellimusi
- Vaatab tellimuse detailvaadet

- Täidab tellimuse küljes oleva tööliigi põhjal töölehe väljad ning lisab selle tellimuse alla
- Muudab kinnitamata töölehe andmeid
- Lõpetab tellimuse

Tellija jaoks ei tohiks olla süsteemis eraldi rolli, kuna neile avaneb spetsiaalse lingi kaudu vaid üks vaade. Neile ei kehti ka sektsiooni alguses välja toodud üldised toimingud peale keele valiku.

Tellija toimingud:

- Saab SMS-iga teate uue kinnitamata töölehe kohta
- Avab kinnitusvaate sõnumis oleva spetsiaalse lingi abil
- Kontrollib töölehe andmete õigsust
- Vajadusel sisestab ehitusplatsipõhise šifri
- Kinnitab töölehe
- Lükkab töölehe tagasi

Logistiku toimingud peaksid tagama süsteemi kasutatavuse teiste rollidega kasutajatele. Kuna logistiku ülesanded ettevõttes on administratiivsed peaks see kajastuma ka neile lubatud funktsionaalsustes.

Personali vaade:

- Vaatab firma tööliste tabelit
- Lisab uue firmasisese kasutaja
- Muudab firmasisese kasutaja kontaktandmeid
- Deaktiveerib firmasisese kasutaja

Masinate vaade:

- Vaatab firma masinate tabelit
- Valib uue masina süsteemi poolt pakutavate masinatüüpide seast
- Määrab masinale parameetrid, asukoha ja hinna
- Lisab tabelisse uue masina
- Muudab masina andmeid
- Deaktiveerib masina

Klientide vaade:

- Vaatab firma klientide (ettevõtete) tabelit
- Lisab uue kliendi
- Muudab kliendi andmeid
- Määrab kliendi maksevõimekuse
- Lisab kliendi alla kontaktisiku
- Muudab kontaktisiku andmeid
- Deaktiveerib kontaktisiku
- Deaktiveerib kliendi

Tellimuste vaade:

- Vaatab töölehtede koondtabelit (kõikide tellimuste töölehed)
- Saadab kliendile meeldetuletuse kinnitamata töölehest
- Muudab töölehe andmeid
- Kustutab töölehe

- Vaatab tellimuste tabelit
- Lisab uue tellimuse mustandi
- Määrab tellimusele algus- ja lõppkuupäeva, objekti nime, asukoha
- Määrab tellimusele klientide nimistust kliendi, kliendi nimistust kontaktisiku
- Määrab tellimusele masinate nimistust masina, tööliste nimistust masinale operaatori
- Lisab tehnika tüübile uue tööliigi, tööliigile nimetuse ja hinna
- Valib tööliigi varem lisatud tööliikide seast
- Salvestab tellimuse mustandi
- Aktiveerib tellimuse mustandi
- Vaatab spetsiifilise tellimuse alla kuuluvaid töölehti
- Lisab tellimuse alla töölehe (kui operaator on tegemata jätnud)
- Tühistab tellimuse mustandi
- Muudab tellimuse andmeid
- Kustutab tellimuse

4 Tehnoloogiad

Selles peatükis on kirjeldatud olulisemad tehnoloogiad, mille peale veebirakendus üles ehitatud on. Tehnoloogiad on jaotatud API ehk tagarakendi- ja esirakendipoolsetesse osadesse. Nagu sai eelnevalt mainitud oli API juba mobiilrakenduse tõttu osaliselt üles ehitatud. Kuna mobiilrakenduse arendustöö toimus firmaväliselt on mõne tehnoloogia kirjelduse juurde lisatud autoripoolne spekulatsioon selle valiku suhtes.

4.1 Tagarakend

Tagarakendi ülesanne on suhelda andmebaasiga ning serveerida andmeid mobiil- ja veebirakendusele vastavalt olukorrale. Oma olemuselt on tegemist võrdlemisi standardse pilveteenusega [10]. Tagarakendi ülesanded on jaotatud erinevate teenuste vahel, mis suhtlevad klientrakendusega läbi interneti. Siin töös käsitletava teenuse puhul jookseb API server Heroku [11] platvormil, andmebaasid MongoDB Atlas [12] platvormil ning kasutajate autentimine toimub läbi Google Firebase'i [13].

Infrafly OMS-i puhul on andmetöötlus viidud osaliselt esirakendisse, kasutades selleks React state'i [14] ja EasyPeasy [15] teeki, millest tuleb rohkem juttu esirakendi osas 4.2.2.

4.1.1 MongoDB ja Firebase

Andmete talletamiseks on kasutuses MongoDB [6] *NoSQL* [16] tüüpi andmebaas. *NoSQL* andmebaaside puhul peetakse silmas kõiki andmebaase, mis ei ole rangelt relatsioonilise [17] struktuuriga. See tähendab, et omavaheliste seostega andmeid ei pea ilmtingimata jagama ja pärima eraldi tabelitest. Samas aga on võimalik luua seoseid ning neid pärida üheses andmestruktuuris. MongoDB puhul on selleks JSON [18].

NoSQL andmebaaside eelised tulevad peamiselt välja pilveteenuste [19] puhul. Võrreldes relatsiooniliste andmebaasidega on selline lahendus tihti skaleeritavam ja oma talituselt efektiivsem. Samuti võimaldavad paindlikud andmestruktuurid uusi funktsionaalsusi kiiremini välja arendada ja testida, kuna andmebaasi ülesehitust ei ole vaja ilmtingimata uuendada [20]. MongoDB puhul on eeliseks veel see, et andmeid hoitakse samal kujul nagu neid kasutatakse ka esirakendis.

Tõenäoliselt valiti Infrafly poolt palgatud varasemas arendustöös MongoDB kahel põhjusel. Esiteks võimaldab MongoDB dünaamiline struktuur arendusprotsessi alguses funktsionaalsusi kiirelt prototüüpida. Teiseks on ehitustöödel kahtlemata oluline geograafiline info (ehitusplatside ja masinate asukohad) ning MongoDB pakub koordinaatide talletamiseks spetsiaalset andmetüüpi GeoJSON [21], mida saab esirakendis lihtsasti kasutada näiteks Google places API-ga [22].

MongoDB andmebaase on tegelikult kaks: *staging* ja *production* [23]. *Production* andmebaasi peal jookseb teenus, mis on klientidele kättesaadav. Arendustöö ja testimine aga toimub vastu *staging* andmebaasi, et vähendada võimalikke teeninduskatkestusi ja muid vigu.

Kasutajainfo talletamiseks on kasutusel Google Firebase [13]. Firebase on terviklik pilveteenus ning pakub palju erinevaid võimalusi, kuid Infrafly OMS-i puhul on neist kasutuses ainult kasutajate autentimine. Esialgu talletati seal kõiki süsteemi andmeid, kuid arenduse käigus migreeriti andmed MongoDB peale. Ilmselt valiti see Infrafly poolt palgatud arendustöös selle pärast, et vastava süsteemi loomine oleks olnud liiga kulukas ning ka ebavajalik, arvestades rakenduse funktsionaalsust.

4.1.2 Google Places

Google Places API [22] on tagarakendis kasutusel tõlgendamaks MongoDB-sse salvestatud GeoJSON tüüpe asukohale spetsiifiliste andmetega objektideks. Talletatud koordinaatide põhjal saadetud REST päringud saadavad tagasi tegelikud linna, riigi ja tänava nimetused ning tänavanumbrid.

4.1.3 Segment

Kuna ehitussektori digitaliseerimine ei ole veel nii laialdaselt levinud on üks oluline osa käsitletavast teenusest analüütika. Tähtis on hoida tulevaste klientidega tihedat kommunikatsiooni, et aru saada nende soovidest ja muredest, kuid veel tähtsam on logida nende rakenduskasutust. H.H. Olssoni ja J. Boschi sõnul [24] leiab kõige konstruktiivsema informatsiooni teenuse arendamiseks läbi põhjaliku analüütikasüsteemi ja klientide kommunikatsiooni kombineerimise.

Infrafly OMS-is on selleks peamiselt kasutuses Segment [25]. Segment aitab koondada erinevad spetsiifilisemad analüütikalahendused ühte kesksesse platvormi, millega on

kergem saadud tulemusi koguda ja töödelda. Kuigi tegemist on arendusele olulise osaga kajastub see pigem tulevaste sprindieelsete otsuste langetamisel kui kirjutatavas koodis, kuna tänu Segmentile toimuvad kõik analüütika päringud ühe API kaudu.

4.1.4 Nexmo

Mõlemas Infrafly teenuses saadetakse mõningad teated kliendile sõnumiga, kasutades selleks Nexmo SMS API-t [26]. Nexmo API võimaldab läbi REST päringute saata ja vastu võtta SMS-e ülemaailmselt. See on lihtsasti kasutatav ja skaleeritav lahendus, vältimaks kulukat arendustööd. Samuti on see taskukohane valik arenevale ettevõttele, kuna raha küsitakse ainult saadetud ja vastu võetud sõnumite eest.

4.2 Esirakend

Esirakendi tehnoloogiate valikutes lähtuti peamiselt asjaolust, et arendatav rakendus peaks olema hästi optimiseeritud, et kasutuskogemus oleks sujuv. Kuna funktsionaalsused keskenduvad suuresti erinevate andmete lisamisele ja muutmisele oli tähtis valida raamistikud, mis võimaldavad komponente dünaamiliselt renderdada ja andmeid ka esirakendis töödelda. Uusi andmeid sisestades ja kuvades peaksid muutused võimalikult kiiresti lehel kajastuma.

4.2.1 React

React.js [7] on JavaScriptile kirjutatud avatud lähtekoodiga [27] teek, mis võimaldab luua klientrakendusele kasutajaliidese, tarvitades selleks taaskasutatavaid komponente. Enamikku Reactil põhinevatest veebilehtedest kutsutakse üheleherakendusteks [28] ehk SPA-deks.

Üheleherakenduste peamisteks eelisteks on esiteks see, et HTML-i genereerimine on viidud serveri poolt kliendi poolele. See tähendab, et peale esialgset lehe laadimist saadetakse ja päritakse serverist ainult andmeid, mille abil uuendatakse klientrakenduse sisu dünaamiliselt. Niiviisi ei ole tarvis kasutaja interaktsiooni korral tervet lehekülge uuendada, piisab vaid alamkomponentide taasrenderdamisest. Teiseks piisab rakenduse paigaldamiseks vaid staatilisest serverist [29]. Staatiline server kuvab lehed kliendile *as-is* [30] ehk serveri pool mingisugust töötlust ei toimu. Selline lahendus võimaldab Infrafly OMS-il hõlpsasti realiseerida pilveteenuse arhitektuuri, kus veebilehe sisu päritakse üle interneti teistest serveritest.

