TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Tamberg 179501IAIB

MIKROTEENUSENA KASUTATAV PDF FAILIDE GENEREERIMISE RAKENDUS

Bakalaureusetöö

Juhendaja: Evelin Halling

tarkvarateaduse instituut, PhD

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: Karl Tamberg

18.05.2020

Annotatsioon

Paljud süsteemid vajavad võimalust automaatseks PDF failide genereerimiseks. Selleks luuakse mall, kus kirjeldatakse ära trükise staatiline sisu, disain, loogika ning dünaamilised andmed. Jättes kogu malli haldamise vastutuse arendajatele, muutub mallis muudatuste tegemine aeganõudvaks ja kalliks.

Käesoleva töö eesmärgiks on luua rakendus, mis lubaks lihtsamaid muudatusi mallides teha ka äri poolel. Töö käigus sai analüüsitud erinevaid võimalusi PDF failide genereerimiseks ja paika pandud nõuded, millele loodud rakendus vastama peab.

Töö lõpuks valmis klientrakendusest, serverirakendusest ning andmebaasist koosnev PDF failide genereerimiseks mõeldud teenus. Loodud rakendus on kasutatav mikroteenuste arhitektuuris ning toetab erinevate keskkondade kasutamist. Rakendus jätab arendajale võimaluse kasutada kõikvõimalike madala taseme tööriistu malli struktuuri ja disaini paika panemisel, kuid lubab lihtsamaid muudatusi malli tekstis ja disainis teha mugavalt läbi kasutajaliidese.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 7 peatükki, 12 joonist.

Abstract

PDF file generation service, usable in microservice architecture

Many systems need a way to automatically generate PDF files. For that, a template is used, which describes the static text, design, logic and handling of dynamic data for the printout. Leaving the full management of the templates to developers makes any changes to the template time–consuming and expensive.

The aim of this thesis is to create an application, which allows business side to make simple changes in the template. Possibilities for generating PDF files were analysed and requirements were set, which the finished application must meet.

The product is a service meant for PDF file generation, which consists of client–side application, server–side application and database. This service is usable in microservice architecture and supports using many different environments. The application allows developers to use all low–level tools to create the design and structure of the template. At the same time, the application allows simple changes in the text and design to be made through the user interface.

The thesis is in Estonian and contains 34 pages of text, 7 chapters, 12 figures.

Lühendite ja mõistete sõnastik

PDF	*Portable Document Format.* Dokumendiformaat.
SaaS	*Software as a Service.* Tarkvara kui teenus mudel..
XML	*Extensible Markup Language.* Laiendatav märgistuskeel.
API	*Application Programming Interface.* Rakendusliides.
MIT	*Massachusetts Institute of Technology.* Massachusettsi Tehnoloogiainstituut.
IDE	*Intelligent Development Environment.* Arenduskeskkond.
REST	*Representational State Transfer.* Tarkvaraarhitektuuri laad.
URI	*Uniform Resource Identifier.* Infoallika identifitseerimiseks kasutatav sõne (ühtne ressursi identifikaator).
JSON	*JavaScript Object Notation.* Failiformaat, milles hoitakse võti-väärtus paare.
SEO	*Search Engine Optimisation.* Otsingumootori optimeerimine.
JWT	*JSON Web Token.* Autentimisel väljastatav luba, mis sisaldab kasutaja andmeid.
CSS	*Cascading Style Sheets.* Veebilehtede kujunduse loomiseks kasutatav märgistuskeel.
CDN	*Content Delivery Network.* Sisuedastusvõrk.
CORS	*Cross–Origin Resource Sharing.* Mehhanism, mis lubab brauseril pääseda ligi teise aadressiga allika ressurssidele.
CSRF	*Cross–Site Request Forgery.* Päringuvõltsingu rünnak.
SQL	*Structured Query Language.* Struktuurpäringukeel.
ORM	*Object–Relational Mapping.* Objekt–relatsiooniline vastendamine.
DSL	*Domain-Specific Language.* Domeenispetsiifiline keel.
Copyleft	Litsentsi tüüp, mis käsib teose põhjal loodud uuele teosele anda sama litsentsi.
JSX	*JavaScript XML*. JavaScripti laiendus, mis lubab HTML notatsioonide kasutamist.
ms	Millisekund.
HTML	*HyperText Markup Language*. Veebilehtede märgenduskeel.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Automatiseeritud PDF failide genereerimine erinevate lepingute, arvete, raportite, teavituste või muude dokumentide jaoks on vajalik paljudes erinevates süsteemides. Selleks kasutatakse malle, mis sisaldavad endas äripoolsete nõuete järgi koostatud reegleid, staatilist teksti, disainerite poolt ette antud disaini ning dünaamilist sisu. Trükise genereerimisel võetakse mall, lisatakse sinna dünaamiline tekst ettenähtud kohale ning luuakse dokument.

Mallide loomiseks on mitmeid erinevaid lahendusi, kuid üldjuhul on vaja mall luua arendaja poolt ning ka malli muudatusi teha arendajatel. Kui erinevaid trükiseid ei ole väga palju, siis on selline lähenemine mõistlik, aga mallide arvu kasvades on vaja paratamatult tihti mõnes mallis teha muudatusi. Kui muudatuse enda tegemine ei ole hästi disainitud süsteemi puhul suureks probleemiks, siis selliste muudatustega võib kaasneda nõudeid, mille täitmine on ajakulukas ning ebamugav. Näiteks seadusemuudatuse tõttu tehtavad parandused peavad jõudma süsteemi kindlast kuupäevast, mis tähendab, et nende muudatuste produktsioonikeskkonda panemisega tuleb tegeleda öösel.

Turul on saada ka tasulisi lahendusi, mis võimaldavad graafilise liidese kaudu igaühel malle luua ning muuta, kuid nende lahendustega kaasnevad ka mitmed piirangud mallidele. Sellised rakendused kasutavad äriloogika kirjeldamiseks domeenipõhist keelt (*DSL*), mis keerulise loogika korral võib muutuda väga raskesti hallatavaks. Lisaks on graafilise liidesega trükiste genereerimise tarkvaral üldjuhul puudu palju võimalusi, mida arendajatele suunatud teegid võimaldavad.

Käesoleva töö raames luuakse vabavaraline PDF failide genereerimiseks mõeldud rakendus, mis koosneb serverirakendusest ja klientrakendusest. Loodava rakenduse eesmärgiks on pakkuda lahendust, kus mallide loomine jäetakse arendajatele, kuid staatilise teksti ning väiksemate disainimuudatuste tegemist võimaldatakse ka teistel süsteemi kasutajatel.

2 Taustainfo

Trükiste genereerimise vajadus tekib varem või hiljem kõikides infosüsteemides, mis kliendile mingil kujul dokumente esitavad. Kliendile esitatavad dokumendid genereeritakse üldjuhul PDF failidena, sest selles formaadis failid on lihtsasti avatavad peaaegu igas seadmes. PDF failiformaat on lisaks väga mitmekülgne, võimaldades kuvada väga erinevat sisu, olgu selleks tekst, tabel või pildid.

Panga infosüsteem on hea näide olukorrast, kus on vaja hallata väga suures koguses erinevaid trükiseid. Näiteks ühes Eestis tegutseva panga süsteemis on viie riigi peale ainuüksi laenudega seotud trükiste malle rohkem kui 80. Seejuures kasutatakse ühte malli mitmete erinevate sarnase sisuga dokumentide loomiseks, ehk tegelik trükiste arv on veel kordades suurem.

