TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Janar Männistu 179233IADB

DOKUMENDIMALLIDE MUUTMISE ISETEENINDUSKESKKOND

Bakalaureusetöö

Juhendaja: Märt Kalmo
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: Janar Männistu

18.05.2020

Annotatsioon

Töö eesmärgiks on uurida võimalike lahendusi dokumendimallide muutmise iseteeninduskeskkonna loomiseks ning implementeerida lahendus, kus kasutajad saavad ilma arendaja sekkumiseta dokumente kujundada.

Autor uuris olemasolevaid lahendusi ning analüüsis neid. Analüüsi käigus selgus, et piisavalt head lahendust ei ole olemas, mis lahendaks püstitatud probleemi. Olulisemad probleemid, mida töös käsitletakse: dokumendimalli kirjatüübi valik ja kasutaja laetud kirjatüübi kasutamine süsteemis. Dünaamiliste väljade kasutamine ning nende liigutamine. Andmevahetus serveriga ning andmete struktuur ja dokumendi genereerimine.

Töö tulemusena valmis prototüüp lahendus, mille abil saab implementeerida terviliku süsteemi. Prototüüp valmis kasutades ReactJS, Kotlinit, MySQL ning NodeJSi. Samu tehnoloogiad saab kasutada ka terviliku süsteemi arendamisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 4 peatükki, 14 joonist, 1 tabelit.

Abstract

Self-service Environment for Editing Document Templates

The purpose of this thesis is to research and analyze solutions for self-service environment for editing document templates and to implement the solution where users can edit document templates without the help of a developer.

The Author researched solutions that already exsist to find out if there are any solutions that would solve the problem of developers creating custom document templates for various system users. Author did not find solutions that would eliminate the problem easily therefore desicion was made to research and develop the self-service environment.

The Author analyzed how to select font for document template and how to implement custom fonts for the system. How to implement dynamic fields and how to move fields around in the template. Data transfer and structure between front and backend and how to generate documents.

As a result of this thesis, a prototype of self-service environment for editing document templates was created. The prototype shows the base functionality of the system and how to implement it. The prototype was created using ReactJS, Kotlin and NodeJS which will be used to develop the actual system itself.

The thesis is in Estonian and contains 26 pages of text, 4 chapters, 14 figures, 1 table.

Lühendite ja mõistete sõnastik

API	Rakendusliides ehk reeglistik juba eksisteeriva programmiga suhtelmiseks. (ik *Application programming interface*)
BLOB	Binaarsete andmete salvestamiseks mõeldud andmetüüp andmebaasis (ik *Binary large object*)
CSS	Küljendamisel kasutatav märgendikeel (ik *Cascading style-sheets*)
Docker	Virtualiseerimiseks mõeldud programm, mis võimaldab koodi jooksutada olenemata originaalsest operatsioonisüsteemist
DOCX	Microsoft Office Word dokumendi formaat
Flyway	Andmebaasi migratsioone haldav teek
HTML	Tekstipõhine kodeerimissüsteem veebidokumentide loomiseks (ik *Hypertext markup language*)
JSON	Andmevahetus formaat, kus on kirjeldatud võti-väärtus paarid (ik *Javascript object notation*)
JSX	ReactJS kasutatav süntaks, mis on javascripti laiendus
Multi-tenant	Arhitektuur, kus üks instants serveerib mitut kasutajat
ReactJS	Javascripti teek mõeldud kasutajaliideste ehitamiseks
SQL	Andmebaasiga suhtluseks kasutatav keel (ik *Structured Query Language*)
XSS	Murdskriptimine, rünnakutüüp (ik *Cross-site scripting*)

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Eestis on registreeritud üle 1000 idufirma. Neist 180 tegutseb äritarkvara valdkonnas, moodustades sellest suurima idufirmade valdkonna Eestis. Ettevõtted üle maailma otsivad mugavamaid tööriistu, mis kergendavad tööd, hoiavad aega ja raha ning muudaksid töö tegemise efektiivsemaks [1].

TRUKKS AB on tarkvara kui teenusena pakkuv ettevõte, mis pakub täislahendust ettevõtele, kes pakuvad paigaldust ja hooldust koos nendega seotud teenuslepingutega. Töö kirjutamise hetkel on peamisteks klientideks liftide ja turvasüsteemidega tegelevad ettevõtted.

Autor püstitab töös probleemi, selgitab töö eesmärki, kirjeldab lähteolukorda ning paneb paika metoodika mille abil probleemi lahendada. Järgmiseks selgitab autor süsteemi nõuded, mille alusel analüüsitakse süsteemi funktsionaalsust ja väljanägemist, rahuldamaks klientide ning ettevõtte vajadused.

Autor töötab välja lahenduse jälgides klientide ja TRUKKS AB nõudeid ning analüüsides erinevaid võimalusi. Peale lahenduse väljatöötamist kirjeldab autor tehnilist lahendust, mille põhjal saab süsteemi arendada. Töö tulemusena valmib süsteem, mis lahendab püstitatud probleemi.

1.1 Probleem

Tarkvara kui teenusena mudelil töötavad ettevõted on mugavad kasutajatele, kuna kasutajad saavad kasutada juba valmis olevat lahendust mis lahendab kasutajate probleemid. Olgu selleks töö efektiivsus, aja ja raha kokkuhoid või parem ülevaade ettevõtte käekäigust. Väiksematele ettevõttele on personaalse lahenduse väljatöötamine ning arendamine väga kulukas, mistõttu on selliste teenuste olemasolu väga tähtis.

Tihitpeale on sellised teenused igale kasutajale samasugused, mis tähendab, et kõik kasutajad, kes kasutavad süsteemi toodavad ka samasuguseid dokumente, mille disain või meili sisu on igal kasutajal samasugune. Süsteemi kasutavate ettevõtete sooviks on

teistest erineda, mistõttu soovitakse luua personaliseerituid dokumente või klientidele saadetavaid meile.

Süsteemi kasutaval ettevõttel on soov ümber disainida dokumente oma brändingule vastavalt, mistõttu peab arendaja tekitama süsteemi eraldi dokumendimalli. Seega kasvavate kasutajate arvuga kasvab ka soov personaliseeritud dokumentidele, mis tekitab olukorra, kus arendajad ei saa tegeleda süsteemi edasi arendamisega vaid peavad implementeerima personaliseerituid lahendusi klientidele.

1.2 Eesmärk

Autori eesmärk on vähendada arendajate koormust dokumendite disainimises ning välja töötada dokumendimallide muutmise iseteeninduskeskkond. Töö raames selgitab autor parima viisi kirjeldatud probleemi lahendamiseks. Lahenduse leidmiseks selgitab autor välja süsteemi nõuded kliendilt ja ettevõttelt, mille põhjal selgitatakse välja milline peaks kasutajaliides olema ja analüüsitakse, kuidas antud lahendust saab implementeerida. Tuues välja võimalikud murekohad ja leides neile lahenduse. Lõpplahendusena on kasutajatel võimalik infosüsteemis iseseisvalt, vastavalt ettevõtte brändingule ehitada soovitud dokumendimalle, millest valmib klientidele saadetav dokument.

1.3 Lähtetingimused

TRUKKS AB süsteemis saab kasutaja välja saata arveid, mis on koostatud tehtud töö põhjal. Arvel on kuvatud arve koostava ettevõtte logo ja andmed, saaja andmed ning arve read. Töö kirjutamise hetkel on kõik arved ühesugused, süsteemi andmebaasis on standardne arve mall mille põhjal koostatakse arve. Ainus erinevus arvetel on keel ning andmed arve peal. Lisaks on võimalus saata kliendile meili, kuhu on lisatud dokument. Meili sisu on sarnaselt arvetele samasugune ning puudub võimalus muuta kirja sisu. Arveid ja meile saab saata kolmes eri keeles: eesti, inglise, rootsi. Joonisel 1 on välja toodud saadetava arve näidis. PDF dokumendi genereerimiseks on varaseamalt autori poolt loodud teenus, mis töötab üle API ja mis vastab POST päringule PDF dokumendiga. Päringuga peab kaasa saatma soovitud dokumendi HTMLi ning CSSi.