React põhineb JSX [31] süntaksiga loodud komponentidel. JSX võimaldab komponentide sees kasutada JavaScripti lauseid, mille abil täidetakse komponendi sisu dünaamiliselt või seatakse tingimused selle renderdamiseks. Programmi kompileerimisel tõlgendatakse mainitud laused tavalisteks JavaScripti funktsioonideks.

Komponendid ise kirjutatakse reeglina eraldi failidesse, kuna need on korduvkasutatavad. Tegelikkuses on ka Reacti komponentide puhul tegemist funktsioonidega, mis tagastavad HTML elemendid ja täidavad need sisuga. Tänu modulaarsusele aitab React tõhusalt vähendada koodi kordamist.

4.2.2 EasyPeasy

EasyPeasy [15] on populaarse teegi Reduxi [32] abstraktsioon. See pakub alternatiivset liidest Reduxi funktsionaalsuse kasutamiseks. Redux ise on lahendus haldamaks JavaScript rakenduste ja selle komponentide olekut (ingl *state*) ehk tegemist on õigupoolest esirakendi andmetevoo arhitektuuriga. See lahendus võimaldab talletada sessioonipõhiseid andmeid (olekut), mida on võimalik ülejäänud rakenduses läbi ühese struktuuri kasutada. Tegemist on olulise funktsionaalsusega, kuna Reactil põhinevad rakendused lähtuvad suuresti komponentide olekust. Reduxi ja EasyPeasy kasutamine tsentraliseerib oleku haldamise, tänu millele on seda kergem kasutada ja testida. Infrafly OMS-i puhul mängib see ka olulist rolli vormidesse sisestatud andmete ja valitud filtrite säilitamisel, kuna tihtipeale võib tekkida vajadus korraga erinevatel lehtedel andmeid töödelda.

4.2.3 Next.js

Next.js [33] on raamistik Reactil [7] põhinevatele klientrakendustele. Kui Reacti ülesanne on luua veebilehe (või mobiilrakenduse) sisu, siis Next.js-i ülesanne on seda sisu kompileerida, jaotada ja kasutajale serveerida. Kuigi Next.js võimaldab rakenduses kasutada alamlehti koos vastavate aadressidega on Infrafly OMS puhul tegemist siiski üheleherakendusega. Erinevate lehtede ja aadresside jaotamine toimub virtuaalselt (v.a. sisselogimisleht).

Üks põhilisi probleeme mida Next.js parandab on esialgse lehe laadimise kiirus. Üheleherakenduse esimene avamine sessioonis kipub olema üsna aeglane, kuna veebilehitseja ootab, et kogu lehes sisalduv JavaScript ära laeks enne kui üldse midagi kuvada saab [34]. Sellist lahendust nimetatakse CSR-iks (ingl *Client side rendering*).

Next.js võimaldab juba rakenduse ehitamisel genereerida iga alamlehe jaoks vajaliku HTML koodi. Edaspidised päringud taaskasutavad seda sama HTML-i ning veebilehe sisu kuvatakse osaliselt kohe peale esimese päringu tegemist. Olles ära laadinud vajaliku JavaScript koodi muutub veebileht ka interaktiivseks. Sellist lahendust nimetatakse SSG-ks [35] (ingl *Static site generation*).

4.2.4 Yarn

Yarn [36] on Node.js arenduskeskkondadele mõeldud paketihaldur. See võimaldab oma projektis kasutada pakettidena teiste loodud lahendusi. Pakett tuleb käsurea abil alla laadida ning seejärel salvestatakse see sinu projekti kausta. Sealjuures salvestatakse iga paketi ja nende sõltuvuste versioonid faili *yarn.lock*. Edaspidi lähtub Yarn pakettide paigaldamisel selles failis olevatest versioonidest. See tähendab seda, et mitmeliikmelised arendustiimid saavad alati kindlad olla, et pakettide versioonid nende arenduskeskkondades on samad. Paketi kasutamiseks on vaja see vaid faili alguses importida.

4.2.5 SASS

Keerukamate projektide juures võib tihti juhtuda, et tavalisest CSS-i [37] süntaksist jääb kas mugavuse või kasutatavuse poolest midagi puudu. Selle jaoks on võetud kasutusele CSS-i eeltöötleja SASS [38]. Põhilised eelised käsitletavas projektis on klassi visuaalne kapsuleerimine ja mixinid [39]. Visuaalse kapsuleerimise all on mõeldud seda, et mainitud eeltöötleja süntaksiga on võimalik stiiliklassid HTML-i elementidele sarnaselt üles ehitada. Selline süntaks muudab koodi ja omaduste grupeerimise loetavamaks.

Mixinid imiteerivad oma olemuselt JavaScripti funktsioone, ning võimaldavad luua taaskasutatavaid stiilide kogumeid. Käesolevas töös on neid kasutatud peamiselt adaptiivse veebidisaini [40] realiseerimiseks st. disain mis reageerib vastavalt erinevatele ekraanisuurustele.

4.2.6 Ant Design ja Ant Mobile

Ant Design [8] on Reactile loodud kasutajaliidese teek. See sisaldab lisaks erinevatele stiiliklassidele ka kasutamiseks valmis Reacti komponente, mille peale esirakendi visuaalne pool üles ehitada. Kuigi see ei ole käsitletava projekti jaoks terviklik lahendus, on see hea baas mille pealt teenuse arendamist alustada. Programmeerijad

saavad keskenduda rohkem funktsionaalsuste implementeerimisele kui levinud komponentide loomisele.

Kuna Ant Design ei lähtu *mobile first* [41] kujundusprintsiipidest on mobiilvaadete jaoks kasutusele võetud sama firma poolt loodud Ant Design Mobile teek. See tähendab seda, et erinevalt näiteks populaarsest Bootstrapist [42] on Ant Designi komponentide loomisel keskendutud eelkõige lauaarvutikasutajatele.

Üldjuhul on kindlasti parem alustada mobiilvaadetest või kasutada selleks vastavat teeki, kuid Infrafly OMS-i puhul on tavavaatel ja mobiilvaatel väga selge funktsionaalne lahusus – tavavaade on mõeldud kontoritöölisele ning mobiilvaade ehitustöölisele. Võib üsna julgelt eeldada, et kontoritööline tegutseb peamiselt arvutis ning ehitustööline telefonis. Arvestades sellega pakub mainitud teekide valik isegi suuremat paindlikkust, kuna kummagi loomisel on lähtutud ainult funktsionaalsusele vastavatest seadmetest.

5 Tööriistad

5.1 Kommunikatsioon

Ülemaailmse eriolukorra tõttu väärivad mainimist ka mõned kommunikatsioonivahendid, kuna füüsiline kohalolek kontoris on olnud piiratud.

5.1.1 Atlassian Jira ja Confluence

Atlassiani Jira [43] on laialdaselt kasutuses olev tarkvara agiilsetele [44] arendustiimidele. Infrafly puhul kasutatakse peamiselt *Scrum boardi* [45], kuhu erinevad meeskonna liikmed ülesandeid kirja panevad. Iga nädal toimuvad sprindid, kuhu määratakse edasi ülesanded mis too nädal tehtud peaks saama. Iga nädala lõpus tehakse kokkuvõte tehtud tööst ja räägitakse järgmise nädala plaanidest.

Confluence [46] on tarkvara dokumentide haldamiseks. See võimaldab kõigil meeskonna liikmetel erinevaid dokumente luua nii, et need on ka teistele nähtavad ja muudetavad. Sealjuures on selgelt näha kes millised muudatused teinud on. Kuna Confluence on samuti Atlassiani toode on dokumente võimalik Jira ülesannetele külge panna, et teatud olukordades tööd paremini kirjeldada.

5.1.2 Zeplin

Selleks, et programmeerijate ja disainerite vaheline töö edeneks ladusalt on kasutusele võetud Zeplin [47]. Zeplin kogub disainerite poolt üles laetud kavandid ühte kohta, mis on tervele tiimile kättesaadav. Näha on kavandite ajalugu ja vajalikud muud failid (nt. pildid) ning programmeerijatele ka kujunduse elementide CSS omadused ja mõõtmed.

5.2 Arendus

5.2.1 Visual Studio Code

VSCode ehk Visual Studio Code [48] on Microsofti poolt loodud vabavaraline tekstiredaktor. Tänu suurele laiendite kogumile on tegemist väga paindliku ja mugava arendusvahendiga. Lisaks sellele oli autoril juba eelnev kogemus programmi kasutamisega, mis tegi tööprotsessi sujuvamaks.

5.2.2 Postman

Tagarakendi testimiseks kasutati Postmani [49]. Tihtipeale on vaja testida rakenduse funktsionaalsust spetsiifiliste tingimustega, mille jaoks oleks tülikas luua eraldi loogika. Postman võimaldab hõlpsasti saata ja salvestada erinevaid API päringuid.

6 Arendus

Selles peatükis kirjeldatakse täpsemalt diplomitöö fookuses olevat veebiteenust. Ülevaade on tehtud olulistest rakenduse osadest, mis anti töö autorile arendada.

6.1 Ülevaade teenuse osadest

Käsitletavad teenuse osad on jaotatud eraldi alampeatükkidesse lähtudes funktsionaalsusest. Koodinäitede puhul on suuremad failid vajadusel loetavad töö lisade sektsioonis. Kogu rakenduse talitus on keskendunud tellimuse olemi ümber.

6.1.1 Rollidepõhine autoriseerimine

Esmalt tasuks rääkida autoriseerimisest ja valitud neljast rollist. Esirakendis kajastub kasutaja roll peamiselt selles, millised vaated neile lubatud on. Operaatorid suunatakse sisse logimisel automaatselt mobiilvaatesse, mis on spetsiaalselt neile loodud, arvestades sellega, et ehitusplatsil olles on telefon juba üks nende töövahenditest. Tavavaade on neile keelatud. Logistikud ja raamatupidajad pääsevad mobiilvaatele ligi, kuid vajalikke toiminguid saavad nad teha ainult tavavaates vastavatel lehtedel. Võib öelda, et siin on kasutajaliidese koostamisel lähtutud murede lahususe printsiibist [50] (ingl *separation of concerns*). Selline vaadete lahusus aitab Infrafly OMS-i kui töövahendi loogiliselt kontori ja ehitusplatsi vahel ära jaotada, mis muudab kasutuskogemuse intuitiivsemaks.