2.1 Valmislahendused

Otsides lahendust, mis võimaldaks malli kirjeldamise järgselt selle malli põhjal trükiseid luua ning hallata, leiab palju tasulisi dokumentide genereerimiseks loodud teenuseid. Seejuures leiab ka palju vabavaralisi teeke, mille abil malle luua, kuid lihtsalt kasutusse võetavaid valmislahendusi on vähe. Valmislahenduste uurimisel keskenduti peamiselt vabavaralistele trükiste genereerimise teenustele, sest selle töö käigus loodud rakendus on samuti vabavaraline ning seega konkureeriks teiste vabavaraliste lahendustega. Seejuures sai katsetatud ka tasulisi lahendusi.

2.1.1 JasperReports

JasperReports[1] on vabavaraline raportite genereerimiseks mõeldud teek, mida saab kasutada mitmes erinevas formaadis dokumentide genereerimiseks. Teek kasutab XML formaadis *.jrxml* laiendiga faile trükiste mallidena. Jaspersoft[2] pakub lisaks

[1] https://community.jaspersoft.com/project/jasperreports-library

[2] https://www.jaspersoft.com/

JasperReports teegile tasuta ka Jaspersoft Studio[1] töölauarakendust, mis võimaldab mallide loomist ning muutmist graafilise liidese kaudu. JasperSoft Studio rakenduse kasutajaliides on näha joonisel 1.

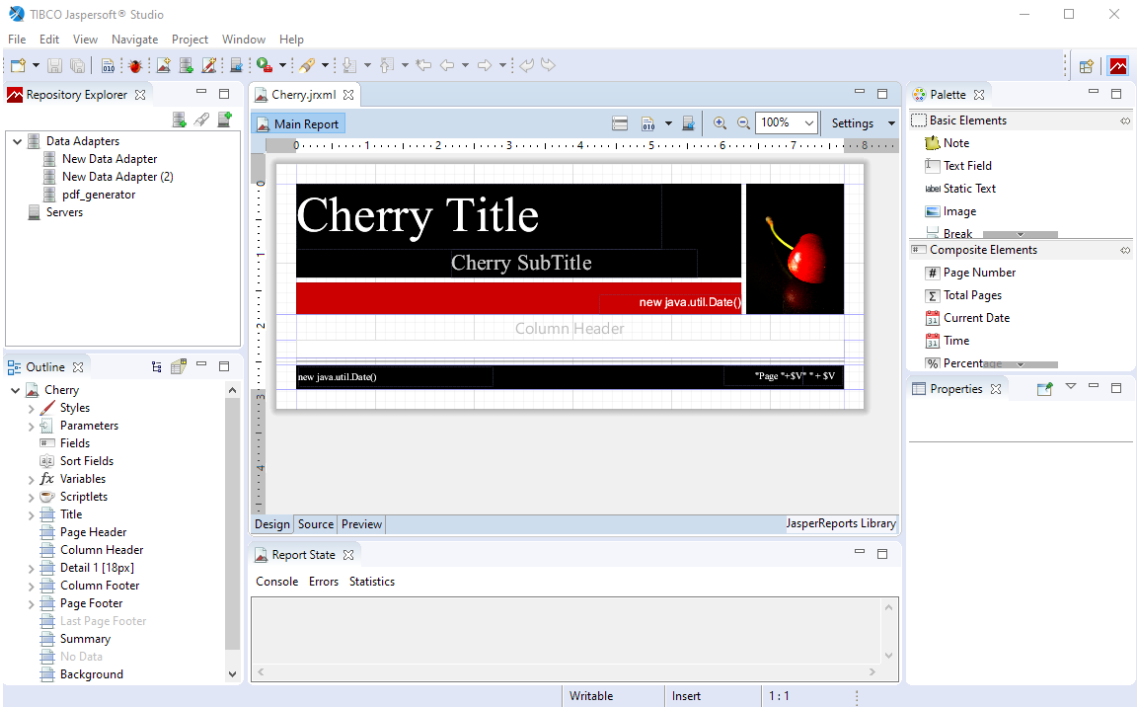
Teegi tugevused:

- Võimaldab genereerida mitmeid erinevaid failiformaate.
- JasperSoft Studio töölauarakendus võimaldab malli muutmist graafilise liidese kaudu.
- JasperSoft Studio töölauarakendus võimaldab andmebaasiga ühendudes andmeid eelvaates kuvada.
- Dokumentatsioon on hea.

Teegi nõrkused:

- PDF failidele spetsiifiliste elementide, näiteks kasutaja poolt täidetavate vormiväljade, lisamine ei ole mugav.
- Keskendutakse raportite genereerimise funktsionaalsusele, muud tüüpi failide loomine on keerulisem ning aeganõudev.
- JasperSoft Studio kasutamine on keeruline ning vajab tehnilisi teadmisi. [1]
- Kasutab teegi spetsiifilist XML keelt, mille õppimine võtab aega.
- Trükiste genereerimine serveris nõuaks palju teegispetsiifilisi teadmisi ning mallide käsitsi muutmist. Näiteks JasperSoft Studio kaudu malli luues kirjutatakse ka SQL päringud XML faili sisse, mida on võimalik serveri pool teistmoodi lahendada.

[1] https://community.jaspersoft.com/project/jaspersoft-studio



Joonis 1. JasperSoft Studio töölauarakenduse kasutajaliides.

2.1.2 Carbone

Carbone[1] on Apache-2 litsentsiga dokumentide genereerimiseks mõeldud teek. See teek võimaldab luua malli LibreOffice Writer'i või Microsoft Word'i abiga ning kasutab lihtsat domeenipõhist keelt (*DSL*), et dünaamilist infot kuvada ning äriloogikat lisada. Näide Carbone mallist on joonisel 2.

Teegi tugevused:

- Väga lihtne kasutada.
- Dokumentatsioon on hea.
- Eksisteerib eraldi brauseris töötav rakendus eelvaate nägemiseks[2].

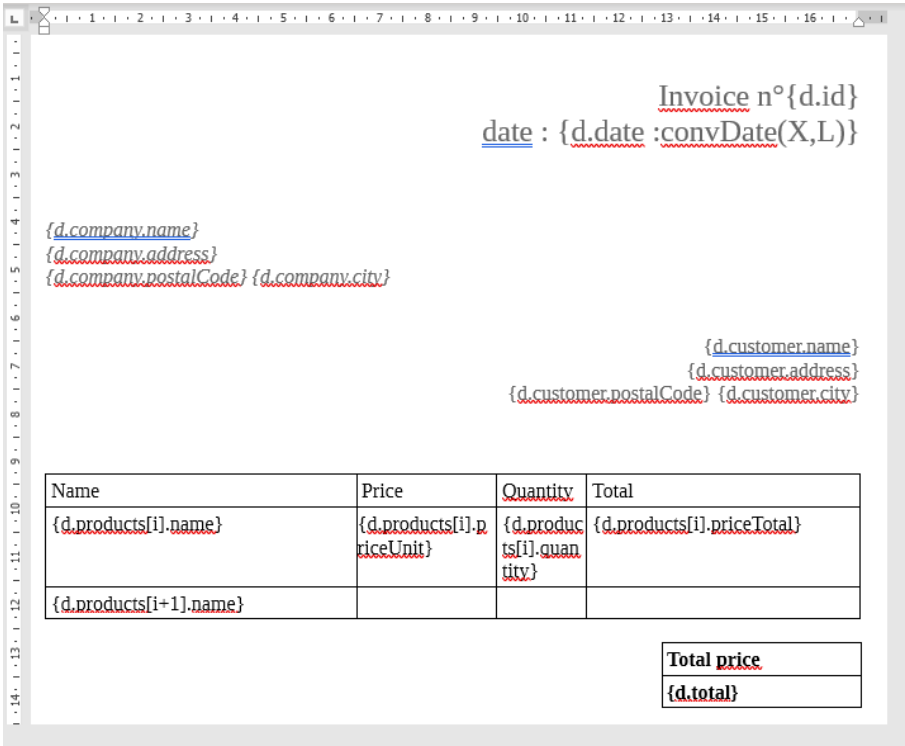
Teegi nõrkused:

- Aeglane, näiteks lihtsa 1 leheküljelise arve genereerimine võttis üle 200 ms, sarnase trükise genereerimine koodi kirjutatud malli puhul võttis alla 25 ms.
- Brauserirakendus eelvaate nägemiseks on tasuline.