Trukks

Ettevõte B
EE1034434
Raja 15, Tallinn
isik@ettevoteb.com

TELLIJA

ARVE NR
AR1412

420420
Ettevõte A
isik@ettevotea.com
+372 5005005
Käibemaksukohuslase number EE12345
Arve seoses K009,Tallinn,Tänav

2020-04-25

Kirjeldus
Tehtud töö kirjeldus

Artikkel	Hind	Kogus	Kokku
0000 - Arve rida	100.0 € / tk	1.0	100.0 €
	Vahesumma		100.00 €
	Käibemaks	0 %	0.00 €
	Kokku		100.00 €

Käibemaks pöördmaksustatud	BG number	Maksetähtpäev	Käibemaksukohustuslase number
	561-67823	2020-05-25	SE012345678

Joonis 1. Ekraanitõmmis saadetavast arvest

1.4 Metoodika

Töö valmib arendusuuringu metoodikaga, analüüsides süsteemi nõudeid ning leides võimalikke lahendusi nõuete implementeerimiseks. Analüüsi käigus tekkinud probleemidele leitakse võimalikud lahendused ning tuginedes nõuetele leitakse parim meetod süsteemi realiseerimiseks.

Lahenduse realiseerimise faasis selgitab autor tehnilisi detaile süsteemi implementeerimiseks. Lahenduse realiseerimise faasis selgitab autor tehnilisi detaile

süsteemi täide viimiseks, kirjeldades süsteemi osade töötamist ja tervikliku lahenduse töötamist.

Tulemuste osas selgitab autor valminud süsteemi ning kas see lahendab ettevõte probleemi. Lisaks pakub välja võimalike edasiarendusi mis jäävad töö skoobist välja.

2 Lahenduse väljatöötamine

Käesolevas peatükis selgitab autor välja klientide ja ettevõtte nõuded ning formuleerib need funktsionaalseteks ning mitte-funktsionaalseteks nõueteks. Analüüsitakse olemasolevaid lahendusi ja nende sobivust ning seejärel kirjeldadakse süsteemi olemust ning käitumist, tuginedes nõuetele. Nõuetest lähtuvalt tuuakse välja probleemsemad kohad ning pakutakse võimalikke lahendusi.

2.1 Klientide nõuded

Iduettevõtete käekäiku mõjutavad suuresti nende kliendid. Selleks, et saada võimalikult palju kliente ja pakkuda neile parimat teenust on vaja kuulata ja uurida mida kliendid vajavad. Üks peamisi põhjuseid miks idufirmad läbi kukuvad on klientide mitte kuulamine ning ebamugav kasutajaliides [2]. Selleks, et süsteem valmiks klientide soove järgides uuris töö autor klientidelt mida nad täpsemalt vajavad esitati küsimused: „Kui täpselt peab saama paigutada elemente? Näiteks firma logo ja erinevad tekstid“, „Kas kirjatüüp on tähtis?“. Ettevõtetel on kujunenud oma nägu ja ning ühtne disain, mistõttu on klientide jaoks tähtis kirjatüübi valimine. Ühe kliendi sooviks on paigutada asju pigem lihtsalt ning liigutada antud mallis mõningaid välju ringi. Teiste klientide sooviks on aga paigutada täpsemalt dokumendi välju ning teksti, selleks et süsteem suudaks kliendi brändingule võimalikult sarnast dokumenti produtseerida.

2.2 Ettevõtte nõuded

Ettevõtte nõuded selgitas autor välja TRUKKS AB esindajatega rääkides leides vastused küsimustele kui palju ajalist ja rahalist ressursi peaks panustama antud süsteemi väljatöötamiseks ning kui prioriteetne on antud arendus võrreldes teiste arendamist ootavate funktsioonaalsustega.

Ettevõtte jaoks on tähtis võimalikult varakult klientidele süsteemi näidata ja saada vajalikku tagasisidet. Tagasiside sisaldab endast kliendi rahulolu lahendusele ning tagasiside põhjal lisatakse vajadusel hilisemates faasides funktsionaalsust. Kasutaja

peab saama koostada dokumendile malli, mida oleks võimalik hiljem kasutada. Dokumendimall peab olema piisavalt kujundatav, et klient saaks meelepärase dokumendimalli. Terve TRUKKS AB süsteemi kasutajaliides on ehitatud kasutades ReactJS, mistõttu on tehnoloogiate valikul piirangud. Antud kasutajaliides tuleb ehitada ReactJSi kasutades. Serveri poolseteks tehnoloogilisteks piiranguteks on Kotlin, MySQL. PDF teenuse edasiarendus mis on ehitatud NodeJSi kasutades ning mida peab saama jooksutada eraldiseisvalt Dockeri konteineris.

Klientide ning ettevõtte nõuded formuleeris autor funktsionaalseteks ja mitte-funktsionaalseteks nõueteks.

Funktsionaalsete nõuete loetelu:

- Kasutaja peab saama dokumendi väljasid ning teksti positsioneerida
- Kasutaja peab saama valida dokumendi kirjatüüpi ning suurust
- Kasutaja peab saama dokumendile logo või pilte lisada
- Kasutaja peab saama lisada tinglike välju
- Kasutaja peab saama valida kliente kellele antud dokumendimall mõeldud on
- Süsteem peab pakkuma eesti, inglise ja rootsi keele tuge

Mitte-funktsionaalsete nõuete loetelu:

- Lahendus peab olema loodud kasutades Kotlinit, MySQL, NodeJSi ja ReactJSi
- Kasutajaliides peab vastama ettevõtte TRUKKS AB disainile
- Kasutajaliides peab olema piisavalt intuitiivne, et kasutaja saaks seda kasutada ilma tehnilise toeta ning vajadusel toetudes õpetusele

Loodava süsteemi dokumendimallid leiavad kasutust:

- Kasutajaliideses kuvamiseks
- PDF dokumendi genereerimiseks
- Meeldetuletus meilide saatmiseks
- Meili saatmine peale toimingut, näiteks töötaja määramine tööobjektile

2.3 Olemasolevad lahendused

Olemasolevate lahenduste leidmiseks kasutas autor Google otsingumootorit järgnevate märksõnadega: „document templating software“, „react document templating library“, „document template management software“. Otsingu tulemusel ei leidnud autor töö kirjutamise hetkel mitte ühtegi teeki mis rahuldaks nõudeid.

Autor leidis tarkvara kui teenusena pakkuvaid ettevõtteid, kes pakuvad võimalust koostada dokumendimalle ning täita neid soovitud andmetega. Lähemalt uuriti kolme lahendust:

- Legito
- Templafy
- Windward studios

Kõik teenused pakuvad sarnast lähenemist, kus klient saab luua dokumendimalli, need salvestada ning saata dokumente, mis põhinevad dokumendimallil, täites need vajalike andmetega. Esmapilgul antud teenused sobiksid lahendamaks püstitatud probleemi, kui pakutavaid lahendusi saaks integreerida süsteemi Antud lahendused on eraldiseisvad produktid, kus kasutajad saavad hallata dokumendimalle, mistõttu ei ole võimalik neid süsteemi integreerida. Töös püstitatud probleemi põhjal peab kasutaja saama süsteemi siseselt koostada dokumendimalle, mis täidetakse vajaliku informatsiooniga kasutaja eest. Seetõttu ei lahenda antud teenused töös püstitatud probleemi.