Tagarakendis on rollide eesmärgiks kontrollida ja täpsustada kasutajale lubatud toiminguid. Kõik päritud andmed on lisaks rollile limiteeritud vähemalt firmasisestele kirjetele, kuna vastasel juhul tekiks suur konfidentsiaalsusrisk. Firmasiseseid andmeid saavad pärida kõik kasutajad, aga operaatorite puhul on see piiratud vaid neile vajalikele olemitele. Lisaks sellele on logistikutel õigus andmeid muuta ja kustutatuks märkida ehk „pehmelt kustutada“ [51]. See tähendab, et eemaldamise asemel lisatakse kirjele atribuut *isDeleted*. Niiviisi jääb kirje tegelikult andmebaasi alles, kuid seda ei kuvata esirakendis tavalisi päringuid tehes.

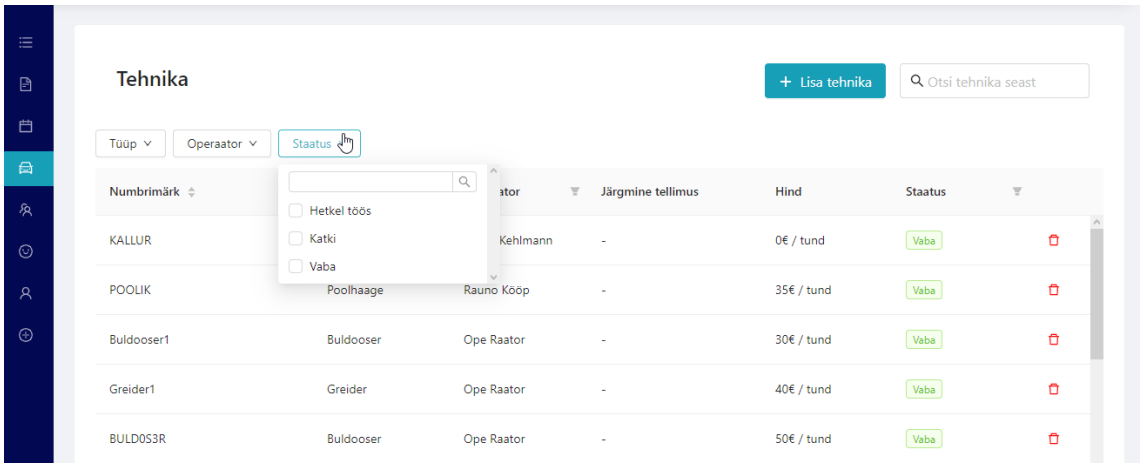
Nende õiguste läbi viimise aluseks on loodud API-sse vahevara, mis leiab läbi Firebase'i autenditud kasutaja tokeni [52] põhjal MongoDB kasutaja kirje ning määrab päringu parameetritesse vastavad õigused.

```
export async function getCurrentUserFromRequestHeader(request) {
  const authHeader = request.headers.authorization
  const token = await getTokenFromHeader(authHeader)
  const decoded = await decodeToken(token)
  const user = decoded.user
  if (user.uid) {
    user.firebaseUID = user.uid
    const mongoUser = await User.findOne({ $or: [{ email: user.email }, { oldId: u
ser.uid }] })
    user.mongoUID = mongoUser._id ? mongoUser._id.toString() : null
    delete user.uid
  }
  return user || null
}

export async function getCurrentUserDataFromRequest(req, res, next) {
  try {
    let currentUser = await getCurrentUserFromRequestHeader(req)
    if (currentUser && currentUser.isDeleted) {
      return res.status(statusCode.FORBIDDEN).json({
        error: 'USER_DELETED',
        mesage: 'The user has been deleted and does not have access.'
      })
    }
    req.currentUser = currentUser
    currentUser = currentUser || { isAdmin: false, isCompanyOwner: false, role: 'n
one' }
    req.canModifyAllCompanyData = currentUser.isAdmin || currentUser.isCompanyOwne
r || ['logistician'].indexOf(currentUser.role) !== -1
    next()
  } catch (e) {
    res.status(statusCode.BAD_REQUEST).json({
      error: e.message
    })
  }
}
```

Joonis 2. Rolle implementeeriv vahevara

6.1.2 Tabelid

Suur osa toote eesmärgist on pakkuda ülevaatlikku infot firmasiseste protsesside kohta. Seetõttu koosnevad ka paljud rakenduse lehed peamiselt tabelitest. Kuigi iga lehe tabel on teatud määral erinev jääb üldine lahendus suuresti samaks. Aluseks on võetud Ant Designi Table [53] komponent, mis kuvab arendaja poolt määratud tulpadesse andmeid ning vajadusel jaotab need mitme alamlehe peale laiali, et vältida ruumipuudust. Samuti võimaldab see komponent tulemusi tulpade põhjal järjestada.



Joonis 3. Tehnika sektsiooni tabel

Peamine osa tabelite funktsionaalsusest on kuvatud andmete filtreerimine. Esiteks on igale tabelile loodud otsingulahter, mis võimaldab tabelis kuvatavaid andmeid otsingufraasi järgi täpsustada. Juhul kui otsingust ei piisa on tabelile lisatud valik lisafiltreid, mis seavad tabelis olevale informatsioonile tingimusi. Sealjuures osade filtrite puhul on omakorda võimalik filtri valikuid otsingufraasiga täpsustada. Kõik filtrid peale otsingufraasi on talletatud sessioonipõhiselt EasyPeasy Store'i [15]. Kasutaja jaoks tähendab see seda, et kõik nende valitud filtrid kehtivad ka peale lehe taasavamist või uuendamist. See on vajalik, kuna tihtipeale on vaja andmeid teistest tabelitest kas muuta või kontrollida.

```
export const useFilters = key => {
  const filters     = useStoreState(state => state.filters.items[key])
  const saveFilters = useStoreActions(actions => actions.filters.save)

  const saveKeyFilters = useCallback((payload) => {
    saveFilters({ key, ...payload })
  }, [key, saveFilters])

  return [filters, saveKeyFilters]
}


const filterModel = {
  // states
  items: {
    calendar: { filtered: {}, pagination: { pageSize: 1000 } },
    orders: { filtered: {}, pagination: { pageSize: 40 }, sorted: {}, view: {} },
    worksheets: { filtered: {}, pagination: { pageSize: 40 }, sorted: {}, view: {}
 },
    equipment: { filtered: {}, pagination: { pageSize: 40 }, sorted: {} },
    contacts: { filtered: {}, pagination: { pageSize: 10 }, sorted: {} },
    peopleContacts: { filtered: {}, pagination: { pageSize: 10 }, sorted: {} },
  },

  // actions
  save: action((state, { key, ...payload }) => {
    state.items[key] = payload
  }),
  clear: action((state, placeKey) => {
    if (placeKey) {
      delete state.items[placeKey]
    } else {
      state.items = {}
    }
  }),
}
```

Joonis 4. useFilters konks ja vastava lokaalse andmehoidla struktuur ja funktsioonid

Filtreid kasutatakse koodis *useFilters* React.js konksu [54] (ingl *hook*) abil. See funktsioon võtab sisendiks võtmesõna, milleks on vastava tabeli olemi nimetus. Funktsiooni väljundiks on massiiv, mis sisaldab meetodit salvestamaks filtreid mainitud lokaalsesse andmehoidlasse ning viidet vastavatele talletatud filtritele. Andmehoidlas määratakse ka tabelite vaikimisi olek.

6.1.3 Vormid

Kuna tabelitesse on vaja andmeid käsitsi sisestada on igale tabelile loodud eraldi vorm. Kui välja arvata mõned väiksemad erinevused, siis nagu ka tabelite puhul on üldine lahendus siin valdavalt samasugune. Vormi sisu on üsna konkreetne. Erinevad elemendid jaotatakse veergudesse ja tulpadesse. Erinevad sisendid kapsuleeritakse *Form.Item* komponendi vahele, mis võimaldab *getFieldDecorator* funktsiooniga neile reegleid määrata ning hiljem reeglite alusel andmeid valideerida. Sisendite väärtuseid saab pärida *getFieldsValue* funktsiooniga ning vajadusel *setFieldsValue* funktsiooniga programmaatiliselt muuta. Eelnevad funktsionaalsused sisalduvad kõik Ant Designi teegis [55]. Nende abil on loodud omakorda rakendusele spetsiifilised sisendid.

```
        <Col span={12}>
          <Form.Item label={<FormattedMessage id={`saas.orders.form.fields.dateEnd
`} />}>
            <Row type="flex" justify="start" align="middle" gutter={8}>
              <Col span={16}>
                {getFieldDecorator('dateEnd', {
                  rules: [
                    { required: true },
                    { validator: validateDates }
                  ],
                })(
                  <DatePicker
                    disabledDate={endValue => {
                      const startValue = getFieldValue('dateStart')
                      if (!endValue || !startValue) {
                        return false
                      }
                      return endValue.valueOf() < startValue.valueOf()
                    }}
                    allowClear={false}
                    format={dateFormat}
                  />
                )}
              </Col>
              <Col span={8}>
                <TimePicker value={moment(dateEnd)} format={timeFormat} allowClear
={false} onChange={(value) => {
                  const mergedDate = moment(moment(dateEnd).format(dateFormat) + '
 ' + moment(value).format(timeFormat), datetimeFormat)
                  setFieldsValue({ dateEnd: mergedDate })
                }} />
              </Col>
            </Row>
          </Form.Item>
        </Col>
```

Joonis 5. Kuupäeva valiku sisend vormis

Põhiline eripära siin seisneb vormi andmete talletamises. Nagu eelnevalt mainitud on vormi sisestatud andmed ühes sessioonis püsivad ehk muudatuste tegemisel salvestab vormi *onChange* funktsioon *setAdding* funktsiooni abil andmed vastavasse objekti lokaalses andmehoidlas. Seda objekti kasutades on võimalik ka sisendeid eeltäita, kui on teada, et sisendi väärtuse muutmine on ebatõenäoline. Hea näide sellest on töölehe lisamise vormis olev alguskuupäeva sisend. Lisamise vormi lehe avamisel jooksutatakse Reacti *useEffect* [56] konks, mis määrab *setAdding* funktsiooni abil loodavale töölehele hetkelise kuupäeva ja aja. Uue kirje edukal lisamisel jooksutatakse *clearAdding* funktsioon, mis tühjendab *adding* objekti ja seega ka vormi väljad.