[1] https://carbone.io/

[2] https://account.carbone.io/

- Ei toeta mitmeid PDF spetsiifilisi võimalusi, näiteks kasutaja poolt täidetavate vormiväljade lisamist.



Joonis 2. Carbone malli näide.

2.1.3 PDF Generator API

PDF Generator API[1] on tasuline rakendus, mis on mõeldud kasutamiseks tarkvara kui teenus (*SaaS*) mudelil. Vaadeldud tasulistest rakendustest pakkus see teek kõige mugavamat kasutajaliidest malli loomiseks ning muutmiseks. Rakenduse kasutajaliides on kuvatud joonisel 3. Teenust pakkuva ettevõtte kontor asub Eestis.

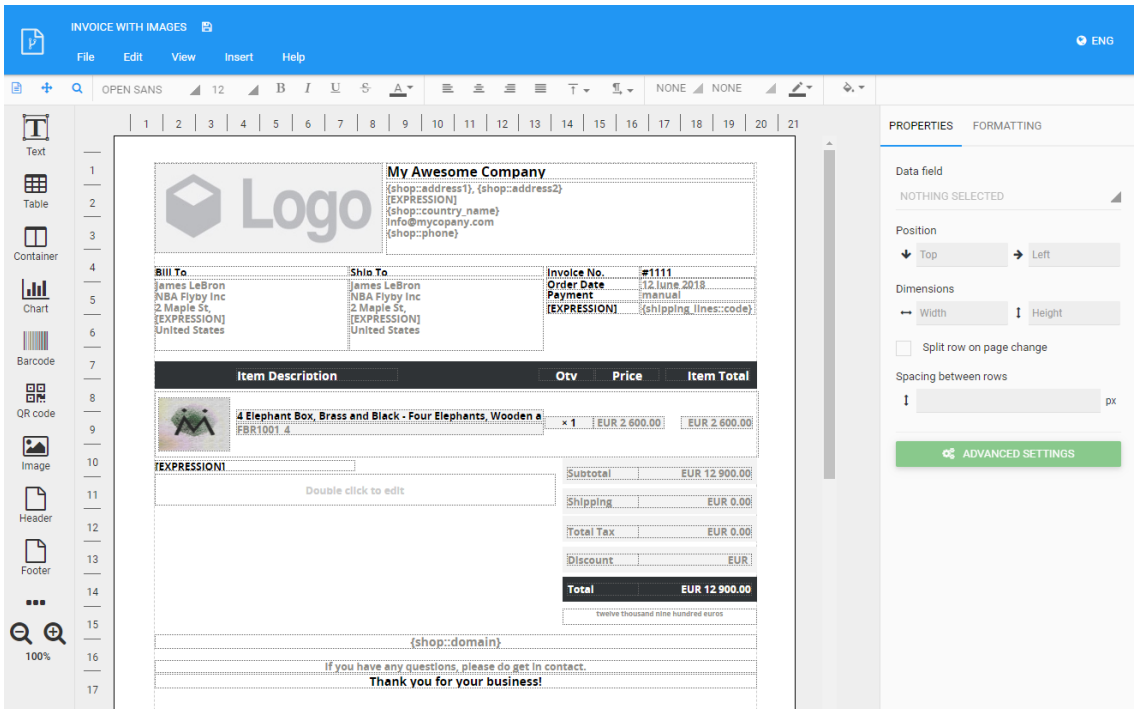
Lahenduse tugevused:

- Mallide tegemise ja muutmise keskkond on lihtsalt kasutatav.
- Vaatamata keskkonna lihtsusele võimaldatakse seal teha paljut.
- Küsides pakuti võimalust ka rakenduse lokaalseks jooksutamiseks, mis võimaldab teenust kasutada andmeid oma serveritest välja saatmata.

Lahenduse nõrkused:

[1] https://pdfgeneratorapi.com/

- Tasuline.
- Keerulisemad loogikad, näiteks laenugraafik, kus on mitu tsüklit üksteise sees, ei ole graafilise liidese kaudu päris igasuguse disainiga loodavad.



Joonis 3. PDF Generator Api kasutajaliides.

3 Analüüs

Analüüsi käigus said paika pandud nõuded, mille abil saaks töö tulemust valideerida. Palju soovitusi tuli kolleegidelt, kellel on aastatepikkune kogemus trükiste loomisega finantssektoris. Seejuures sai sellel teemal arutatud ka ühe Eestis tegutseva panga süsteemiarhitektiga, mis aitas paika pandud nõudeid valideerida.

Samuti sai testitud erinevaid meetodeid ning teeke, mille abil PDF faile genereerida. Teekide tugevuste ning nõrkuste põhjal sai nõuetele vastava rakenduse loomiseks valitud parim teek.

3.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded kasutajaliidesele:

- Ülevaate andmine mallidest. Suure mallide koguse korral on oluline, et oleks võimalik näha olemasolevate mallide teksti ja kujundust ka ilma selleta, et oleks vajadus jäljendada (*mock*) dünaamilisi sisendandmeid.
- Teksti muudatuse tegemise võimaldamine mallides. Tihti on vaja muuta mallis staatilist teksti ning seda muudatust on vaja kindlast kuupäevast, näiteks seadusemuudatuse korral. Selliste muudatuste tegemine ei tohiks kiirkorras koodiparandusi (*hotfix*) nõuda.
- Mitmekeelsuse tugi. Mallist trükise genereerimisel peab olema võimalus valida ka teksti keelt ning keele lisamine peaks tähendama vaid tõlkimise vaeva, mitte täiesti uue malli loomist.
- Rollide tugi. Paljudel inimestel võib olla vajadus trükiste vaatamiseks, aga trükiste muutmise võimalus produktsioonikeskkonnas võiks olla üksikutel kasutajatel.