Legito on 2014. aastal loodud lahendus, mis on mõeldud legaalsete dokumendimallide koostamiseks ja hoiustamiseks. Lisaks saab dokumente täita vajalike andmetega. Pakutav teenus töötab järgenvalt: kasutaja logib nende süsteemi, loob dokumendimalli ja salvestab selle ning siis saab kasutaja näiteks üle API dokumenti genereerida saates API päringuga vajalikud andmed. Töö probleemi see ei lahenda kuna TRUKKS AB süsteemi kasutaja peab saama süsteemi siseselt dokumendimalli koostada ja salvestada. Lisaks on Legito teenuse kasutamine tunduvalt kulukam. Legito süsteem maksab 90€ kasutaja kohta aastas, kui tasumine on aasta kaupa. Mis tähendab seda, et iga kasutaja, kes sooviks dokumendimalle koostada on äärmiselt suur lisakulu [3] [4].

Templafy on samuti 2014. aastal loodud lahendus, mis on sarnane Legito pakutavale lahendusele, kuid Templafy lahendus pakub paremat API võimalusi, kus üle API on

võimalik ka uusi dokumendimalle ülesse laadida. TRUKKS AB süsteemiga ei sobitu Templafy lahendus kuna üleslaetav dokumendimall peab olema Microsoft office wordis valmistatud ja faililaiend peab olema DOCX. Templafy kodulehel puudub informatsioon hinna kohta kuna pakutakse personaliseeritud hinda [5] [6].

Windward studios lahendus kasutab dokumentidemallide koostamiseks Microsoft Office programme (Word, Excel, Powerpoint) selline lähenemine samuti ei sobi, kuna seda ei saa süsteemi integreerida. Windward studios hinnad on äärmiselt kallid alates 480€ kuus [7].

Võrreldes erinevaid lahendusi jõuti otsusele, et tuleb arendada süsteem mis on spetsiaalselt mõeldud TRUKKS AB jaoks ning mis rahuldaks kõik klientide ja ettevõtte nõuded.

2.4 Kasutajaliides

Süsteemi kasutajaliidese väljatöötamiseks lähtub autor seatud nõuetest ning nende põhjal analüüsib milline peaks kasutajaliides olema. Vajalike nõuete täitmiseks ja selleks, et kasutajal oleks arusaam milline dokument välja võiks näha on vajalik dokumendi eelvaade. Kui kasutaja vormindab dokumendimalli näeb kasutaja koheselt kuidas dokument reaalselt välja näeks.

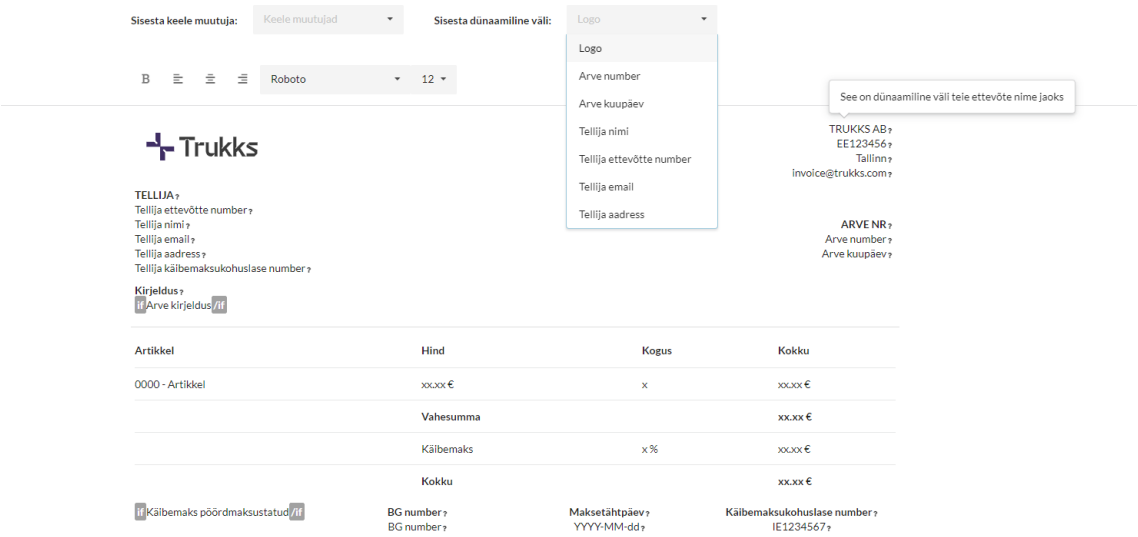
Kasutaja saab lisada staatilist teksti ja dünaamilisi välju. Dünaamilist välja saab eristada tekstis sellele järgneva pisikese „?“ abil mille peale hiirega minnes kuvatakse abistav tekst. Dünaamiliste väljade jaoks on vajalik kasutajat aidata, kuna need peavad olema õigesti märgitud mallile, selleks tasub teha loetelu olemasolevatest väljadest mida kasutaja kasutada saab. Dünaamilised väljad on mingi kindla dokumendiga seotud muutuvad väljad (arve number, kliendi nimi) või süsteemis kasutusel olevad keelemuutujad (Arve saaja – *Recipient*).

Kasutaja saab valida kirjatüüpi, suurust ja tugevust. Kui kasutaja koostab dokumendimalli peab olema mugav võimalus muuta kirjatüüpi, suurust või tugevust autor leiab, et selline asi on hästi lahendatud tekstiredaktorites. Ning arendatavas süsteemis peaks lähenema sarnaselt, kuna kasutaja on juba tuttav antud lähenemisega.

Kasutaja saab määrta tinglike välju selleks tuleks sisestada dünaamiline väli ning selle kõrval märkida, et tegemist on tingliku väljaga, mis tähendab, et antud tekst lisatakse dokumendile ainult kui antud tingimus on täidetud.

Dokumendile ettevõtte logo või mõne muu pildi lisamine. Pildi lisamiseks on vaja pilt ülesse laadida või valida juba olemasolev pilt. TRUKKS AB loodud süsteemis on võimalik pilte üleslaadida, kuid puudub võimalus juba üleslaetud piltide valimiseks. Ettevõtte logo on aga kättesaadav eraldi ning selle sisestamiseks saab kasutada vastavat märgendit.

Dokumendimalli valmimisel saab kasutaja selle salvestada ja antud malli kasutatakse dokumentide väljastamiseks. Hiljem on võimalik kasutajal muuta tehtud dokumendimalli.



Joonis 2. Ekraanitõmmis kasutajaliidese kujundusest

Joonisel 2 on kujutatud kasutajaliidese näidis, milline võiks lõpplahendus kasutajale välja näha. Joonisel 2 on kujutatud arve mall mille põhjal genereeritakse joonisel 1 kujutatud arve. Joonisel 2 on näha, et dokumendimalli koostamisel näeb kasutaja väga sarnast vaadet nagu on genereeritav dokument. Kasutaja ettevõtte andmed täidetakse mallil koheselt ära, kui kasutaja valib rippmenüüst sobiva valiku, näiteks logo või firma nimi. Valides kliendi väljasid kuvatakse tavatekstina väli mallile. Selline kasutajaliides ei tekita kasutajale segadust võrreldes kui kasutada märgendikeelt, kus dünaamilised väljad on mingisuguse märgendi sees.

2.4.1 Kirjatüübi valimine

Joonisel 2 on kujutatud, kuidas kasutaja saab valida dokumendile kirjatüüpi. Veebist võib leida üle 600 000 erineva kirjatüübi [8]. Selleks, et kirjatüüp igalpool toimiks nii nagu soovitud on vaja, et antud kirjatüüp oleks veebilehel olemas mis garanteerib selle, et antud kirjatüüp laaditakse alla, kui kasutaja külastab lehte [9]. Teiseks peab see kirjatüüp leiduma serveris kus dokumendi mallist dokumenti genereeritakse, kuna PDF dokumendi ja emaili jaoks on vajalik kirjatüüp kaasa anda. Kui seda ei tehta võib juhtuda, et arvutis või telefonis kus soovitakse dokumenti vaadata puudub antud kirjatüüp ja dokumendil on vale kirjatüüp või dokumenti ei saa lugeda.