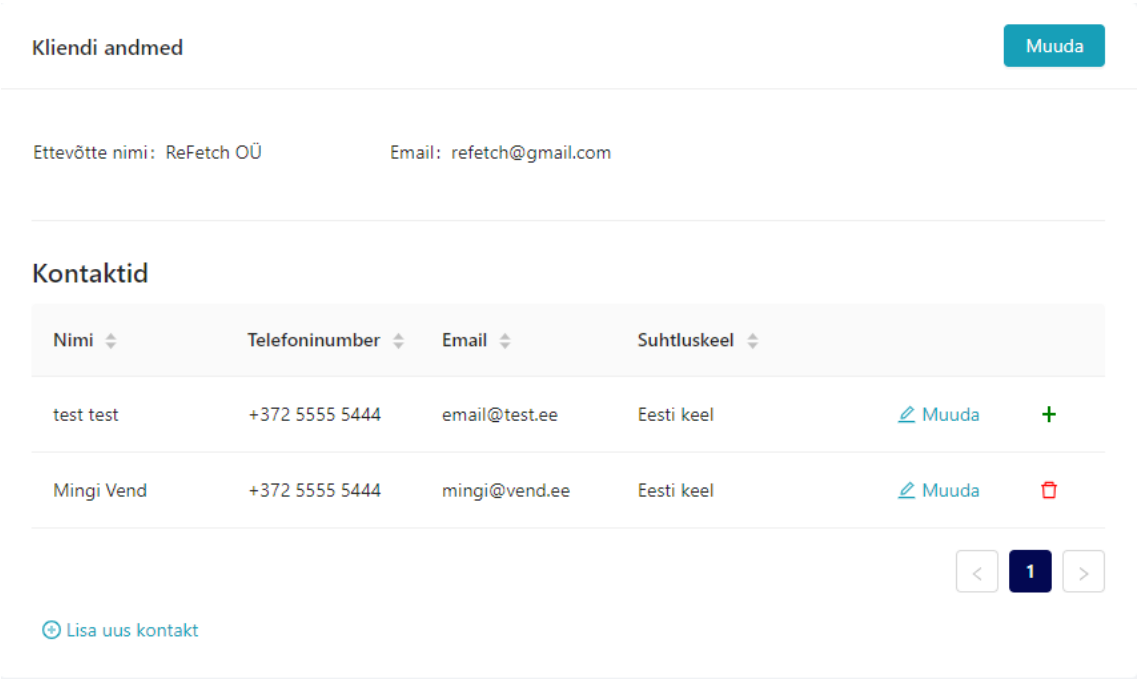
```
  adding: {},
  editing: {},
  error: {},

  setAdding: action((state, { ...payload }) => {
    for (const key of Object.keys(payload)) {
      if (['durationHours', 'weight', 'distance'].indexOf(key) !== -1) {
        payload[key] = parseFloat(payload[key])
      }
      if (['cypher'].indexOf(key) !== -1 && payload[key]) {
        payload[key] = payload[key].toUpperCase()
      }
    }

    state.adding = _.mergeWith(state.adding, payload, (objValue, srcValue
) => {
      if (_.isArray(objValue)) {
        objValue = _.uniq(srcValue)
        return srcValue
      }
    })
  }),
  clearAdding: action((state, { keys } = {}) => {
    if (keys && keys.length) {
      for (const i in keys) {
        delete state.adding[keys[i]]
      }
    } else {
      state.adding = {}
    }
  }),
```

Joonis 6. Töölehe mudel andmehoidlas

6.1.4 Personali, tellimuste, klientuuri ja tehnika haldamine

Kasutades eelnevalt mainitud tabelit ja vormi on loodud vaated firmasisese personali käsitlemiseks. Kuigi need vaated ei ole ehk ettevõtte igapäevases talituses nii palju kasutusel on nad kahtlemata olulised. Kõigi firma tööliste kasutajad lisatakse siin. Uute klientide puhul lisab esmase logistiku rollis oleva kasutaja Infrafly poolne korrespondent. Edaspidised kasutajad saab logistik juba ise luua. Samuti on tal õigus tagantjärele loodud kasutajate kontaktandmeid muuta.

Sama ülesehitust kasutades on loodud vaated ettevõtte klientide haldamiseks. Tabelis on kuvatud ettevõtte nimi, aktiivsete tellimuste arv ning maksevõimekus. Tabeli reale klikkides avaneb personalivaatele analoogne tabel kuhu saab lisada kontaktisikuid. Siin lisatud kliendid ja kontaktid ilmuvad valikutena tellimust luues või filtreid kasutades. See tabel on ka üks kohtadest kus logistikul on võimalik andmeid kustutatuks märkida või uuesti aktiveerida. Selline kustutamise talitus aitab tellimuse valikutes vaid aktuaalseid kliente kuvada, säilitades samaaegselt ülevaate ka endistest klientidest.



Joonis 7. Kontakti detailvaade kontaktisikute tabeliga

Kolmas ja viimane vaadete kogum mida võib eelnevatega samalaadseks lugeda on ettevõtte masinate ehk tehnika sektsioon. Siinne ülesehitus on taas kord analoogne kahele eelnevale. Küll aga on juba realiseeritud mõned lisafiltrid. Masinaid saab

filtreerida masina tüübi, operaatori ja tööstaatuse järgi. Tabeli reale klikkides saab muuta erinevaid parameetreid vastavalt masina tüübile ning määrata API funktsioonide abil geograafilise asukoha. Selles tabelis on samuti võimalik kirjeid pehmelt kustutada ja taas aktiveerida.

Viimased kaks sektsiooni: tellimused ja töölehed on samuti välimuselt esimestele väga sarnased. Peamine erinevus seisneb selles, et esimestes hallatavad kirjed on mõeldud valikuteks tellimuste loomisel ning töölehed on mõeldud tellimusi komplementeerima informatsiooniga tehtud tööde kohta.

Kõik vaated lähtuvad *RESTful* [57] veebiteenuse printsiipidest. See tähendab, et erinevatele toimingutele vastavad aadressid on iga sektsiooni puhul sama kujuga. Näiteks uue masina lisamise puhul on selleks */equipment/add/* ning töölehe lisamise puhul */worksheet/add/*. Iga päringuga käivad kaasas selle läbi viimiseks vajalikud andmed. Samuti on sessioonipõhiselt talletatav info hoitud ainult kliendi (brauseri) pool.

6.1.5 Töölehed ja digitaalne allkirjastamine

Kuigi tellimus on vaieldamatult käsitletava rakenduse ülesehituse tuumik, on seda täiendav tööleht firma tööprotsesside jaoks ehk veel olulisem. Tellimus määrab ehitusobjektil tehtavad tööliigid, asukoha ja kuupäevad ning masinad ja nende operaatorid, kuid töölehed, mis luuakse operaatorite poolt, kirjeldavad tehtud töid ja nende parameetreid. Firma aruanded luuakse peamiselt just selle informatsiooni põhjal. Töölehtedes talletatav informatsioon on välja toodud järgnevas tabelis.

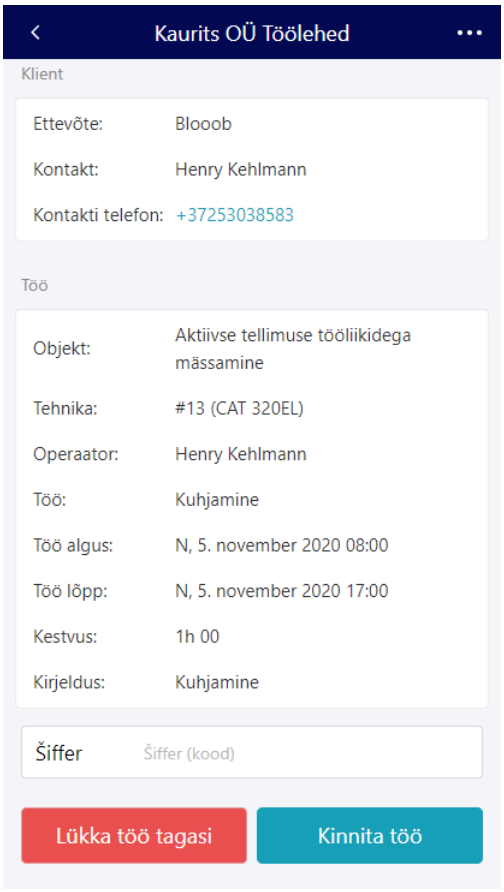
Tabel 4. Töölehe väljad

Väli	**Kirjeldus**	**Kohustuslik**
Objekt *supplierOrder*	Vastava tellimuse ehitusobjekti nimetus	Jah, eeltäidetud
Töö *workType*	Töö liik, millele töölehte koostatakse	Jah, eeltäidetud
Töö algus / lõpp *dateStart/dateEnd*	Käsitletava töö algus- ja lõppkuupäev / aeg	Jah, eeltäidetud hetkelise kuupäevaga kella 08:00st 17:00ni

Tonnid *weight*	Töödeldud tonnide arv kui see on tehtud töö põhjal märgitav	Oleneb tööst
Kilomeetrid *distance*	Läbitud kilomeetrite arv kui see on tehtud töö põhjal märgitav	Oleneb tööst
Kestvus *durationHours*	Töö kestvus tundides	Oleneb tööst
Kirjeldus *description*	Lisainformatsioon töö kohta	Jah (igaks juhuks, eristamaks samaliigilisi töid)

Töölehed luuakse peamiselt operaatorite poolt neile loodud mobiilvaadetes, millest tuleb rohkem juttu peatükis 6.1.7. Erandjuhtudel, kui operaator on näiteks unustanud töölehe luua, saab logistik tavavaates seda tagantjärele teha.

Loodud töölehtede kohta saab tellimuse klientettevõtte kontaktisik SMS-i kaudu automaatse teavituse, et tööleht ootab temapoolset kinnitust. Selleks kasutatakse peatükis 4.1.4 mainitud Nexmo teenust. Sõnumis sisalduvale spetsiaalsele lingile vajutades avaneb vaade, kus on näha sisestatud töölehe informatsioon ning osades ehitusfirmades kasutusel olev valikuline šifri lahter. Klient kas kinnitab töö ja töölehe loomise protsess lõpeb või ta lükkab selle tagasi ja tööd teinud operaator saab omakorda vastava SMS teavituse.



Joonis 8. Töölehe kinnitamise vaade

Infrafly poolt pakutavates digitaalsetes töölehtedes realiseeruvad tegelikult lahendused kõigile kolmele projekti alguses seatud probleemile. Võimaldades ehitustöölistel oma tööd mobiiltelefoni abil otse andmebaasi sisestada kaob vajadus vastavate paberite ja nende kokku kogumise järele. Samuti on kohe sisestatud andmete õigsus kindlam kui päeva lõpus täidetud dokumendid. Lisaks sellele ei ole töölisel vaja otsida platsilt tellimuse kontaktisikut. Töö osapooled viib kokku süsteem.

6.1.6 Eksportimine

Loodud tellimusi ja töölehti on võimalik eksportida kahes erinevas formaadis: PDF ja XLSX. Kummagi formaadi puhul on võimalik valida mis tahes arv kirjeid, millest süsteem genereerib vastava dokumendi. Mõlemal juhul on selleks esmalt tellimuse ja seejärel tööliikide järgi grupeeritud töölehtede väljavõte. Mõlemat tüüpi dokumendid koostatakse klientrakenduse poolel.