Funktsionaalsed nõuded serverirakendusele:

- Mitme erineva keskkonna tugi. Ükskõik millises keskkonnas tehtud disaini- või tekstimuudatused peavad olema lihtsasti üle kantavad ka teistesse keskkondadesse.
- Kasutatav eraldiseisvana kasutajaliidesest. Kui klientrakendus soovitakse oma süsteemi integreerida või ümber teha, siis serverirakendus peab toimima edasi ja klientrakenduse muudatused ei tohi kuidagi serverirakenduse tööd segada.
- Mallide modulaarsus. Suur osa elemente, mida kasutatakse kümnetes või sadades erinevates mallides, korduvad (näiteks päis, jalus, laenulepingute puhul graafik) ja seega ei ole mõistlik seda osa igas mallis korrata.
- Mallide mugav refaktoreerimine. Igasugused muudatused sisendandmetes või mallinimedes peavad olema mugavalt tehtavad.
- Interaktiivsete PDF failide loomise võimalus. Mitmed kasutusjuhud nõuavad PDF faili, kus klient saab osa välju vajadusel oma arvutis ära täita, seejärel faili allkirjastada ning tagasi saata. Selline vajadus on näiteks liisingu puhul vara üleandmise trükistel, kuhu vara saaja peab märkmeid tegema.
- Mallide kiire ja mugav loomine. Finantsasutuste puhul tähendab üldjuhul uue tootega turule tulemine ka uute trükiste loomist. Uue 2-3 leheküljelise lepingu loomine peaks olema tehtav kiiremini kui ühe tööpäevaga.
- Kõikvõimalike disainivariantide tugi. Kasutatav vahend ei tohiks piirata disainivõimalusi.

Nendest viimased kaks nõuet on veidi konfliktsed, sest kui tahta mallide mugavat ning kiiret loomist, siis peaks see käima kasutajaliidese kaudu. Graafilised tööriistad aga loomu poolest ei võimalda üldjuhul päris kõike seda, mida madalamal tasemel koodis teha on võimalik. Töö käigus keskenduti sellele, et disainivõimalused ei oleks piiratud, kuid seejuures jäeti mallide loomise protsess võimalikult lihtsaks.

3.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded projektile:

- Loodud kood peab jääma vabavaraliseks ning olema avaldatav lihtsa lubava litsentsiga.
- Klientrakendus peab töötama kõigis moodsates laialt kasutuses olevates brauserites (Google Chrome, Mozilla Firefox, Microsoft Edge ja Safari).

- Ühe 2-3 leheküljelise lihtsa dokumendi loomine peaks käima alla 100ms. Kui jooksutada protsessi, mis genereerib ning saadab välja kõik päeva jooksul tekkinud arved, siis aeglasem lahendus võib suurte trükiste koguste juures tekitada probleeme.

3.3 Litsentsid

Ühe nõudena sai määratud, et tarkvara oleks avaldatav lihtsa lubava litsentsiga, milleks sai valitud MIT litsents. Kasutatavate teekide valimisel arvestati, et kasutatavate teekide litsentsid MIT litsentsiga vastuollu ei läheks. Probleemide vältimiseks ei kasutata töös teeke, mis on litsentseeritud tugeva *Copyleft* litsentsiga (GPL, AGPL). Küll aga kasutatakse nõrga *Copyleft* litsentsiga (LGPL) teeke, kuid selliste teekide lähtekoodis muudatusi ei tehta. Et vältida probleeme, on projekti kirjelduse juures ka eraldi esile tõstetud LGPL litsentsiga teegid, mida töös kasutatud on. [2]

3.4 PDF failide genereerimise võimalused

Ülevaate saamiseks, milliseid võimalusi on PDF failide genereerimiseks, tuli läbi vaadata suur hulk erinevaid teeke ning teekide kombinatsioone. Mitmed teegid on suunatud väga kitsale kasutajaskonnale, kuid selle ülesande raames keskenduti üldotstarbeliseks kasutamiseks mõeldud teekidele. Suures pildis jagunevad kõik proovitud võimalused neljaks, järgnevalt kirjeldatakse erinevaid lähenemisi ja nende tugevusi ning nõrkusi.

3.4.1 Mall koodis

Malli koodis hoidmine toob endaga kaasa palju tugevusi, näiteks saab probleemide korral kasutada koodi silumiseks mõeldud vahendeid, genereerimine on kiire ning mallid on lihtsalt refaktoreeritavad ja modulaarsed. Lisaks võimaldab selline lähenemine kasutada kõiki arenduskeskkonna (*IDE*) poolt pakutavaid tööriistu ka mallide arenduse juures ja lihtsustab teegi õppimist. Teisest küljest nõuab mallide loomine ja suuremate struktuursete muudatuste tegemine arendaja poolset sekkumist. Sellise lahenduse juures tuleks mõelda sellele, et mallide kirjeldamise kood ei tohiks liiga tihedalt olla seotud ülejäänud süsteemi osadega.

- IText[1]: Pika ajalooga tuntud teek, mis on saadaval erinevatele programmeerimiskeeltele. Selle kasutamine oli välistatud seetõttu, et ta on litsentseeritud AGPL litsentsiga. Üheks suurimaks tugevuseks on väga hea dokumentatsioon ning näidete küllasus.
- LibrePDF[2]: IText teegi haru (*fork*) sellest ajast, kui tegemist oli veel LGPL litsentsi all oleva teegiga. Seejärel on seda teeki eraldiseisvalt edasi arendatud, kuid sellest hoolimata IText versioonide 4 ja 5 dokumentatsioon ning näited töötavad suures osas ka LibrePDF teegi puhul.
- PdfBox[3]: Hea teek PDF failide manipuleerimiseks, kuid failide disainimiseks mugavad tööriistad puuduvad. Selle probleemi lahendamiseks on eraldiseisvaid kolmanda osapoole teeke[4], kuid nad jäävad LibrePDF võimalustele alla.

3.4.2 Mall dokumendis

Mitmed lahendused võimaldavad Microsoft Word'i või LibreOffice Writer'i abil trükiste mallide loomist. Sellist lähenemist kasutavad ka mitmed tasulised lahendused, näiteks Windward Studios[5]. Tasuta lahendustest on saadaval peatükis 2.1.2. kirjeldatud Carbone ja XDocReport[6] kombineerituna Apache Freemarker'iga[7].

XDocReport koos Apache Freemarker'iga võiks teoorias pakkuda rohkem võimalusi kui Carbone, sest Freemarker'i eelprotsessor (*template engine*) on võimekam. Testides selgus aga, et XDocReport koos Apache Freemarker'iga ei olnud töökindel lahendus. Näiteks teatud disainielementide kasutamisel mallis tekkisid vead ja trükiste genereerimine enam ei toiminud. Seejuures mallile peale vaadates ei olnud võimalik aru saada, et seal midagi valesti oleks. Carbone seevastu toimis testimisel probleemideta.

[1] https://itextpdf.com/en

[2] https://github.com/LibrePDF

[3] https://pdfbox.apache.org/

[4] https://github.com/ralfstuckert/pdfbox-layout

[5] https://www.windwardstudios.com/

[6] https://github.com/opensagres/xdocreport

[7] https://freemarker.apache.org/

3.4.3 XML mallid

Malli XML failis hoidmist võimaldavaid lahendusi on palju. Selline lähenemine aitab eraldada malli koodist ning malli mugavalt teenuste vahel edastada. Samuti võimaldab malli XML failis hoidmine malle versioneerida. Teisest küljest on XML failides olevat üsna keeruline modulaarseks muuta ning refaktoreerida. Probleeme tekib ka siis, kui mall sisaldab palju loogikat ning ühte malli kasutatakse mitme erineva dokumendi loomiseks. See võib juhtuda näiteks olukorras, kus on palju väga sarnaseid dokumente. Sellisel juhul kaob kiiresti ülevaade, mida millal ja kus kuvatakse.