2.4.1.1 Potentsiaalsed lahendused

Kirjatüübi valikuks pakub autor välja kolm potentsiaalset lahendust:

- Eeldefineeritud kirjatüüpide loetelu
- Enamlevinumate kirjatüüpide pakkumine
- Kasutaja saab ise laadida oma soovitud kirjatüübi keskkonda

Eeldefineeritud kirjatüüpide loetelu võib jääda liiga väikseks ja ei paku õiget kirjatüüpi mistõttu kliendi soov jääb rahuldamata. Enamlevinumate kirjatüüpide pakkumise probleemiks võib kujuneda kirjatüüpide üleküllus, Google Fonts pakub pea tuhandet kirjatüübi perekonda [10]. Kirjatüübi perekonda kuuluvad samasuguse disainiga kuid erineva stiiliga kirjatüübid (tugevus, nurk). Kolmandaks kui kasutaja saab ise keskkonda oma soovitud kirjatüübi ülesse laadida võib tekkida probleem, et igal ettevõttel ei ole veel kujunenud oma stiili ja kindlat disaini, kuid nad soovivad siiski muuta millised näevad nende dokumendid välja, sest algne kirjatüüp ei sobi neile. Selleks sooviks ettevõtte siiski valida eeldefineeritud valikute seast endale meeldivad kirjatüübi.

Autori arvates on kõige optimaalsem pakkuda tavalisi enamlevinuid kirjatüüpe nagu Times New Roman, Georgia, Palatino Linotype, kõige populaarsemaid Google fontsi kirjatüüpe ning lisaks pakkuda võimalust kasutajal ka oma kirjatüüp ülesse laadida.

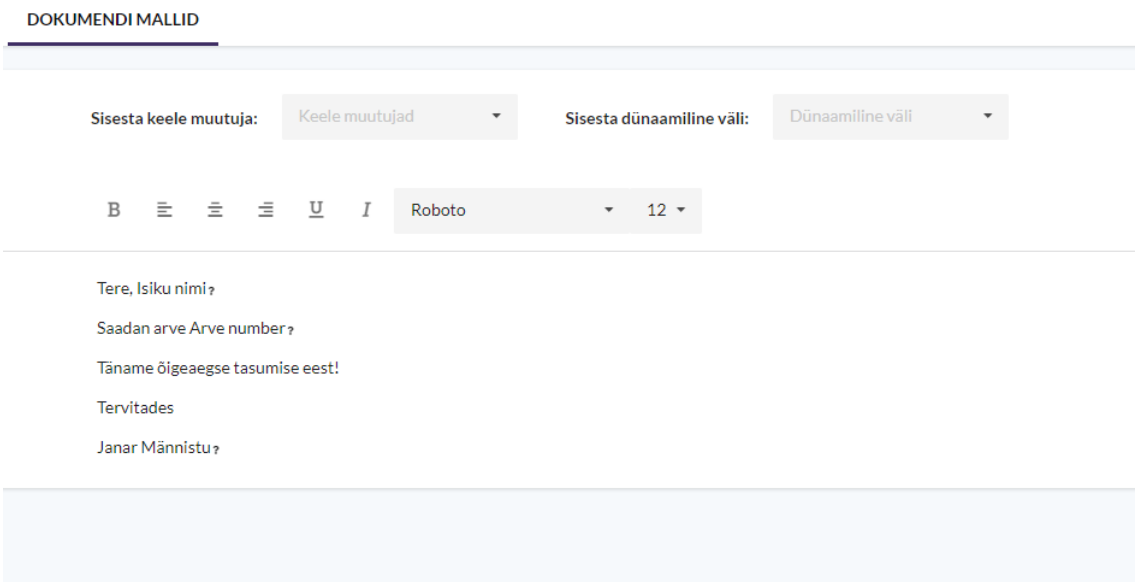
2.4.2 Väljade paigutamine

Väljade paigutamine on üks peamisi funktsioone loodava süsteemi juures, kuna see annab suure vabaduse kasutajale kujundada dokumente just nii nagu soovitud. Paigutada vajalikud väljad sellistesse kohtadesse nagu soovitakse. Selleks, et kasutaja

saaks liigutada dokumendi välju vastavalt oma soovile on vaja pakkuda teksti redigeerimise võimalust.

2.5 Dokumendimallide tüübid

Süsteemis on võimalik luua erinevaid dokumendimalle, kasutajad saavad luua PDF failide malle (arved, lepingud) kui ka emaili malle. Töö kirjutamise hetkel on süsteemis võimalik kasutajal saata kliendile email mis sisaldab teksti ning manusena on juurde lisatud arve.



Joonis 3. Ekraanitõmmis arve emaili malli näidis

Joonisel 3 on kujutatud arve emaili malli, ekraanitõmmiselt on näha, kuidas mall koosneb staatilisest tekstist ning dünaamilistest väljades. Dünaamiliste väljadena on isiku nimi, arve number ning saatja nimi, mis on juba süsteemi poolt täidetud õige nimega, et malli tegija saaks kohese tagasiside milline dokument võiks välja näha.

2.6 Tehniline lahendus

Tehnilise lahenduse väljatöötamiseks tugineb autor eelnevalt kirjeldatud nõuetele ning kasutajaliidese teostusele. Antud peatüki eesmärk on välja selgitada kuidas antud süsteemi ehitada kirjeldades keerukamad protsessid lahti ja leides neile lahenduse.

2.6.1 Kirjatüübi üleslaadimine

Kasutajad, kes soovivad oma valitud kirjatüüpi ülesse laadida olgu selleks nende ettevõttele tehtud spetsiaalne kirjatüüp või veebist leitud, ostetud kirjatüüp mis sobib nende brändiga. Selleks, et kaitsta nende kirjatüüpi peavad kasutajate poolt ülesse laetud kirjatüübid olema nähtaval ainult neile kasutajatele, kes kuuluvad antud ettevõttese.

Kasutaja üleslaaditud kirjatüüpi peab hoiustama selleks, et see oleks kättesaadav dokumendi genereerimiseks.

Üks võimalus oleks neid hoida serveris, kuhu kogutakse kõikide kasutajate kirjatüübid. Antud lahendusega tekib probleem, kui süsteem jookseb erinevate serverite peal sel juhul peaks iga serveri peale dubleerima kirjatüüpe mis lisab märgatavat keerukust.

Teine võimalus oleks kirjatüübid laadida mõnda failihaldus teenusesse, TRUKKS AB kasutab Amazoni teenuseid ning pilte hoitakse Amazon S3 (ik Amazon Simple Storage Service) teenuses. Antud lahendus elimineeriks probleemi, kus faile peab dubleerima erinevate serverite vahel, kuna kõik failid on ühes kohas ja kättesaadavad.

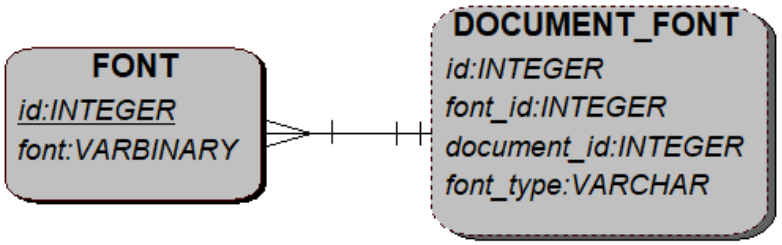
Kolmas variant on üleslaetud kirjatüüp salvestada andmebaasi kasutades *BLOB* andmetüüpi. Süsteemis on kasutusel *MySQL* andmebaas, mis võimaldab nelja erinevat *BLOB* andmetüüpi määrata: *TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB. BLOB* andmetüüp on mõeldud binaarsete andmete hoiustamiseks andmebaasis erinevate *BLOB* tüüpide vahe on nende suurus *TINYBLOB* hoiab kuni 255 baiti informatsiooni, *BLOB* 65,535 baiti ehk 64 kilobaiti informatsiooni, *MEDIUMBLOB* 16,777,215 baiti ehk 16 megabaiti informatsiooni ning *LONGBLOB* suudab hoida kuni 4 gigabaiti informatsiooni. Selleks, et aru saada millist andmetüüpi kasutada võrdles autor Google Fontsi poolt pakutavat kümmet kõige populaarsemat kirjatüübi perekonda ning nende ükskikute kirjatüüpide suurusid. Tabelis 1 on välja toodud kirjatüübi perekonna nimi, kirjatüüpide arv perekonnas ning keskmine kirjatüübi suurus kilobaitides. Valitud kirjatüüpide hulka sattus *Noto Sans JP* mis oli teistest võrreldud kirjatüüpidest tunduvalt suurem ning viis üldise keskmise suuruse 580,50 kilobaidini, kui *Noto Sans JP* jätta võrdlusest välja on keskmine kirjatüübi suurus 145,00 kilobaiti. Antud võrlduse tulemusena võib järeldada, et sobiksid ainult *MEDIUMBLOB* ja *LONGBLOB* andmetüübid. *MEDIUMBLOB* andmetüübi kasutamine on kõige mõistlikum nii jääb