Eksporditavate PDF failide koostamisel on kasutatud React-pdf [58] teeki. Selle jaoks on loodud *BuildPDF* komponent, mis võtab parameetriteks kas tellimuste või töölehtede

ID-de massiivi, valitud kuupäevade vahemiku ja failinime. *Builder* komponent laetakse dünaamiliselt, kasutades Next.js-i *dynamic import* [59] funktsiooni ehk alles siis kui sessioonis eksporditakse esimene dokument. See lahendus parandab märkimisväärselt eksporditavate andmetega tabelite laadimisaega, kuna tegemist on võrdlemisi mahuka failiga.

```
import React, { useState } from 'react'
import { UploadOutlined } from '@ant-design/icons'
import { Button } from 'antd'
import { useIntl } from 'react-intl'
import dynamic from 'next/dynamic'

const Builder = dynamic(() => import('./BuildPDF'))

export const PDFLink = ({ fileName, orderIDs, worksheetIDs, dates, ...prop
s }) => {
  const [loading, setLoading] = useState(false)
  const intl = useIntl()

  return (
    <Button loading={loading} onClick={() => setLoading(true)} {...props}>
      {loading ?
        <Builder
          fileName={fileName}
          orderIDs={orderIDs}
          worksheetIDs={worksheetIDs}
          dates={dates}
          linkProps={{ onClick: () => setLoading(false) }} /> :
        <><UploadOutlined /> {intl.formatMessage({ id: 'saas.pdf.export' }
)}</>}
    </Button>
  )
}
```

Joonis 9. PDFLink komponent, mis kasutab PDFBuilder komponenti

ID-de põhjal päritakse andmebaasist vastavad kirjed, mille põhjal koostatakse React-pdfi dokument. *BuildPDF*-is sisalduv *build* funktsioon filtreerib nende kirjete ja sisestatud kuupäevade põhjal välja vastavad töölehed, mis saadetakse edasi *exportData* funktsioonile. *exportData* funktsioon koostab valitud kirjete põhjal React-pdf dokumendi ja sellest omakorda *blob* tüüpi objekti, kasutades React-pdfi *toBlob* funktsiooni. Seejärel salvestatakse *blob* FileSaver [60] teegi abil lokaalse failina,

kasutades parameetrites määratud failinime ning eksporditava failitüübi info saadetakse analüütikale. *BuildPDF* komponendiga saab tutvuda diplomitöö Lisade osas.

```
const exportData = async ({ orders, workTypes, intl, fileName }) => {
  const data = <PDFDocument orders={orders} workTypes={workTypes} intl={in
tl} />
  const blob = await pdf(data).toBlob()
  saveAs(blob, `${fileName}.pdf`)
  analytics.track('Export', {
    'Export format': 'PDF'
  })
}
```

Joonis 10. exportData funktsioon

PDFDocument komponent koosneb valdavalt PDF faili kujunduse realiseerimisest React-pdfi poolt pakutavate primitiivide ja stiilimiseks mõeldud API abil. Stiilimine põhineb valitud CSS omadustel ning *Flexbox* [61] tüüpi paigutusel. Mitme tellimuse seast töölehtede valimisel algab PDF-i puhul iga tellimuse sektsioon uuelt lehelt ning võib vajadusel jätkuda ka järgnevatel lehtedel.

XLSX-i puhul on loodud komponent *BuildXLSX*, mis käitub üldjoontes täpselt samamoodi nagu *BuildPDF*. Erinevus seisneb selles, et dokumendi koostamiseks kasutatakse SheetJS [62] teeki. Kuna Exceli puhul erilist kujundust luua vaja ei ole, toimub dokumendi genereerimine kahetasandiliste massiivide abil. Funktsioon *worksheetData* võtab parameetriteks töölehed, vastava töölehe hinna ühiku ja intl objekti. Töölehtede põhjal tehakse esmalt rekursiivsete funktsioonide abil kokkuarvutused. Seejärel käiakse nad map funktsiooni abil ükshaaval läbi, luues igale lehele vastava massiivi. Hinna ühik määrab selles massiivis sisalduvad andmed. Tulemuseks on uus massiiv, mis sisaldab iga töölehe ja lõpuks kokkuarvutuste massiive. *WorksheetData* funktsiooniga saab tutvuda diplomitöö Lisade osas.

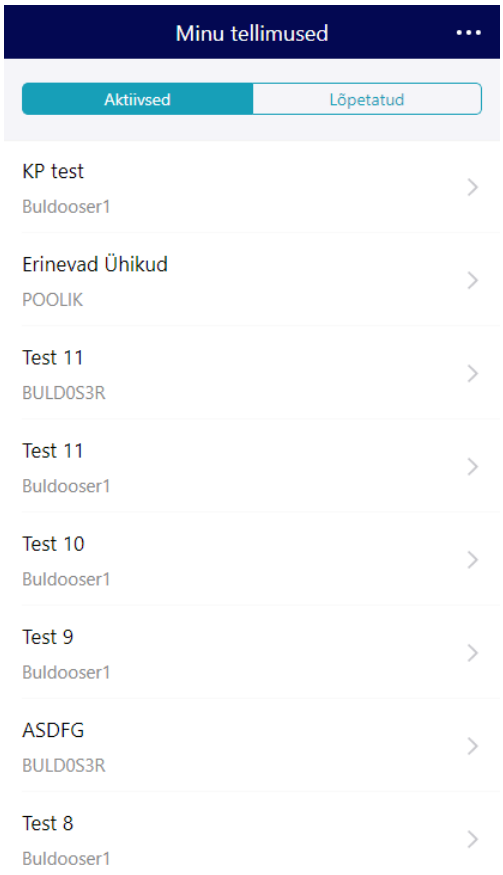
Oma olemuselt võib neid komponente pidada täiesti eraldiseivateks React rakendusteks, kuna nad täidavad ainult töölehtede väljavõtete koostamise funktsiooni ja ei sõltu ülejäänud rakenduse sessioonist. Tänu sellele on nende komponentide loogika vajadusel kasutatav mis tahes eesrakendi vaates ning ka tagarakendis, kui tulevikus peaks tekkima vajadus väljavõtteid serveri poolel koostada (nt. regulaarselt kasutajale

saadetavad raportid). Mõlemate eksporditavate failide näidistega saab tutvuda töö lõpus Lisade osas.

6.1.7 Mobiilvaated

Mobiilvaateid võib Infrafly OMS-is lugeda spetsiaalseteks operaatori vaadeteks. Need lehed on loodud, pidades silmas operaatori rollis olevate kasutajate vajadusi. Erandiks siin on töölehtede allkirjastamise vaade, mis on vastava lingiga ligipääsetav ka autentimata kasutajale (ehitustööde kliendile). Nende vaadete funktsionaalsus on keskendunud töölehtede loomisele.

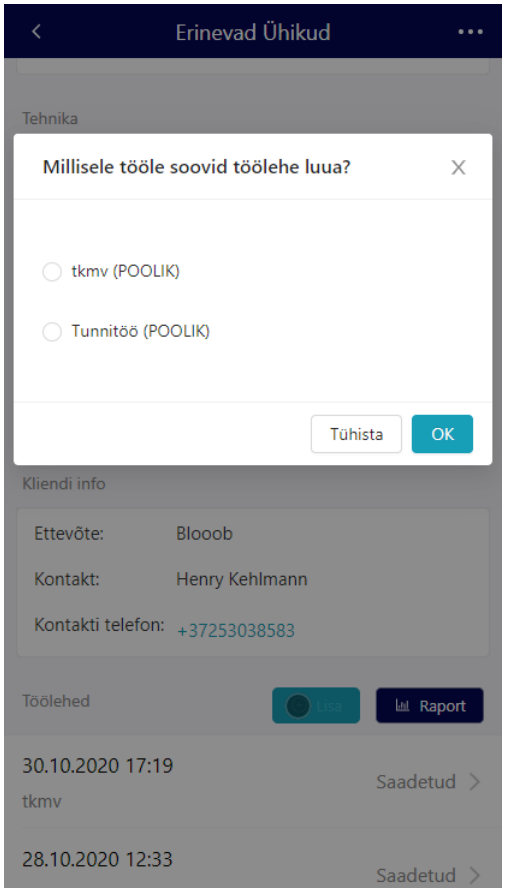
Sisse logides avaneb esmalt vaade „Minu tellimused“, kus on näha loetelu kasutajaga seotud tellimustest. Valida on võimalik aktiivsete ja lõpetatud tellimuste vahel.



Joonis 11. Minu tellimuste vaade

Valides loetelust tellimuse näeb täpsemat infot ehitusobjekti, tehnika ja kliendi kohta. Vaate alumises osas on töölehtede sektsioon, kus toimuvad operaatori peamised tegevused: töölehe lisamine ja tellimuse lõpetamine. Klikkides „Lisa“ nupule avaneb

modaalaken tööliigi valikuga. Iga tööliigi nimetuse taga on sulgudes masina registrinumber, millele tööliik kuulub. Valides tööliigi suunatakse kasutaja töölehe lisamise vaatesse, kus on eeltäidetud objekti nimetuse- ja modaalakna valiku põhjal tööliigi lahtrid. Kuigi tööliiki saab lisamise vaates uuesti valida, aitab modaalaken rohkem tähelepanu pöörata õigele tööliigi valikule, mille põhjal hiljem väljavõtteid tehes töölehti grupeeritakse. Töölehe lisamise ja muutmise vormi kood on analoogne peatükis 6.1.3 mainitud vormidele, kuid kasutab Ant Designi asemel Ant Mobile'i komponente.



Joonis 12. Tööliigi valik

Kliendi poolt kinnitamata töölehti on võimalik muuta lisamisele analoogses vaates. Sedaviisi töölehte muutes saadab süsteem uuesti kliendile SMS teavituse.

6.1.8 Tõlked, Automaattestimine

Siin osas on välja toodud väiksemad funktsionaalsused, mis autori arvates eraldi peatükki ei vääri. Need funktsionaalsused ei ole ilmtingimata omavahel seotud.

Rakenduses on implementeeritud React Intl [63] teegi abil tõlkesüsteem. See lubab mis tahes vaates veebilehe sisu valikule vastavasse keelde tõlkida. Koodi kirjutamise vaatepunktist tähendab see seda, et kõik rakenduses kuvatav tekst mis ei ole universaalne (nt. numbrid, sümbolid) kasutab kas React Intl poolt pakutavat *FormattedMessage* komponenti või *formatMessage({ id })* funktsiooni. Tekst ise tuleb keele valikule vastavast JSON failist, kus kirjed on vaadete ja otstarvete järgi alamobjektidesse grupeeritud.