- XSL–FO[1] koos Apache FreeMarker'iga: Lahendus, mida kasutavad Eestis mitu ettevõtet mallide loomiseks. XSL–FO teegi abil pannakse paika dokumendi disain ning Apache Freemarker on eelprotsessor, mis pakub väga palju võimalusi loogika lisamiseks. Apache FreeMarker on hästi dokumenteeritud, kuid disaini paika panemine XSL–FO abil on üsna ebamugav. Ei võimalda interaktiivsete PDF failide loomist.
- JasperSoft Reports: Teek, mille võimalusi sai täpsemalt kirjeldatud peatükis 2.1.1. Võimalik kasutada ka ilma JasperSoft Studio rakenduseta kasutades vaid teegipõhiseid XML faile.

3.4.4 HTML koos CSS'iga

Arvestades, et PDF failide genereerimisega kaasnevad probleemid on suures osas väga sarnased veebilehe loomisega seotud probleemidele, siis oleks väga mugav kasutada veebilehtede loomiseks mõeldud tööriistu PDF failide genereerimisel. Sellise lähenemise juures kaoks üldjuhul vajadus ka midagi juurde õppida, sest suurem osa arendajaid juba oskavad veebilehti disainida. Peamiseks probleemiks on seejuures, et mitmed trükise ja veebilehe elemendid vajavad erinevat lähenemist. HTML ei võimalda mõistlikult kirjeldada näiteks trükiste päiseid, jaluseid ning ääriseid. Samuti tekib probleeme CSS stiilide üle kandmisega trükisele: osa kirjeldatud stiile läheb kaotsi ning osa näeb trükisel teistsugune välja kui brauseris.

[1] https://xmlgraphics.apache.org/fop/fo.html

- Dompdf[1]: Üks mitmetest headest teekidest, mis HTML ja CSS failidest PDF faile genereerida võimaldab. Kui osata arvestada nende CSS võimalustega, mida teek ei toeta, on võimalik toota üsna hea välimusega dokumente. Oluliste dokumentide, nagu lepingute, genereerimiseks see siiski sobiv pole, sest korrektsete päiste ja jaluste lisamine on keeruline ning täpse elementide paigutamise saavutamine ei ole mõistlikult tehtav. [3]

3.4.5 Muud lahendused

Samuti on saadaval mitmeid lahendusi, mis on mõeldud spetsiifilisteks kasutusjuhtudeks, näiteks LaTeX või Markdown keelest PDF failide genereerimiseks. Sellised lahendused on aga väga piiratud kasutusjuhtudega. Lisaks on teeke, mis kombineerivad varasemalt mainitud viise. Näiteks kombineerib React-pdf[2] raamistik malli koodis hoidmise ning HTML'i ja CSS'i kasutamise, võimaldades kasutada React veebiarenduse raamistiku tööriistu mallide loomiseks.

- React–pdf: Võrdlemisi uus, kuid populaarsust koguv teek, mis võimaldab React raamistiku JSX keele abil kokku panna PDF faili malli, kasutades sealjuures JavaScript'i programmeerimiskeelt ning valikut CSS disainitööriistadest.

3.4.6 Teegi valik

Nõudeid arvesse võttes sobisid testitud variantidest töö tegemiseks JasperReports ning LibrePdf. Need teegid pakkusid võimaluste poolest kõike olulist ning mõlema puhul oli lihtne leida dokumentatsiooni.

JasperReports teegil on palju tugevaid külgi: arendajad saaksid olemasoleva JasperSoft Studio abil mallid valmis teha ning loodav rakendus lubaks lihtsaid muudatusi läbi kasutajasõbralikuma kasutajaliidese. JasperSoft Studio võimaldas küll väga kergelt baasist tuleva info kuvamist, kuid kui nullist oma disainiga malli looma hakata, siis võttis malli loomine kohati rohkemgi aega kui malli koodis kirjeldades. Selle põhjuseks oli liialt keeruliselt kasutatav kasutajaliides ning probleemid andmete adapteriga (*Data adapter*). Nimelt iga andmebaasimuudatus nõudis mitut käsitsi tehtavat sammu, et

[1] https://github.com/dompdf/dompdf

[2] https://react-pdf.org/

muudatused ka rakendusse jõuaksid. Lisaks nõuaks selle teegi kasutusse võtmine kõigi projektiga seotud arendajate jaoks paljude teegispetsiifiliste lähenemiste õppimist. [4]

LibrePdf teegi võimaluste arv ning kasutamismugavus olid ülejäänud proovitud variantidest paremad. Malli loomine läks väga lihtsalt, sest IText teegi dokumentatsioon ja näited andsid hea ülevaate teegi kasutamisest. Lisaks on LibrePdf üsna tulevikukindel, kui arvestada igasuguseid nõudeid, mida äri edaspidi soovida võib. Malli koodis hoidmine vastab ka eelnevalt püstitatud nõuetele, ehk mallid saab teha modulaarsed ning neid on mugav refaktoreerida. Seda kõike arvesse võttes sai otsustatud LibrePdf teegi kasuks.

4 Realisatsioon

Töö realiseerimisel sai arvestatud sellega, et loodav rakendus oleks võimalikult lihtsalt liidestatav väliste rakendustega. See võimaldab ettevõttel huvi korral loodud rakendust oma süsteemis lihtsalt katsetada ning selle põhjal otsustada, kas rakendust tahetakse oma süsteemi implementeerida.

Töö raames loodava rakenduse realiseerimiseks kasutatavate teekide valimisel lähtuti peamiselt kolmest punktist:

1. Kasutusse võetavad teegid peavad olema vabavaralise litsentsiga, mis lubaks loodava rakenduse MIT litsentsi all avaldada.
2. Kasutusse võetavad teegid peavad olema hästi dokumenteeritud, et rakendust oleks võimalik mugavalt ka teistel edasi arendada.
3. Kui valida on mitme sarnase teegi vahel, mis pakuvad vajalikku funktsionaalsust, siis valitakse nendest lihtsamini kasutatav.

4.1 Arhitektuur

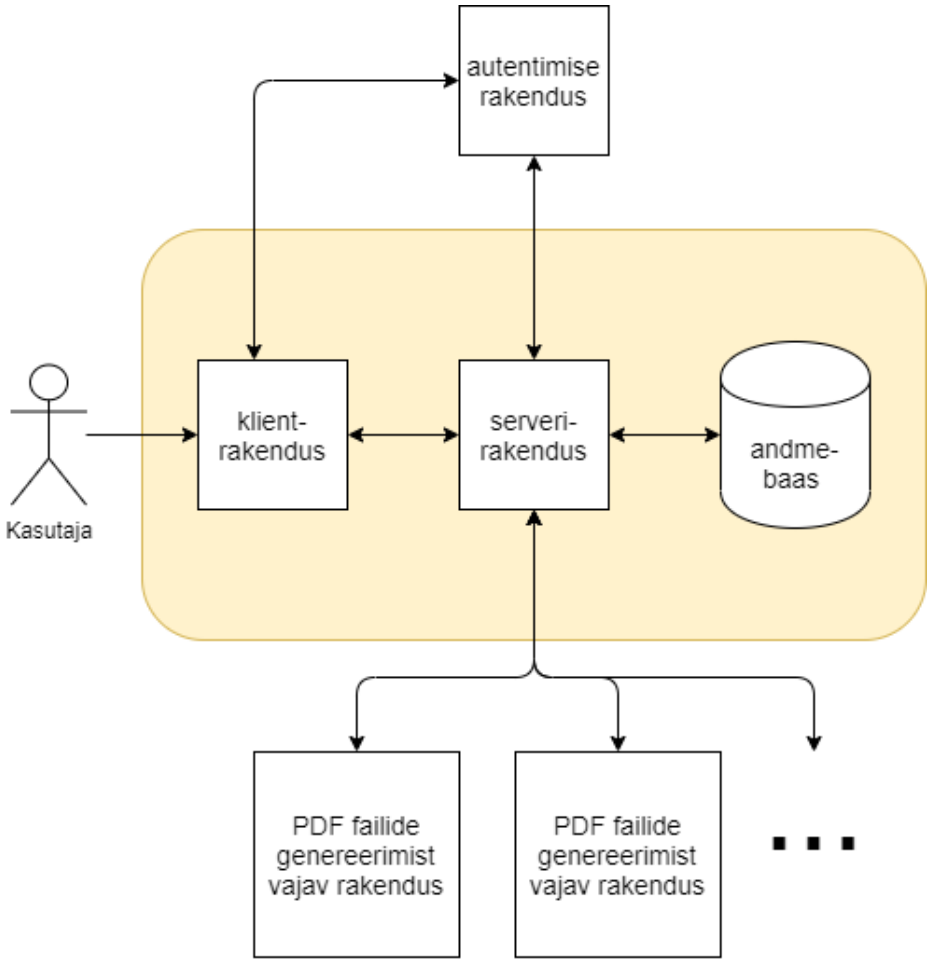
Töö käigus loodud süsteemi arhitektuuri paika panemisel sai lähtutud nõudest, et serverirakendus peab olema kasutatav mikroteenuste arhitektuuris ning olema klientrakendusest eraldiseisev. Loodud PDF failide genereerimise rakendus koosneb kolmest eraldiseisvast osast: klientrakendus, serverirakendus ja andmebaasisüsteem.