piisavalt ruumi ka mahukamate kirjatüüpide üleslaadimiseks ning piiratakse liigselt massiivsete kirjatüüpide üleslaadimist [11].

Tabel 1. Kirjatüüpide suurused

Perekonnanimi	**Kirjatüüpide arv**	**Keskmine suurus(kb)**
Roboto	12	173,30
Open Sans	10	96,80
Lato	10	68,80
Montserrat	18	247,22
Noto Sans JP	6	4500
Roboto Condensed	6	173,83
Source Sans Pro	12	181,17
Nunito Sans	14	92,36
Raleway	18	157,33
Roboto Mono	10	114,10
		580,50

Andmebaasi salvestades ei teki lisa keerukust, kus peab laadima kirjatüübi eraldi teenusesse või serverisse. Kirjatüüpide hoidmiseks andmebaasis on kaks võimalust hoida kogu informatsiooni ühes tabelis või hoida kirjatüübi binaarset infot koos identifikaatoriga eraldi tabelis ning muu info teises tabelis. Selleks, et vähendada mälu nõudlust päringute jaoks mis ei kasuta kirjatüübi binaarset infot on mõistlik hoida neid eraldi tabelites nagu on kujutatud joonisel 4 [12].



Joonis 4. Kirjatüüpide hoiustamise mudel

2.6.2 Dokumendimalli redigeerimine

Dokumendimalli redigeerimiseks on võimalik kasutada tekstiredaktorit mille saab koodi lisada kui teegina ning kasutada. Autor otsis tekstiredaktoreid kasutades otsingumootorit, otsingusõneks oli: „Reactjs text editor“. Autor leidis kolm tekstiredaktorit mida uuris lähemalt:

- TinyMCE
- CKEditor
- DraftJs

Kõik kolm tekstiredaktorit olid sobilikud ReactJs okösüsteemiga ning autor proovis kõiki kolme kasutada. CKEditor ei käitnud vastavalt soovile, keeruline oli tekitada tabeleid ja muuta veerge ning ridu. DraftJs suurim probleem on dokumentatsiooni vähesus ning autori jaoks on dokumentatsioon äärmiselt vajalik kui kasutada kolmanda osapoole teeki. TinyMCE on suur projekt mis pakub tasulist ja tasuta võimalust tekstiredaktori kasutamiseks. Tasuline variant ei sobi, kuna see on lisakulu ettevõttele mis ei õigusta end finantsiliselt ära. Antud süsteemis on vaja suhteliselt lihtsat tekstiredaktorit, kuid TinyMCE redaktoris on funktsionaalsust äärmiselt palju ning antud süsteemi jaoks ebavajalikku funktsionaalsust leidub palju.

Tekstiredaktorite võrdluses selgus, et kõik tekstiredaktorid väljastavad erinevas formaadis HTMLi ning keeruline on määrata oma soovitud väljundit.

Kuna kasutajale kuvatav dokumendimall koosneb staatilisest tekstis ning dünaamilistest väljadest mis on kasutaja mugavuse jaoks täidetud kas kasutaja andmetega või tekstidega nagu „Kliendi nimi“ lisaks saab dokumendimallile lisada tinglike välju mida kuvatakse mallil kahe „if“ sõna vahel tekib olukord, kus dokumendimall mida kasutajale kuvatakse ning dokumendimall millest dokumenti genereeritakse on erinevad.

```
<div th:if=“customer.reverseVat“>
	<div>Käibemaksukohuslase number</div>
	<div>[[${customer.configuration.vatNumber}]]</div>
</div>
```

Joonis 5. Dokumendimalli genereerimiseks vajaminev kood

Üleval väljatoodud koodiblokis on näide, milline näeks välja dokumendimalli HTML kus on tingimus, et kliendil peab olema aktiveeritud käibemaksu pöördmaksustamine mille puhul kuvatakse kliendi käibemaksu kohuslase number. Antud dokumendimall on teostatud Thymeleaf mallimootoriga [13]. Kasutajale kuvatakse dokumendimall kliendirakenduses mis on ehitatud ReactJS teeki kasutades. ReactJS toob kaasa mõningase süntaksi erinevuse. ReactJSi ametlik dokumentatsioon soovitab kasutada kasutajaliidese loomiseks JSX süntaksit mis ei ole HTML vaid on Javascripti juurdeehitus JSX süntaks võib meenutada mõnda mallimootorit, kuid tuleb täielike javascripti võimalustega. JSX soovitatakse kasutada sellepärast koos Reactiga kuna selle kirjutamine produtseerib Reacti elemente mida Reacti mootor renderdab [14].

```
<div>
	<Popup content={helpText}
		trigger={<IfWrapper condition={fieldValue} />}
	/>
</div>
```

Joonis 6. ReactJSi kood, mis genereerib kasutajaliideses dokumendimalli bloki

Üleval kirjeldatud ReactJSi kood kuvab kasutajale „if“ sõne vahele tingimuse mida hoitakse muutujas *fieldValue*. Popup element kuvab abistava teksti mida on näha kui kursoriga elemendi peal olla joonisel 2 on näha milline näeb välja abistav tekst kui kasutaja on kursoriga elemendi peal mis on dünaamiline väli või keele muutuja.

Analüüsides erinevaid tekstiredaktoreid ning tulenevalt kasutajale kuvatavast ja serveris kasutatavst dokumendimalli HTMLi erinevust jõuti järeldusele, et kasutajapoolne dokumendimalli redigeerimine tuleb ehitada ise, kuna puuduvad lahendused mis suudaksid rahuldada vajalike nõudeid.

2.6.3 HTMLi turvariskid

Lastes kasutajatel lisada omapoolset HTMLi tekib koheselt risk, kus kasutajad saavad potentsiaalselt korda saata rünnakuid peamiselt *XSS* rünnakuid. Stsenaarium kuidas antud rünnakut saaks keskkonnas teha:

- Kasutaja koostab dokumendimalli, kus on *XSS* rünnaku elemendid
- Dokumendimall salvestatakse
- Saadetakse dokument mis on koostatud dokumendimalli põhjal
- Kasutaja avab dokumendi ja on *XSS* rünnaku ohever

Kui kliendile saadetakse PDF dokument siis ei teki XSS rünnaku probleeme, kuid rakenduses on võimalik saata klientidele emaile ning kui emaili mallis esineb rünnakut on klient rünnaku ohver.