Mida suuremaks kasvab teenus, seda tähtsamaks muutub automaattestide olemasolu. Hetkel on Infrafly OMS-is realiseeritud Jest [64] raamistiku abil mõned lihtsamad *snapshot* [65] tüüpi testid. Need testid kontrollivad kas veebilehe visuaalne sisu (kompileeritud HTML kood) vastab projekti salvestatud referentsfailidele ehk *snapshotidele*. Testid jooksutatakse automaatselt uute arenduste üleslaadimisel GitHubi keskkonda. Uuendusi ei ole võimalik kinnitada enne kui testid läbivad. Visuaalsete uuenduste põhjal on tarvis ka *snapshote* uuendada ning need koos arendusega kaasa saata. Selline lahendus aitab üldiselt vältida kiire arendustsükli käigus tekkivaid väiksemaid vigu, kuid potentsiaalselt ka suuremaid kuna äriloogika muutumise korral muutuvad ka kuvatavad väljundid ning testi jooksutamine tuvastab erinevuse.

```
import React from 'react'
import OrdersPage from '../pages/orders/index'
import renderer from 'react-test-renderer'

import model from 'services/store/index'

Object.defineProperty(window, 'matchMedia', {
  writable: true,
  value: jest.fn().mockImplementation(query => ({
    addEventListener: jest.fn(),
    removeEventListener: jest.fn(),
    dispatchEvent: jest.fn(),
  })),
})

test('Desktop orders list view renders correctly', () => {
  const app = withIntl(<OrdersPage />, model)
  const component = renderer.create(app)

  let tree = component.toJSON()
  expect(tree).toMatchSnapshot()
})
```

Joonis 13. Tellimuste tabelvaate testide fail

6.2 Publitseerimine

Et rakendus oleks kõigile kasutajatele kättesaadav tuleb see veebi publitseerida. Infrafly OMS-i esirakend on publitseeritud läbi Verceli [66] teenuse. See teenus haldab lisaks veebiserverile ka domeene. Domeenidele saab publitseerida korraga versioonihaldussüsteemi Giti [67] erinevaid harge, kasutades selleks CNAME [68] viiteid. Käsitletava teenuse puhul on loodud erinevad aadressid arendus- ja produktsioonkeskkondadele. Uute arenduste üleslaadimisel GitHubi keskkonda ehitab Verceli server vastavale harule ja domeenile automaatselt uue publikatsiooni.

Kuna Vercelis hoitakse vaid esirakendit on tagarakend publitseeritud Heroku platvormil eraldi domeenil. Esirakend suhtleb andmebaasiga läbi Herokus kasutataval aadressil asuva API-ga. Andmebaasi ise hoitakse MongoDB Atlas [69] platvormil. Tegemist ei ole renditava virtuaalmasinaga, vaid andmebaasiteenusega ehk DBaaS-iga [70] (ingl *Database as a service*). See tähendab, et andmebaasiserveri haldamine on jäetud teenusepakkujale ning makstakse hoopis kasutatavate ressursside eest.

7 Tulemused

Siin osas on antud autoripoolne hinnang tehtud arendusele ja loodud süsteemile. Süsteemi hinnangu osas on välja toodud mõned võrdluspunktid taustauuringus käsitletud rakendustega.

7.1 Hinnang arendusele

Üldiselt võib öelda, et tehtud arendus kulges edukalt. Eesmärk oli kolme kuu jooksul välja arendada minimaalne kasutatav toode ehk MVP, mis pakuks lahendusi töö alguses püstitatud kolmele peamisele probleemile. Autori hinnangul jõuti selleni umbes nädal aega enne seatud tähtaega, mis võimaldas toodet ka pisut põhjalikumalt testida ja mõned olulised vead tuvastada. Peale seda anti Infrafly OMS-i ligipääs esimesele kliendile. Ilmselt tasuks siin veel välja tuua, et esimese kliendi ja partnerettevõtte puhul ei ole tegemist sama firmaga.

Iganädalane arendustsükkel oli üsna hästi modereeritud. Tootetiimi koosolekud toimusid kaks korda nädalas – teisipäeval ja reedel. Teisipäeval seati paika uue sprindi ülesanded ja reedel võeti tehtud töö kokku ning arutati mis võiks jutuks tulla järgmisel teisipäevasel koosolekul. Nädalavahetus ja esmaspäev oli nn. testimisperiood, kus tootetiimi liikmed ise välja arendatud funktsionaalsuste toimivust ja kasutuskogemust kontrollisid. Testides ei lähtutud ainult enda spekulatsioonidest, vaid aeg ajalt konsulteeriti ka partnerettevõtteks oleva ehitusfirmaga, et paremini aru saada milline võiks oodatav programmi kasutus olla. See lahendus oli üldiselt üpris tõhus – paljud arenduse käigus tekkivad vead tuvastati juba samal nädalal. Testijateks ei olnud mitte ainult arendajad, vaid ka Infrafly ja partnerettevõtte kontoritöötajad, kes lähtusid testimisse teisest vaatepunktist. Samuti aitas tihe modereerimine planeeritavad funktsionaalsused jõukohasteks ja loogiliselt grupeeritud ülesanneteks jaotada, et neid koosolekutel paremini hinnata. Siin tuleks veel märkida, et osad vead ilmnevad alles stabiilses kasutuses ehk täiuslik lahendus see ei olnud.

Suurimaks väljakutseks osutus korraliku ja toimiva koodi kirjutamine. Peamiselt oli diplomitöö autoril sellega probleeme projekti alguses. Kuigi planeeritavad funktsionaalsused olid paigas ja ülesanded selged, ei olnud autor kasutatavate tehnoloogiatega piisavalt mugaval tasemel, et täielikult hea koodi kirjutamisele

keskenduda. Esialgu oli suurem fookus just tehnoloogiate kasutamise mõistmisel, mis nõudis ka teatud määral töötundidevälist iseseisvat õppetööd. Iteratsioonid olid kiired ja tähtis oli hoida tempot, et esimese kliendiga kokku lepitud kuupäevaks oleks toode valmis. Peamisteks materialideks olid siin vastavate raamistike dokumentatsioonid. Kuna materiali oli palju ning efektiivse tööprotsessini jõudmiseks oli vaja osata neid koos kasutada, siis võib öelda, et kõige olulisem autoripoolne areng siin seisneski dokumentatsioonide töötlemisoskuses. Mida rohkem aeg edasi läks, seda kergemaks muutus vajaliku informatsiooni välja noppimine. Õnneks hoidis eelnevalt mainitud töönädala struktuur autori ka esimestel nädalatel piisavalt produktiivsena, et projekti reaalset väärtust anda. Sellega aitas palju ka arendajatevaheline koodi kontrollimine. Ükski uuendus ei läinud üles enne kui teine arendaja oli selle kooskõlastanud. Vigade esinemise korral anti autorile asjakohast tagasisidet, mille põhjal oli kergem neist õppida. Peale esimest kuud oli tekkinud juba teatud mugavus, mida alustades ei olnud ja kirjutatava koodi kvaliteet oli jõudnud piisavale tasemele, et sujuvalt uusi ülesandeid juurde võtta.

7.2 Hinnang süsteemile

Tänu partnerettevõttega konsulteerimisele on valitud andmebaasi olemid asjakohased. See tähendab, et ootuspäraselt rakendust kasutades saab valdava osa ehitusfirma tööprotsesside jaoks vajaliku info süsteemi talletada. Kliendid võivad talletatud andmed paberile eksportida, et neid arhiveerida, aga otsest vajadust selleks ei ole. Erandiks siin on raamatupidaja rollis olevad kasutajad, kes peavad arveid siiski käsitsi koostama, kuid selleks vajalikud andmed komplekteerib süsteem. Tulevikus tasuks kindlasti mõelda ka selle funktsionaalsuse implementeerimisele. Hetkel on rakendus valdavalt suunatud logistikule, kuid ideaalis võiks Infrafly OMS täita ka raamatupidaja tööülesanded nagu seda teeb taustauuringus käsitletud Texada. Intuitiivsema kasutuskogemuse tõttu tuleks kasutegur palju kiiremini välja, kuna nii Infrafly kui ka klientettevõte ei peaks kulutama ressursse töötajate väljaõpetamisele.

Sisestatud andmete õigsuse probleemi puhul tuleb vaadata operaatorite poole. Esimestelt klientidelt saadud tagasiside põhjal selgus, et paljud neist olid väga rahul loodud lahendusega. Kasutades mobiiltelefoni ja neile loodud vaateid said masinajuhid tehtud tööd kohe õigesse kohta sisestada. Enne rakenduse kasutusele võtmist tehti seda tööpäeva

lõpus ehk igasugused varem kirja pandud andmed vajasid õigesse dokumenti ümberkirjutamist. Mitmel ümberkirjutamisel või mälust lähtumisel on palju suurem võimalus andmete õigsuses eksida. Veel rohkem oldi rahul sellega, et ei olnud enam tarvis ehitusplatsilt otsida kontaktisikut kes tööd kinnitaks ehk operaator ei vastutanud enam temaga kontakteerumise eest. Teavitustega sõnumid lähevad välja automaatselt ning kommunikatsiooniahel muutus palju efektiivsemaks.

Küll aga ilmnes siin ilmselt arendatud süsteemi suurim probleem – veakindlus. Nimelt jäi rakendusse lühikese ja kiire arendusperioodi tõttu paar viga, mis moonutasid sisestatud töölehti. Kuigi operaatorid olid rakendust ootuspäraselt kasutanud läks osa informatsioonist ikkagi kaotsi ehk püstitatud probleem kerkis tootetiimile teadmata ajutiselt uuesti üles. Kuigi see viga sai lahendatud oleks see olukord tagantjärele mõeldes täiesti välditav olnud. Vigade tekkimine rakendust arendades on paratamatu, kuid kui API poolel oleks olnud tehtud vastavad veakindlusarendused ei oleks vigaseid töölehti saanud isegi sisestada ning probleem oleks juba esimesel ettejuhtumisel lahenduse leidnud. Nagu näha siis implementeeritud *snapshot* testidest ei olnud sel puhul kasu. Enne järgmisi suuremaid arendusi võiks autori meelest implementeerida Infrafly OMS-i põhjalikuma automaattestide süsteemi, kuna taoliste vigade tagajärgede parandamine võib osutuda kordades suuremaks katsumuseks kui igasugused uued arendused.