Autentimise rakendusena on töös kasutatud OAuth2 protokolli jälgivat Google sisselogimise teenust[1]. Failide genereerimist vajavaid rakendusi jäljendati töö käigus Postman[2] rakenduse abil.

Joonisel 4 on kuvatud rakenduse arhitektuur, millel on kollase taustaga märgitud käesoleva töö raames loodud osad.

[1] https://developers.google.com/identity/protocols/oauth2

[2] https://www.postman.com/



Joonis 4. Rakenduse arhitektuur.

4.2 Andmevahetus

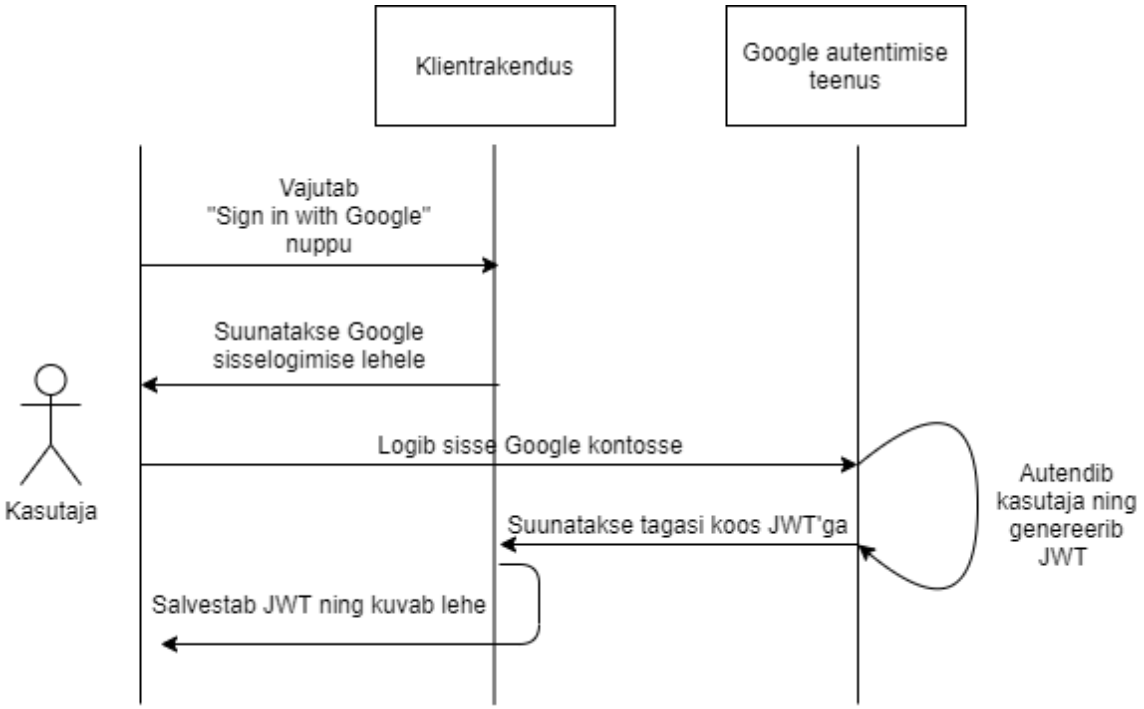
Serverirakendusega suhtlemiseks kasutatakse REST (*Representational State Transfer*) API liidest. Termini REST defineeris Roy Fielding oma doktoritöös 2000. aastal [5], kasutades seda, et kirjeldada hästi kavandatud veebirakenduse käitumist. REST seab veebirakenduse loomisele järgmised piirangud:

- Klientrakendus ning serverirakendus on üksteisest eraldatud.
- Server ei hoia sessioone ning olekuid, olekut hoitakse vajadusel kliendi poolel.
- Vastuste meelde jätmise võimalus, et vähendada vajaminevat kliendi ja serveri vaheliste päringute arvu.
- Kihilise süsteemi tugi, ehk kliendi ning serveri vahele teenuse lisamine ei tohi mõjutada kliendi ja serveri vahelist suhtlust.
- Ühtne liides: serveriga suhtlemiseks kasutatakse ühtset ressursi identifikaatorit (*URI*). [6]

REST printsiipide järgimine sobib hästi mikroteenuste arhitektuuris kasutamiseks, sest serveriga suhtlemiseks ühtsete ressursi identifikaatorite kasutamine võimaldab rakendust mugavalt liidestada ka teistes programmeerimiskeeltes kirjutatud rakendustega. [7] Samuti sobib mikroteenuste arhitektuuri REST printsiibi juurde käiv põhimõte, et server ei hoia olekuid ning sessioone, sest nii on võimalik luua teenusest mitu instantsi ning kõik instantsid on võimelised klienti teenindama. [8]