Selleks, et ära hoida taolisi rünnakuid peab antud HTMLi valideerima ja lubama ainult selliseid elemente millele ei saa lisada Javascripti koodi. Kuid seal tekkivad probleemid, kus ei osata eemaldada õigeid elemente, kuna mallis võib leiduda <img>, <style> elemente, kui soovitakse pilti või on ettenähtud kus mingi väli peaks asuma. *XSS* rünnaku jaoks ohtlikud elemendid on:

- <script>
- <body>
- <img>
- <link>
- <table>
- <object>
- <style>

Kõikidele loetletud elementidele on võimalik lisada javascripti koodi ühel või teisel moel või javascripti sündmusi. Lihtsam on leida kirjutatud javascripti elemente ja need eemaldada, kuid keerulisemaks läheb kui rünnak on peidetud näiteks:

```
<img src=xyz
onerror=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&
#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

Joonis 7. XSS kodeeritud rünnak img elemendi abil

Antud koodijupp lahti kodeerides:

```
<img src=xyz onerror=”javascript:alert('XSS')”>
```

Joonis 8. XSS lahti kodeeritud kuju

Kui klient avab näiteks emaili kus on antud koodijupp siis osutub ta XSS rünnaku ohvriks, kuna *onerror* kutsutakse välja kui tekib probleeme pildi laadimisega. Antud juhul *src* atribuut ei ole valiidne ja ei leita pildifaili selliselt aadressilt [15].

2.6.4 Andmevahetus

Eelnevas peatükis selgunud HTMLi probleemide vältimiseks kasutab autor andmevahetuseks serveriga JSONit. Saadetav JSON sisaldab dokumendimalli struktuuri ja kujundust. Kliendipool saadab serverile JSONi, ning see salvestatakse andmebaasi. Kuna serveris genereeritav dokumendimall on erinev kasutajale kuvatavast dokumendimallist on vajalik serveri ja kliendipoolele arendada erinev natukene erinev lahendus, mis suudaks JSONist korrektse HTML dokumendi teha.

Tuleb välja töötada JSONi struktuur mille alusel on võimalik genereerida dokumendimall kui ka kasutajale kuvatav dokumendimalli eelvaade mis on kuvatud joonisel 2. Samaaegselt kui kasutaja loob süsteemis dokumendimalli genereeritakse vastavat JSON dokumenti ning salvestamisel saadetakse JSON dokument serverisse.

2.6.5 PDF dokumendi genereerimine

TRUKKS AB kasutab PDF dokumentide genereerimiseks juba varem arendatud süsteemi, mis on eraldiseisev teenus ehitatud NodeJSi ja Pupeteer teeki kasutades. Antud teenus võimaldab saata HTML dokumendi koos stiililehega ning vastu saadetakse PDF dokument. Antud teenust on vaja täiustada selliselt, et see töötaks erinevate kirjatüüpidega. Kirjatüüpide kuvamiseks dokumendil tuleb need kirjeldada CSSis, selleks kasutatakse @font-face elementi, kus saab kirjeldada kirjatüübi nime ja koha kus kirjatüüp asub või lisada kirjatüüp kui base64 kodeeritud kirjena. Base64 on algorütm mis tranformeerib iga sümboli tähestikku mis koosneb ladina tähtedest,

numbridest pluss märgist ning kaldkriipsust [16]. Kirjatüüpe hoiustatakse andmebaasis BLOB andmetüübina, kirjatüüp tuleb valida andmebaasis ning kodeerida base64 kirjeks. Kirjatüübi lisamine CSS faili kasutades base64 kodeeringuga kirjet:

```
@font-face {
font-family: 'MyCustomFont';
src: url(data:[<mediatype>];charset=utf-8;base64,<base64data>);
}

html { font-family: 'MyCustomFont'; }
```

Joonis 9. Kirjatüübi lisamine CSS faili kasutades base64 kodeeringut

Olemasolev PDF dokumentide genereerimse lahendus Puppeteeri teeki kasutades toob kaasa probleemi, kus HTML dokumendid mis viitavad välistele CSS failidele ei lae PDF genereerimise hetkeks väliseid CSS faile ära.

Üks lahendus sellest probleemist vabanemiseks on teha kuvatõmmis enne PDF dokumendi genereerimist. Selline lahendus tooks kaasa vajaliku ootehetke, mille jooksul oleks võimalik välised CSS failid ära laadida. Samas on selle lahenduse puuduseks asjaolu, et iga dokumendi genereerimisega tekitatakse ka eraldi pildifail, mis ei ole tegelikult vajalik ja võtab lihtsalt ruumi. Seega tuleks omakorda tegeleda veel nende piltide kustutamise probleemiga.

Teine lahendus, mis sai ka antud hetkel valitud, on CSS kirjutada HTML dokumendi sisse <style> blokki. Sellisel juhul laeb süsteem kirjatüübid ära enne kui PDF genereeritakse.

3 Lahenduse realiseerimine

Järgnevas töö osas kirjeldab autor arendusprotsessi, peatükis 2 kirjeldatud ja leitud lahenduste realiseerimine koos koodinäidetega. Kasutajaliides on arendatud ReactJSi kasutades ning serveripool Kotlinit kasutades. PDF dokumentide teenus on arendatud kasutades NodeJSi. Autor kasutab samu tehnoloogiaid, et realiseerida dokumendimallide muutmise iseteeninduskeskkond TRUKKS AB süsteemis.

3.1 Andmebaasi täiendused

Peatükis 2.6.1 selgus, et TRUKKS AB süsteem kasutab MySQL andmebaasi. Selleks, et dokumendimallide iseteeninduskeskkond töötaks rakenduses on vajalik andmebaasi täiendada. Vajalik on tekitada andmebaasi tabelid mis hoiustavad dokumendimalle ning kirjatüüpe.

Süsteemis on kasutusel *Flyway* teek mis tegeleb andmebaasi migratsioonide haldamisega. *Flyway* töötab järgnevalt:

1. Otsida andmebaasist metaandmete tabel, mida manageerid *Flyway*, kui tabelit ei leita luuakse uus tabel
2. Koodist otsitakse migratsioonifaile mida jooksutada
3. Migratsioonifailide versiooninumbreid võrreldakse metaandmete tabeli vastu. Kui migratsioonifaili versiooninumber on madalam või võrdne kui metaandmete tabelis märgitud versioon siis faili ignoreeritakse
4. Migratsioonifailid mille versiooni number on suurem sorteeritakse versiooninumbri järgi ning käivitatakse järjekorras
5. Peale igat migratsioonifaili jooksutamist uuendatakse metaandemte tabelit

Flyway võimaldab kasutada migratsioonide tegemiseks SQLi või Java keelt. TRUKKS AB süsteemis kasutatakse migratsioonide tegemiseks SQLi. *Flyway* võimaldab lisada ning tagasi võtta migratsioone, lisaks saab kirjeldada korduvaid migratsioone. Selleks, et *Flyway* leiaks migratsiooni failid, oskaks neid järjestada ning õiget toimingut sooritada. Peavad olema migratsioonifailid väga täpselt nimetatud. Migratsioonifaili nimi peab algama toimingu eesliitega: V, U, R, kus V-ga märgitakse versiooni põhised migratsioonid, U-ga tagasivõtmisele kuuluvad migratsioonid ning R-iga korduvad

migratsioonid. Peale eesliidest järgneb versiooni number, kui tegemist on korduva migratsiooniga jäetakse nummerdus ära. Versiooni numbrile järgneb kaks alakriipsu mille järel võib meelepäraselt nimetada failinime, kus iga tühiku asemel kasutatakse ühte alakriipsu [17].

Korrektne migratsioonifaili nimi lisamaks vajalikud tabelid dokumendimallide muutmise iseteeninduskeskkonna töötamiseks: V0085__document_templating_system. Migratsioonifaili sisu on välja toodud alloleval joonisel 10.