Vigadel peatudes võiks puudutada ka tulevikus potentsiaalselt esinevaid probleeme. Hetkel on loodud süsteem püstitatud kriteeriumite mõistes üsna töökorras, aga tuleb silmas pidada, et andmebaasis hoitavate kirjete kogus on suuremate teenustega võrreldes minimaalne. Kindlasti tuleks viia igasugune eksportimise loogika tagarakendisse. Kuna MongoDB ID-d on üsna pikad kirjed, siis ühel hetkel tekib töölehti lihtsalt nii palju, et praegused filtreeritud tabelist ID-de valimise teel tehtavad päringud ei lähe enam läbi. Ideaalis võiks ekspordid luua hoopis esirakendist valitud filtrite põhjal, nagu taustauuringus mainitud Navireci dünaamilised graafikud, mille abil tehakse API poolel agregaatpäringud ja saadetakse tagasi koostatud fail.

8 Kokkuvõte

Diplomitöö tulemusena valmis kasutatav süsteem, mis üldjuhul lahendab kõik kolm peamist püstitatud probleemi. Tehnoloogiate valikul ja algsete funktsionaalsuste määramisel võeti arvesse taustauuringus leitud rakendusi ja reaalse ehitusettevõtte töötajate tähelepanekuid.

Kuigi loodud prototüüp oli kohati vigane tõestasid esimesed kasutajad esiteks, et tegemist on täiesti kasutatava lahendusega ning teiseks, et toote vastu on vähemalt Eesti turul piisaval määral huvi, et seda edasi arendada. Valminud prototüüpi kasutavad nüüdseks kolm maksvat klienti ning kaks testperioodil olevat ettevõtet. Prototüübi loomise peamine eesmärk oli jõuda võimalikult kiiresti punkti, kus saaks hakata koguma klientide käest tagasisidet, mille põhjal toote edasist arengut planeerida. Selleks implementeeriti juba alguses põhjalik analüütikasüsteem, mis logib nüüd praeguste klientide igapäevaseid kasutusharjumusi. Samuti hoitakse nendega tihedat kontakti, et teada saada mis on nende ootused ja ettekäänded arendatud tarkvarale. Selleks et Infrafly oma lõplike eesmärkideni jõuaks on tarvis veel palju tööd teha, aga valminud prototüüp on kahtlemata asjakohane ja oluline samm ettevõtte tuleviku perspektiivides.

Diplomitööd koostades ja prototüüpi arendades süvendas autor oma oskusi vastavates tehnoloogiates ja arenduskeeltes. Lisaks sellele sai ta väärtuslikke teadmisi reaalsete ettevõtete tööprotsesside ning veebiteenuste kavandamise kohta. Samuti arenes autori oskus töödelda vastavaid materiale. Kuigi käsitletava töö eesmärgiks oli välja arendada spetsiifiline veebirakendus on võimalik sarnase ülesehituse ja metoodikaga ka teistsuguseid rakendusi luua.

Kasutatud kirjandus

[1] "Muuda töölehed sekunditega arveteks | InfraFly OMS," [Online]. Available: https://oms.infrafly.ee. [Accessed 22 08 2020].

[2] "Infrafly Marketplace," [Online]. Available: https://infrafly.ee.

[3] „React Native,“ [Võrgumaterjal]. Available: https://reactnative.dev/.

[4] "Node.js," [Online]. Available: https://nodejs.org/en/. [Accessed 22 08 2020].

[5] "Express - Node.js web application framework," [Online]. Available: https://expressjs.com/. [Accessed 22 08 2020].

[6] "The most popular database for modern apps | MongoDB," [Online]. Available: https://www.mongodb.com. [Accessed 22 08 2020].

[7] "React – A JavaScript library for building user interfaces," [Online]. Available: https://reactjs.org. [Accessed 05 09 2020].

[8] "Ant Design - The world's second most popular React UI framework," [Online]. Available: https://ant.design. [Accessed 13 09 2020].

[9] "What is Agile Software Development? | Agile Alliance," [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed 15 11 2020].

[10] "Cloud Computing Architecture - javatpoint," [Online]. Available: https://www.javatpoint.com/cloud-computing-architecture. [Accessed 29 08 2020].

[11] "Cloud Application Platform | Heroku," [Online]. Available: https://www.heroku.com. [Accessed 29 08 2020].

[12] "Managed MongoDB Hosting | Database-as-a-Service | MongoDB," [Online]. Available: https://www.mongodb.com/cloud/atlas. [Accessed 29 08 2020].

[13] "Firebase," [Online]. Available: https://firebase.google.com. [Accessed 29 08 2020].

[14] "State and Lifecycle – React," [Online]. Available: https://reactjs.org/docs/state-and-lifecycle.html. [Accessed 29 08 2020].

[15] "Easy Peasy," [Online]. Available: https://easy-peasy.now.sh. [Accessed 29 08 2020].

[16] "What is NoSQL? NoSQL Databases Explained | MongoDB," [Online]. Available: https://www.mongodb.com/nosql-explained. [Accessed 29 08 2020].

[17] "What Is a Relational Database | Oracle," [Online]. Available: https://www.oracle.com/database/what-is-a-relational-database/. [Accessed 29 08 2020].

[18] "Structure your Data for MongoDB," [Online]. Available: https://docs.mongodb.com/guides/server/introduction/. [Accessed 29 08 2020].

[19] "What Is Cloud Computing? A Beginner’s Guide | Microsoft Azure," [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/. [Accessed 29 08 2020].

[20] "Advantages of NoSQL | MongoDB," [Online]. Available: https://www.mongodb.com/nosql-explained/advantages. [Accessed 29 08 2020].

[21] "Geospatial Queries — MongoDB Manual," [Online]. Available: https://docs.mongodb.com/manual/geospatial-queries/#geospatial-data. [Accessed 29 08 2020].

[22] "Overview | Places API | Google Developers," [Online]. Available: https://developers.google.com/places/web-service/overview. [Accessed 29 08 2020].

[23] "Difference Between Development, Stage, And Production - DEV," [Online]. Available: https://dev.to/flippedcoding/difference-between-development-stage-and-production-d0p. [Accessed 29 08 2020].

[24] B. J. Olsson H.H., "Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation," in *Software Business*, 2015.

[25] "Segment | Customer Data Platform (CDP)," [Online]. Available: https://segment.com. [Accessed 29 08 2020].

[26] "Vonage API Developer," [Online]. Available: https://developer.nexmo.com/messaging/sms/overview. [Accessed 29 08 2020].

[27] [Online]. Available: https://opensource.com/resources/what-open-source. [Accessed 05 09 2020].

[28] "Angular SPA: Why Single Page Applications?," [Online]. Available: https://opensource.com/resources/what-open-source. [Accessed 05 09 2020].

[29] "What is a web server? - Learn web development | MDN," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server#:~:text=A%20static%20web%20server%2C%20or,application%20server%20and%20a%20database.. [Accessed 05 09 2020].

[30] "Arendussõnastik," [Online]. Available: https://agiil.github.io/sonastik/#asis. [Accessed 05 09 2020].

[31] "Introducing JSX – React," [Online]. Available: https://reactjs.org/docs/introducing-jsx.html. [Accessed 05 09 2020].

[32] "React Redux | React Redux," [Online]. Available: https://react-redux.js.org. [Accessed 22 11 2020].

[33] "Next.js by Vercel - The React Framework," [Online]. Available: https://nextjs.org. [Accessed 05 09 2020].

[34] "Next.js 101: What Features You Should Know About," [Online]. Available: https://www.netlify.com/blog/2020/06/18/next.js-101-what-you-should-know/. [Accessed 05 09 2020].

[35] "What is a Static Site Generator? How do I find the best one to use?," [Online]. Available: https://www.netlify.com/blog/2020/04/14/what-is-a-static-site-generator-and-3-ways-to-find-the-best-one/. [Accessed 05 09 2020].

[36] "GitHub - yarnpkg/yarn: 📦🐈 Fast, reliable, and secure dependency management.," [Online]. Available: https://github.com/yarnpkg/yarn. [Accessed 13 09 2020].

[37] "Cascading Style Sheets," [Online]. Available: https://www.w3.org/Style/CSS/Overview.en.html. [Accessed 13 09 2020].

[38] "Sass: Syntactically Awesome Style Sheets," [Online]. Available: https://sass-lang.com. [Accessed 13 09 2020].

[39] "Sass: @mixin and @include," [Online]. Available: https://sass-lang.com/documentation/at-rules/mixin. [Accessed 13 09 2020].

[40] "Responsive Web Design – A List Apart," [Online]. Available: https://alistapart.com/article/responsive-web-design/. [Accessed 13 09 2020].

[41] "Mobile First: What Does It Mean? :: UXmatters," [Online]. Available: https://www.uxmatters.com/mt/archives/2012/03/mobile-first-what-does-it-mean.php. [Accessed 18 10 2020].

[42] "Bootstrap · The most popular HTML, CSS, and JS library in the world.," [Online]. Available: https://getbootstrap.com. [Accessed 18 10 2020].

[43] "Jira | Issue & Project Tracking Software | Atlassian," [Online]. Available: https://www.atlassian.com/software/jira. [Accessed 13 09 2020].

[44] "What is Agile Software Development? | Agile Alliance," [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed 13 09 2020].

[45] "Jira Scrum Boards | Atlassian," [Online]. Available: https://www.atlassian.com/software/jira/scrum-boards. [Accessed 13 09 2020].

[46] "Confluence - Accomplish more together | Atlassian," [Online]. Available: https://www.atlassian.com/software/confluence. [Accessed 13 09 2020].

[47] "Zeplin—Collaboration and handoff for product teams | Zeplin," [Online]. Available: https://zeplin.io. [Accessed 13 09 2020].

[48] "Visual Studio Code - Code Editing. Redefined," [Online]. Available: https://code.visualstudio.com. [Accessed 13 09 2020].

[49] "Postman | The Collaboration Platform for API Development," [Online]. Available: https://www.postman.com. [Accessed 13 09 2020].

[50] "1.3.2 – Separation of Concerns - Design Principles | Coursera," [Online]. Available: https://www.coursera.org/lecture/object-oriented-design/1-3-2-separation-of-concerns-nBqPZ. [Accessed 08 11 2020].

[51] "What Are Soft Deletes, and How Are They Implemented? - Brent Ozar Unlimited®," [Online]. Available: https://www.brentozar.com/archive/2020/02/what-are-soft-deletes-and-how-are-they-implemented/. [Accessed 04 10 2020].

[52] "JSON Web Token Introduction - jwt.io," [Online]. Available: https://jwt.io/introduction/. [Accessed 15 11 2020].

[53] " Table - Ant Design," [Online]. Available: https://ant.design/components/table/. [Accessed 04 10 2020].

[54] "Introducing Hooks – React," [Online]. Available: https://reactjs.org/docs/hooks-intro.html. [Accessed 15 11 2020].

[55] " Form - Ant Design," [Online]. Available: https://ant.design/components/form/. [Accessed 30 11 2020].