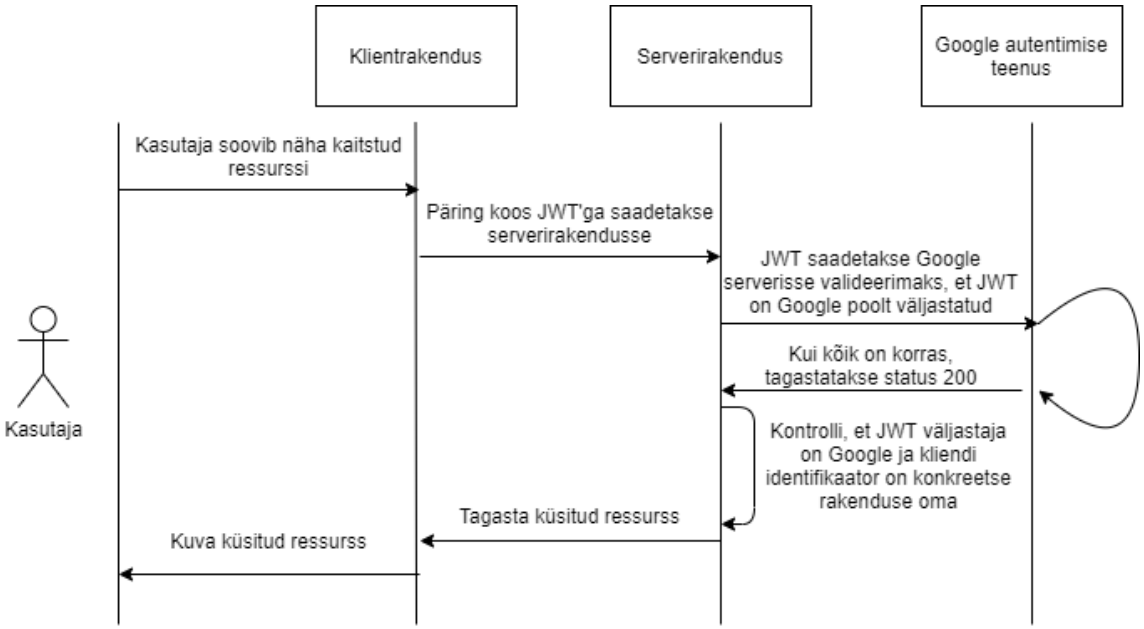
4.3 Autentimine ja rollid

Autentimiseks kasutatakse Google sisselogimise teenust, mis tagastab eduka autentimise korral JWT (*JSON Web Token*) loa klientrakendusele. Seda voogu kirjeldab joonis 5.



Joonis 5. Autentimine klientrakenduse kaudu.

Klientrakenduse kaudu serverirakenduse pihta päringut tehes antakse saadud luba serverile koos päringuga kaasa. Serverirakendus saadab JWT loa uuesti Google teenusele, mis kontrollib, et luba on tõesti Google poolt väljastatud ning kehtib ja seejärel kontrollib rakendus, et ka rakenduse identifikaator JWT loa sees on käesoleva rakenduse oma. Seda voogu kirjeldab joonis 6. [9]



Joonis 6. Serveripoolne JWT kontroll.

Kasutajanimena kasutatakse Google kontoga seotud emaili aadressi. Süsteemis on kasutusel kolm rolli: tavakasutaja, arendaja ja redigeerija. Rollide haldamiseks on andmebaasis tabel *application_roles*, kus on võimalik kasutajanime põhjal kasutajale rolle määrata. Kui tabelis kasutajat ei ole, määratakse talle tavakasutaja roll. Tavakasutaja roll lubab kasutajal kasutajaliidese kaudu malle kuvada ning API liidese kaudu trükiseid genereerida. Teksti– ja disainimuudatuste tegemiseks peab kasutajale olema antud redigeerija rolli õigused ning uute tekstiplokkide lisamiseks peavad olema kasutajale antud arendaja rolli õigused.

4.4 Klientrakendus

Klientrakendus on realiseeritud üheleherakendusena (*Single-Page Application*) ning leht renderdatakse kliendi poolel (*Client-Side Rendering*). Sellise kombinatsiooni peamiste puudustena tuuakse esile kehv otsingumootori optimiseerimine (*SEO*), aeganõudev esmakordne lehe kuvamise aeg ning väliste teekide vajadus. [10] Esimese kahe probleemi vastu on hakatud kasutama ka serveripoolset üheleherakenduste renderdamist, kuid selle töö raames sellist lahendust kasutusele ei võetud, sest loodav rakendus ei pea olema otsingumootorite poolt leitav ning serveripoolne lehe renderdamine kuvab küll lehe sisu kiiremini, kuid lehe kasutamiseks on vaja ikka oodata ära, kuniks kõik vajalik on brauseri poolt alla laetud. [11] Seetõttu sai otsustatud lihtsama, ehk kliendi pool rendertatava üheleherakendus kasuks.

4.4.1 Klientrakenduse raamistik

Klientrakenduse loomisel sai kasutatud Vue.js[1] raamistikku, et rakenduse haldamine ning edasiarendus võimalikult lihtsaks muuta. Vue.js on MIT litsentsi all avalikustatud kergekaaluline vabavaraline raamistik klientrakenduste loomiseks. Teegi loomisel on arvestatud sellega, et Vue.js teeki oleks võimalik projekti integreerida järk–järgult.

Vue.js teegi eesmärk on lihtsustada veebiarendust, pakkudest taaskasutatavaid komponente. Vue komponent koosneb üldjuhul kolmest osast: mall, skriptid ja stiilid. Kui mõnes komponendis skripte või stiile ei kasutata, siis võib need osad ka komponendi struktuurist välja jätta. [12]

Koodi lihtsamini loetavaks ning hallatavaks tegemise eesmärgil sai kasutatud JavaScript'i asemel TypeScript[2] keelt. TypeScript on loodud arendajate abistamiseks, lisades JavaScript'ile staatilised tüübid, moodulid, klassid ja liidesed. Seejuures ei ole nende kasutamine kohustuslik, ehk iga JavaScript'i programm on samal ajal ka TypeScript'i programm. [13]

TypeScript on viimastel aastatel arendajate sead kõvasti populaarsust kogunud, olles 2019 aasta Stack Overflow arendajate lemmikute programmeerimiskeelte küsimustikus esimese kolme seas. [14] Vue.js teegil on hea tugi TypeScript'i kasutamiseks ning oma eeliste ja populaarsuse tõttu sai rakenduse kirjutamisel kasutatud JavaScript'i asemel TypeScript'i.

4.4.2 Koodi ja stiili automaatne kontrollimine

Klientrakenduse koodi kirjutamise lihtsustamiseks on kasutusel tööriistad, mis aitavad tuvastada potentsiaalseid vigu ning koodi stiliseerida. Kuigi juba ainuüksi TypeScript'i õigesti kasutamine aitaks vigu vältida, siis on mõistlik kontrollida, et TypeScript võimalusi ka tegelikult õigesti kasutatakse. Selle juures on suureks abiks ESLint[3] nimeline programm.

[1] https://vuejs.org/

[2] https://www.typescriptlang.org/

[3] https://eslint.org/

ESLint analüüsib kirjutatud koodi, et leida potentsiaalseid vigu ning analüüsimise konfiguratsiooni on võimalik arendajal lihtsasti muuta. [15] Kui teha muudatusi ja koodi refaktoreerida, siis võib lihtsasti mõni märk kuskile kaduma minna või mõne teegi sisse laadimine (*import*) ununeda. ESLint aitab sellised vead leida juba arenduse faasis, mis kiirendab üldist arendusprotsessi.

Koodi formaadi osas vaieldakse tihti, sest inimesed on erinevad ja ka koodi kirjutades kujundab igaüks koodi vastavalt oma arvamustele. Seejuures kui igaüks kirjutab omas stiilis, siis lõpptulemust lugeda on kolmandatel isikutel väga keeruline. Mugavaks ja ühtlaseks koodi kujundamiseks on projektis kasutusel Prettier[1] rakendus koos Airbnb[2] stiilijuhistega. Prettier on rakendus koodi automaatseks kujundamiseks, mis ühildub paljude koodi kirjutamise rakendustega. Prettier konfiguratsiooni on lihtne muuta, selleks on vaja lisada eraldi JSON või JavaScript fail, mis sisaldab juhist, millest koodi kujundamisel lähtuda. [16]

4.4.3 Rakenduse kujundamine

Saadaval on palju teeke, mis pakuvad Vue raamistikule valmis komponente kasutajaliideste loomiseks. [17] Nendest mitmed oleks olnud sobivad sellesse projekti, kuid valik sai langetatud populaarsuse põhjal. Bootstrap[3] on nii laialdaselt levinud, et suur osa arendajaid peaks olema selle raamistikuga kokku puutunud ning selle haldamisega hakkama saama, mis teeb selle teegi väga atraktiivseks.