```
CREATE TABLE font(
  id VARCHAR(36),
    instance_id VARCHAR(36),
    font MEDIUMBLOB,
    PRIMARY KEY pk_font (id),
    FOREIGN KEY fk_font_iid (instance_id) REFERENCES instance (id)
);

CREATE TABLE document_font(
    id VARCHAR(36),
    instance_id VARCHAR(36),
    font_id VARCHAR(36),
    font_name VARCHAR(255),
    font_type VARCHAR(255),
    PRIMARY KEY pk_document_font (id),
    FOREIGN KEY fk_document_font_fid (font_id) REFERENCES font (id),
    FOREIGN KEY fk_document_font_iid (instance_id) REFERENCES instance (id),
    FOREIGN KEY fk_document_font_dtid (document_template_id) REFERENCES
document_template (id)
);

CREATE TABLE document_template_font(
    instance_id VARCHAR(36),
    document_font_id VARCHAR(36),
    document_template_id VARCHAR(36),
    PRIMARY KEY pk_document_template_font (document_font_id,
document_template_id),
    FOREIGN KEY fk_document_template_font_dfi (document_font_id) REFERENCES
document_font (id),
    FOREIGN KEY fk_document_template_font_dti (document_template_id)
REFERENCES document_template (id)
);

CREATE TABLE document_template(
    id VARCHAR(36),
    instance_id VARCHAR(36),
    name VARCHAR(255),
        type VARCHAR(255),
    template JSON,
    PRIMARY KEY pk_document_template (id),
    FOREIGN KEY fk_document_template_iid (instance_id) REFERENCES instance(id)
);
```

Joonis 10. Migratsioonifail vajalike tabelitega

3.2 Kasutajaliidese realiseerimine

Peatükkides 2.4 ning 2.6.2 kirjeldatud kasutajaliidese ning analüüsi põhjal selgus, et dokumendimalli redigeerimine on vaja iseseisvalt arendada, kuna juba olemasolevad tekstiredaktorid ei toimi nii nagu soovitud.

Tööriista ribal kuvatud kirjasuuruse rippmenüü täidetakse staatiliselt, mis tähendab, et valikuvariandid on kirjutatud otse *front-end* rakendusse, kuid dünaamiliste väljade ning kirjatüüpide rippmenüüd tuleb täita serveri poolt saadava informatsiooniga.

Selleks, et kasutaja saaks mugavalt välju liigutada ja kujundada dokumenti kasutatakse kasutajaliidese tagataustal tabeli süsteemi, mille sees on võimalik liigutada dokumendi väljasid. Kasutaja saab tekitada dokumendile ridu ning veerge. Kasutaja saab lisada kuni viis veergu, mille sees saab omakorda joondada sisu vasakule, keskele, paremale. Autor kasutas realiseerimiseks CSS *Grid* kujunduse meetodit. *Gridi* abil on võimalik kerge vaevaga tekitada tabeleid ning neid dünaamiliselt muuta, kasutades eeldefineerituid CSS klasse.

Uue rea lisamiseks peab kasutaja vajutama tööriistaribal vastava nupu peale, kus saab valida mitme veeruga rida soovitakse, kui kasutaja valib kahe veeruga rea saab kasutaja valida kuidas rida jagada. Kas kaheks võrdseks osaks või üheks kolmandikuks ning kaheks kolmandikuks. Selline lähenemine annab kasutajale piisava paindlikuse dokumendi personaliseerimiseks.

Kahe veeru valimisel osadeks jagamiseks on defineeritud järgnevad CSS klassid:

```
.split-1-1 {
    grid-template-colums: 1fr 1fr;
}

.split-1-2 {
    grid-template-colums: 1fr 2fr;
}

.split-2-1 {
    grid-template-colums: 2fr 1fr;
}
```

Joonis 11. CSS klassid erinevate veergude kuvamiseks

3.3 Dokumendi genereerimise mootor

Kuna andmebaasi salvestatakse dokumendimall JSON kujul on vajalik päris dokumendi genereerimiseks dokumendi genereerimise mootor, mis suudab JSON failist luua korrektse HTML dokumendi mida saab kasutada PDF dokumendi genereerimisel või saata otse kasutajale. Selleks, et korrektset HTML dokumenti genereerida on vajalik, et JSONi struktuur oleks täpselt kirjeldatud.

Väljatöötatud dokumendimalli struktuur on jaotatud ridadeks ning veergudeks. Dokumendimall koosneb ühest kuni n arvust ridadest, kus iga rida koosneb 1 kuni 5 veerust ning iga veerg võib sisaldada välja objekti või ridu. Kui veerg koosneb välja objektist siis kuvatakse kasutajaliideses seda kui ühe realist veergu, kui aga veerg koosneb ridadest kuvatakse kasutajaliideses veergu mis koosneb mitmest reast. Reaga antakse kaasa *split* väärtus, mille alusel saab kindlaks teha mitu veergu on reas. *Split* võimalikud väärtused:

- 1 - tegemist on reaga, kus on üks veerg
- 1-1 - kahe veeruga rida, kus rida on võrdselt pooleks jagatud
- 1-2 - kahe veeruga rida, kus rida on jagatud üheks kolmandikuks ja kaheks kolmandikuks
- 2-1 - kahe veeruga rida, kus rida on jagatud kaheks kolmandikuks ja üheks kolmandikuks
- 3 - kolme veeruga rida
- 4 - nelja veeruga rida
- 5 - viie veeruga rida

Välja objekt määrab, mis tüüpi väljaga on tegemist kas pildiväli või tekstiväli. Kui tegemist on pildiväljaga on määratud pildi identifikaator, mis tähendab, et kõik pildid on süsteemi sisesed ja süsteem ei võimalda välise lingi kaudu pilti kasutada. Kasutaja saab oma meelepärase pildi laadida ülesse TRUKSS AB keskkonda, mis on varasemalt juba arendatud. Pilte hoiustatakse Amazon S3 teenuses. Tekstiväli koosneb tekstist ning valideerimise tingimusest. Igal väljal on võimalik määrata paiknemist: vasakul, paremal, keskel.

Korrektne JSON struktuur:

```
{
  rows: [
    {
      split: 1-2,
      cols: [
        { field: { type: "image",
                   value: "1",
                   alignment: "center" }},
        { rows: [
          { field: { type: "text",
                     value: "Arve nr: {{invoice.number}}",
                     eval: "invoice.number != null",
                     alignment: "right" }},
          { field: { type: "text",
                     value: "{{invoice.date}}",
                     alignment: "right" }}
        ]}
      ]
    }
  ]
}
```

Joonis 12. Dokumendimalli JSON struktuuri näide

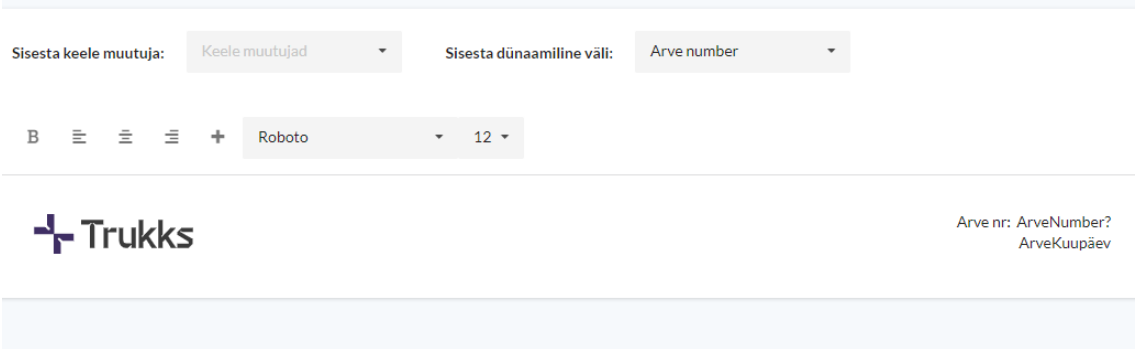
Üleval välja toodud struktuuris on näha, et tegemist on ühe reaga, mis on jaotatud kaheks ning jaotus on tehtud üheks kolmandikuks ning kaheks kolmandikuks. Esimeses veerus asub pilt mis on joondatud keskele ning teises veerus on kaks tekstivälja. Esimene tekstiväli koosneb staatilisest tekstist “Arve nr” ning dünaamilisest väljast mis lisab genereerimise ajal arve numbri. Kasutajaliideses kuvatakse antud välja “Arve nr: ArveNr?”. Lisaks on arve numbril valideerimise nõue, mis on kirjeldatud struktuuris. Teise veeru teine rida on dünaamiline väli arve kuupäev. Antud struktuuris on pildiväli joondatud veeru keskele ning tekstiväljad joondatud veeru paremasse serva.

Loodud struktuuri alusesel on võimalik genereerida HTML dokumente, selleks kasutatakse rekursiivset meetodit. Kasutajaliidese poolel mis on realiseeritud ReactJS-iga on loodud kaks peamist funktsiooni, ridade ja veergude genereerimiseks. ReactJS kood mis genereerib struktuurist kasutajale kuvatava dokumendimalli:

```
const renderRows = rows => rows.map(row => renderRow(row));

const renderCols = cols => cols.map((col, idx) => renderCol(col, idx + 1));

const renderRow = row => {
    const { split, cols } = row;

    return (
        <div contentEditable={true} className={`grid-row split-${split}`} >
            {renderCols(cols)}
        </div>
    )
}

const renderCol = (col, idx) => {
    const { field, rows } = col;

    return (
        <div className={`grid-col-${idx}`}>
            {field && renderField(field)}

            {rows && renderCols(rows)}
        </div>);
}
```

Joonis 13. ReactJSi kood, mis genereerib kasutajaliideses dokumendimallide muutmise HTMLi

Esimesena kutsutakse välja *renderRows* meetod ning seejärel genereeritakse vastavalt struktuurile HTML dokument. *RenderCol* meetodis *renderField* meetod tagastab vastavalt tüübile korrektse HTMLi. kui aga *renderCol* meetodi sisendiks on read kutsutakse uuesti renderCol meetod välja. Antud struktuuriga genereeriti järgnev kasutajaliides:



Joonis 14. JSON struktuurist dokumendi genereerimise mootori abil genereeritud kasutajaliides koos tööriistaribaga.

Dünaamilised väljad asendatakse *renderField* meetodis andmete vastu. Kasutajaliidese poolel on selleks eelnevalt defineeritud väärtused või ettevõtte andmed. Kuid serveris genereeritavas dokumendis asendatakse dünaamilised väljad õigete andmetega.

3.4 Tulemus

Töö kaigus analüüsiti võimalike lahendusi probleemi lahendamiseks. Analüüsiti peamisi probleeme ning leiti neile lahendus. Praktilise väärtusena valmis prototüüp mida on võimalik näidata esimestele valitud klientidele ning koguda tagasisidet. Prototüübi põhjal on võimalik arendada täielik süsteem mis rahuldab töös esitatud nõuded.

Autor usub, et töös leitud lahenduste abil on võimalik püstitatud probleem lahendada ning arendada dokumendimallide muutmise iseteeninduskeskkond.

Järgmiste sammudena näeb autor süsteemi täielikku implementeerimist TRUKSS AB süsteemi, kus kasutajad saavad dokumente kujundada vastavalt soovile. Lisaks on võimalik antud probleemist ning väljatöötada eraldiseisev teenus, mis võimaldaks erinevatel IT-süsteemidel implementeerida dokumendimallide muutmise iseteeninduskeskkonda.

4 Kokkuvõte

Käesoleva töö eesmärgiks oli arendada dokumendimallide muutmise iseteeninduskeskkond, kus TRUKKS AB kasutajad saavad meelepäraseid dokumente koostada.

Eesmärgi saavutamiseks uuris autor klientidelt ning ettevõtte esindajatelt milline peaks loodav süsteem olema. Ettevõtte ning klientide nõuete põhjal formuleeriti funktsionaalsed ning mitte-funktsionaalsed nõuded. Nõuete põhjal otsiti võimalike lahendusi ning analüüsiti võimalusi.

Analüüsi käigus leiti kolm teenust, mis pakuvad dokumentide ja dokumendimallide haldussüsteemi, kuid antud lahendused ei lahendanud püstitatud probleemi. Analüüsi käigus selgus, et puuduvad head lahendused mis suudaks probleemi lahendada ja otsustati, et süsteem tuleb arendada algusest lõpuni autori poolt. Autor implementeeris prototüübi lahendamaks töös püstitatud probleemi. Prototüübi alusel on võimalik edasi arendada reaalne süsteem, mis lahendab ettevõtte TRUKSS AB probleemi.

Töös püsitatud eesmärk täideti, kuna selgitati välja kuidas antud süsteemi realiseerida ning esitati kasutajaliidese näidis. Lisaks valmis prototüüp mille aluses on võimalik implementeerida täielikult dokumendimallide muutmise keskkond. Autor jätkab süsteemi arendamist ning arendab täielikult töös kirjeldatud süsteemi.

Kasutatud kirjandus

[1] M. Mällo, K. Prik ja S. Sillavee, „Recap of 2019 in the Estonian startup sector," Startup Estonia, 17.2.2020. [Võrgumaterjal]. https://startupestonia.ee/blog/recap-of-2019-in-the-estonian-startup-sector. [Kasutatud 27.4.2020].

[2] CBINSIGHTS, „The Top 20 Reasons Startups Fail," 6.11.2019. [Võrgumaterjal]. https://www.cbinsights.com/research/startup-failure-reasons-top/. [Kasutatud 5.4.2020].

[3] Legito, „Kodulehekülg," [Võrgumaterjal]. https://www.legito.com/US/en/. [Kasutatud 25.4.2020].

[4] Legito, „Legito API," [Võrgumaterjal]. https://app.swaggerhub.com/apis-docs/Legito/legito-api/1.0. [Kasutatud 25.4.2020].

[5] Templafy, „Kodulehekülg," [Võrgumaterjal]. https://www.templafy.com/. [Kasutatud 25.4.2020].

[6] Templafy, „API dokumentatsioon," [Võrgumaterjal]. https://api-v1.templafy.com/swagger/ui/index. [Kasutatud 25.4.2020].

[7] Windward studios, „Kodulehekülg," [Võrgumaterjal]. https://www.windwardstudios.com/solution/windward-engine?utm_expid=.L9C-jW9VSSyeR2OHZLV7cA.0&utm_referrer=. [Kasutatud 25.4.2020].

[8] WhatFontIs, „All fonts," [Võrgumaterjal]. https://www.whatfontis.com/all-fonts.html. [Kasutatud 10.4.2020].

[9] Mozilla MDN web docs, „Web fonts," [Võrgumaterjal]. https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text/Web_fonts. [Kasutatud 10.4.2020].

[10] Google, „Fonts," [Võrgumaterjal]. https://fonts.google.com/. [Kasutatud 10.4. 2020].

[11] MySQL, „Data Type Storage Requirements," [Võrgumaterjal]. https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html. [Kasutatud 27.4.2020].

[12] MySQL, „Optimizing for BLOB types," [Võrgumaterjal]. https://dev.mysql.com/doc/refman/8.0/en/optimize-blob.html. [Kasutatud 27.4.2020].

[13] Thymeleaf, „Thymeleaf dokumentatsioon," [Võrgumaterjal]. https://www.thymeleaf.org/. [Kasutatud 25.4.2020].

[14] Facebook Inc, „ReactJS dokumentatsioon," [Võrgumaterjal]. https://reactjs.org/docs/introducing-jsx.html . [Kasutatud 25.4.2020].

[15] Acunetix, „Cross-site scripting (XSS)," [Võrgumaterjal]. https://www.acunetix.com/websitesecurity/cross-site-scripting/. [Kasutatud 12.4.2020].

[16] Base64 Guru, „What is base64?," [Võrgumaterjal]. https://base64.guru/learn/what-

is-base64. [Kasutatud 17.4.2020].

[17] Redgate, „Flyway dokumentatsioon,“ [Võrgumaterjal]. https://flywaydb.org/documentation/migrations. [Kasutatud 28.4.2020].