[56] "Hooks API Reference – React," [Online]. Available: https://reactjs.org/docs/hooks-reference.html#useeffect. [Accessed 30 11 2020].

[57] "What is REST - REST API Tutorial," [Online]. Available: https://restfulapi.net. [Accessed 25 10 2020].

[58] "React-pdf," [Online]. Available: https://react-pdf.org. [Accessed 08 11 2020].

[59] "Advanced Features: Dynamic Import | Next.js," [Online]. Available: https://nextjs.org/docs/advanced-features/dynamic-import. [Accessed 30 11 2020].

[60] "eligrey/FileSaver.js: An HTML5 saveAs() FileSaver implementation," [Online]. Available: https://github.com/eligrey/FileSaver.js. [Accessed 08 11 2020].

[61] "Flexbox - Learn web development | MDN," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox. [Accessed 08 11 2020].

[62] "SheetJS/sheetjs: SheetJS Community Edition -- Spreadsheet Data Toolkit," [Online]. Available: https://github.com/SheetJS/sheetjs. [Accessed 08 11 2020].

[63] "Overview | Format.JS," [Online]. Available: https://formatjs.io/docs/react-intl/. [Accessed 15 11 2020].

[64] "Jest," [Online]. Available: https://jestjs.io/en. [Accessed 15 11 2020].

[65] "Snapshot Testing · Jest," [Online]. Available: https://jestjs.io/docs/en/snapshot-testing. [Accessed 15 11 2020].

[66] "Develop. Preview. Ship. For the best frontend teams – Vercel," [Online]. Available: https://vercel.com. [Accessed 22 11 2020].

[67] "Git," [Online]. Available: https://git-scm.com. [Accessed 22 11 2020].

[68] "What is a DNS CNAME record?," [Online]. Available: https://www.cloudflare.com/learning/dns/dns-records/dns-cname-record/. [Accessed 22 11 2020].

[69] "Managed MongoDB Hosting | Database-as-a-Service | MongoDB," [Online]. Available: https://www.mongodb.com/cloud/atlas. [Accessed 22 11 2020].

[70] "What is a cloud database? | MongoDB," [Online]. Available: https://www.mongodb.com/cloud-database. [Accessed 22 11 2020].

[71] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[72] "How Next.js can help improve SEO - LogRocket Blog," [Online]. Available: https://blog.logrocket.com/how-next-js-can-help-improve-seo/. [Accessed 05 09 2020].

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

Mina, Ra Koort

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Ehitustööde haldustarkvara prototüübi arendamine“, mille juhendaja on Aleksei Talisainen
   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.

04.12.2020

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

Lisa 2 – Eksporditud PDF-i näidis

Kaurits OÜ
Töölehtede väljavõte

Tellimuse ID: 157
Loodud: 25.08.2020 14:22

Kaarepera

Klient: Rauno Testib OÜ
Uno Pool
+372 5561 2135
rauno.testib@gmail.ee

Tehnika: #19 (Hyundai Robex300LC-9A)
Tööd:
Kaevetöö 42 € / tund
3D kaevetöö 75 € / tund
Roksonitöö 63 € / tund

#19 (Hyundai Robex300LC-9A) / Rauno Kööp
3D kaevetöö 75 € / tund

Kuupäev	Kestvus	Šiffer	Töökirjeldus	Kinnitus
25.08.2020 07:00-10:00	4	33001	Planeerisin 50meetrit nõlva	Kinnitamata
Kokku:	4			

#19 (Hyundai Robex300LC-9A) / Rauno Kööp
Roksonitöö 63 € / tund

Kuupäev	Kestvus	Šiffer	Töökirjeldus	Kinnitus
25.08.2020 10:00-19:00	9	33009	Roksonitöö	Uno Pool 25.08.2020 14:26
Kokku:	9			

Lisa 3 – Eksporditud XLSX-i näidis

	A	B	C	D	E	F	G	H	I	J	K
1	Ettevõte	Rauno Testib OÜ									
2	Klient	Uno Pool									
3	Kliendi tele	+37255612135									
4	Kliendi em	rauno.testib@gmail.ee									
5											
6	Objekt	Kaarepera									
7	Tehnika	#19 (Hyundai Robex300LC-9A)									
8											
9	#19 (Hyundai Robex300LC-9A) / Rauno Kööp										
10	3D kaevetöö 75 € / tund										
11	Kuupäev	Alates	Kuni	Kestvus	Distants	Tonnid	Šiffer	Töökirjeldu	Kinnitus		
12	25.08.2020	07:00	10:00	4			33001	Planeerisir	Kinnitamata		
13											
14	Kokku:			4	0	0					
15											
16											
17	#19 (Hyundai Robex300LC-9A) / Rauno Kööp										
18	Roksonitöö 63 € / tund										
19	Kuupäev	Alates	Kuni	Kestvus	Distants	Tonnid	Šiffer	Töökirjeldu	Kinnitus		
20	25.08.2020	10:00	19:00	9			33009	Roksonitöö	Uno Pool 25.08.2020		
21											
22	Kokku:			9	0	0					
23											

Lisa 4 – worksheetData funktsioon

```
const worksheetData = ({ sheets, currentPriceUnit, intl }) => {
  if (sheets) {
    const totalHours = sheets.reduce((prev, cur) => {
      return prev + Number(cur.durationHours ?? 0)
    }, 0)
    const totalWaitingHours = sheets.reduce((prev, cur) => {
      return prev + Number(cur?.waitingHours ?? 0)
    }, 0)
    const totalWeight = sheets.reduce((prev, cur) => {
      return prev + Number(cur.weight ?? 0)
    }, 0)
    const totalDistance = sheets.reduce((prev, cur) => {
      return prev + Number(cur.distance ?? 0)
    }, 0)

    const res = sheets.map((worksheet) => {
      const {
        dateStart,
        dateEnd,
        dateConfirmed,
        cypher,
        description,
        durationHours,
        waitingHours,
        weight,
        distance,
        objectContact,
      } = worksheet

      const signature = dateConfirmed ? `${objectContact?.name} \n${moment(da
teConfirmed).format(dateFormat)}` : intl.formatMessage({ id: 'saas.pdf.unconf
irmed' })

      if (currentPriceUnit === 'priceKMT') {
        return ([
          moment(dateStart).format(dateFormat),
          moment(dateStart).format(timeFormat),
          moment(dateEnd).format(timeFormat),
          distance,
          weight,
          waitingHours,
          cypher,
          description,
          signature
        ])
      } else {
        return ([
          moment(dateStart).format(dateFormat),
```

```
            moment(dateStart).format(timeFormat),
            moment(dateEnd).format(timeFormat),
            durationHours,
            cypher,
            description,
            signature
          ])
        }
      })
      res.push([``])
      if (currentPriceUnit === 'priceKMT') {
        res.push([intl.formatMessage({ id: 'saas.pdf.total' }), ``, ``, ``, tot
alWeight, totalWaitingHours])
      } else {
        res.push([intl.formatMessage({ id: 'saas.pdf.total' }), ``, ``, totalHo
urs])
      }
      res.push([``], [``])

      return res
    }
    return []
  }
```

Lisa 5 – BuildPDF funktsioon

```
const BuildPDF = ({ fileName, orderIDs = [], worksheetIDs = [], dates = [], l
inkProps }) => {
  const { loading: loadingTypes, data: workTypes } = useFetch({
    url: '/workType/list',
    limit: 1000,
  })
  const { loading: loadingOrders, data: allOrders } = useFetchOrders({
    populate: 'orderedEquipment.equipmentUID;orderedEquipment.typeUID;ordered
Equipment.driverUID-
>firstName,lastName;contactUID;objectContact;orderCreator;belongsTo.companyUI
D',
    limit: 1000,
    filters: { filtered: { _id: orderIDs } }
  })
  const { loading: loadingSheets, data: allWorksheets } = useFetchWorksheets(
{
    populate: 'equipmentUID;contactUID;objectContact;belongsTo.userUID',
    limit: 1000,
    filters: { filtered: { _id: worksheetIDs } }
  })
  const intl = useIntl()

  const build = ({ includeUnconfirmed }) => {
    if (orderIDs.length > 0) {
      allOrders.forEach(order => {
        const worksheetsIncluded = includeUnconfirmed ? allWorksheets : allWo
rksheets.filter(worksheet => worksheet.dateConfirmed)
        order.worksheets = worksheetsIncluded.filter(worksheet => {
          if (dates && dates.length > 0) return (
            (moment(worksheet.dateStart).isBetween(dates[0], dates[1], 'day',
 '[]') ||
              moment(worksheet.dateEnd).isBetween(dates[0], dates[1], 'day',
'[]')) &&
            worksheet.supplierOrderUID === order._id)
          return worksheet.supplierOrderUID === order._id
        })
      })
      exportData({ orders: allOrders, workTypes, intl, fileName })
    } else if (worksheetIDs.length > 0) {
      const worksheetsIncluded = includeUnconfirmed ? allWorksheets : allWork
sheets.filter(worksheet => worksheet.dateConfirmed)
      const orders = allOrders.filter(o => worksheetsIncluded.map(w => w.supp
lierOrderUID).indexOf(o._id) !== -1)
      orders.forEach(order => {
        order.worksheets = worksheetsIncluded.filter(worksheet => worksheet.s
upplierOrderUID === order._id)
      })
      exportData({ orders, workTypes, intl, fileName })
    }
```

```
    linkProps.onClick && linkProps.onClick()
  }

  const loading = loadingTypes && loadingSheets && loadingOrders

  if (!loading && allOrders.length > 0 && allWorksheets.length > 0) {
    //Uses component props instead of fetch results because fetch is always >
 0
    const hasUnconfirmedSheets =
      (orderIDs.length > 0 && allOrders.some(order => order.worksheetsUnconfi
rmed)) ||
      (worksheetIDs.length > 0 && allWorksheets.some(worksheet => !worksheet.
dateConfirmed))

    if (!hasUnconfirmedSheets) {
      build({ includeUnconfirmed: false })
      return (<></>)
    } else if (worksheetIDs.length === 1 && hasUnconfirmedSheets) {
      build({ includeUnconfirmed: true })
      return (<></>)
    } else {
      return (
        <ExportModal
          visible={true}
          onCancel={() => {
            linkProps.onClick && linkProps.onClick()
          }}
          onNo={() => {
            build({ includeUnconfirmed: false })
          }}
          onYes={() => {
            build({ includeUnconfirmed: true })
          }}
        />)
    }
  }
  return (<></>)
}

const PDFDocument = ({ orders, workTypes, intl }) => {
  return (
    <Document>
      {orders && orders.map((order, pageIndex) => {
        return (
          <PDFPage order={order} workTypes={workTypes} intl={intl} key={pageI
ndex} />
        )
      })}
    </Document>
  )
}
```