Töö käigus loodud rakendus kasutab klientrakenduse kujundamisel BootstrapVue[4] raamistiku komponente ning Bootstrap 4 raamistiku CSS stiilifaile. Vajadust CSS eelprotsessori järgi projektis ei tekkinud. Kui nõutud oleks olnud vanemate brauserite tugi, siis oleks ka Sass raamistiku kasutusele võetud, et edasist arendust mugavamaks teha, kuid nõutud oli vaid moodsamate laialt kasutuses olevate brauserite tugi. Seetõttu on võimalik kasutada CSS'i enda võimekust näiteks muutujate deklareerimisel, mida nüüdseks toetavad suur osa moodsaid brausereid. [18]

---

[1] https://prettier.io/

[2] https://github.com/airbnb/javascript

[3] https://getbootstrap.com/

[4] https://bootstrap-vue.org/

4.4.4 Ikoonid

Ikoonidena on kasutatud Material Design'i[1] ja ametlikult Google lehelt[2] alla laetud SVG faile, mida on võimalik vue-svg-loader[3] teegi abil kasutada kui Vue komponente. Selline lähenemine võimaldab kasutada ükskõik milliseid ise tehtud või alla laetud SVG ikoone.

Sisuedastusvõrgul (*CDN*) või valmislahendust pakkuvatel teekidel (näiteks Font Awesome[4]) on peamiseks puuduseks see, et ükski ei paku päris kõiki vajaminevaid ikoone. Font Awesome näiteks ei pakkunud ametlikku Google sisselogimise nupu ikooni. Lisaks kaasneb paljude tasuta teekidega ka nõudeid, osade puhul on näiteks veebilehel kuvada ikooni autori nime.

Mitmete erinevate lahenduste korraga kasutamine ikoonide laadimiseks ei tundunud mõistlik ning seega sai otsustatud selle lahenduse kasuks, mis lubaks igal pool koodis ikoone ühtemoodi käsitleda. Seetõttu hoitakse ikoone koodi juures ning võimaldatakse neid Vue komponentidena kasutada.

4.5 Serverirakendus

Analüüsi käigus sai PDF failide genereerimise teegiks valitud OpenPdf, mis tähendas, et serverirakenduse programmeerimiskeelena tuli valida Java[5]. Java jaoks on loodud palju häid tööriistu ning teeke, mis teevad võrgurakenduste arendamise Java keeles väga lihtsaks.

4.5.1 Serverirakenduse loomine Javas

Java on pika ajalooga väga võimekas programmeerimiskeel. Selle esimene versioon avalikustati juba 1995. aastal Sun Microsystems poolt. Oracle Corporation omandas Suni 2009. aastal ning peale seda on Java arendatud Oracle poolt.

---

[1] https://material.io/resources/icons/

[2] https://developers.google.com/identity/branding-guidelines

[3] https://www.npmjs.com/package/vue-svg-loader

[4] https://fontawesome.com/

[5] https://www.java.com/en/

Java lähtekood kompileeritakse baitkoodiks, mille jooksutamiseks kasutatakse Java virtuaalmasinat (JVM). Seetõttu on Java koodi käivitamiseks vaja Java käivituskeskkonda (JRE). [19] Arendustöö tegemiseks on vajalikud Java arenduse tööriistad (JDK), mis sisaldavad endas ka Java käivituskeskkonda. Arenduseks sai kasutatud esialgu Java 13 ning peale Java 14 välja tulemist sai ka rakendus üle viidud uuema versiooni peale.

Väliste teekide haldamiseks ning ehitamise ja testimise protsessi automatiseerimiseks kasutatakse rakenduses Gradle[1] tööriista. Projekti konfigureerimiseks kasutatakse *build.gradle* nimelist faili, kus skriptide kirjutamiseks kasutatakse Groovy[2] keelt. Gradle protsessid jooksutavad väikseid ülesandeid (*task*), millest igaüks peaks täitma ühte eesmärki ning millel võivad olla omavahelised sõltuvused. Gradle sisaldab eeldefineeritud ülesandeid, mida on võimalik vastavalt vajadustele kohandada. [20]

Serverirakenduse loomise lihtsustamiseks kasutati Spring Boot[3] raamistiku. Spring Boot on konfigureeritud versioon Spring raamistikust, et rakenduse tööle saamine võimalikult kiiresti ja mugavalt käiks. Seejuures on võimalik konfiguratsioone vastavalt oma vajadustele lihtsasti üle kirjutada.

4.5.2 Turvalisus

Kuigi autentimiseks kasutatakse välist teenust, on vaja tagada, et sisselogimata või korrektsete rollideta isik loodud rakendust kasutada ei saaks. Spring Security[4] teek muudab kasutaja õiguste ning rollide haldamise väga lihtsaks. Spring Security automaatkonfiguratsioon tuli osaliselt üle kirjutada, et rakendus vastaks nõuetele.

Üle kirjutatud konfiguratsioon ei kasuta sessioone, et tagada rakenduse vastavus REST põhimõtetele. Lisaks on pika pandud CORS konfiguratsioon, mis võimaldab brauserist teha päringuid vaid klientrakendusel. CSRF rünnakute vastast kaitset serveri pool ei rakendata, sest JWT hoiustamine toimub kliendi poolel ning seda hoitaks kohalikkus

[1] https://gradle.org/

[2] https://groovy-lang.org/

[3] https://spring.io/projects/spring-boot

[4] https://spring.io/projects/spring-security

hoidlas (*local storage*), mis hoiab ära võimaluse, et kasutaja CSRF rünnaku ohvriks langeb. [21]

4.5.3 Andmebaasi haldamine serverirakenduse kaudu

Serverirakendusest andmebaasi SQL lausete käivitamiseks kasutatakse MyBatis[1] raamistiku. Erinevalt objekt–relatsioonilise vastendamise (*ORM*) mudelist ei vastenda MyBatis Java objekte andmebaasi tabelitega, vaid Java meetodeid SQL lausetega. Selline lähenemine aitab andmebaasiga seotud loogikat ülejäänud koodist eraldada. MyBatis pakub ka palju lisavõimalusi, näiteks võimalust SQL lausete dünaamiliseks loomiseks. [22] MyBatise abiga dünaamilise SQL lause loomise näide on näha Lisas 1.

MyBatis võimaldab ka kindlate SQL lausete täitmist jälgida. See on töö raames loodud rakenduse jaoks väga oluline, sest seda võimalust kasutatakse kõikide malli muutmisega seotud SQL lausete faili kirjutamisel. See omakorda võimaldab kõik mallidega tehtud muudatused ühe failina alla laadida ning muudatused ka teise keskkonda üle kanda.

Andmebaasi skeemi jälgimiseks ja haldamiseks on kasutusel Liquibase[2] teek. Liquibase lubab vastavalt soovile kasutada kas XML, YAML, JSON või SQL skripte andmebaasis muudatuste tegemiseks. Defineeritakse üks peamine (*master*) fail, kus hoitakse jooksutatavate skriptide nimekiri ja järjekord. Liquibase'i jooksutamisel tehakse kindlaks, millised skriptid on juba jooksnud ning millised skriptid on uued. Uued skriptid jooksutatakse rakenduse käivitamise hetkel. Lisaks on võimalik teegi abil juba jooksutatud skriptide muudatusi tagasi võtta. [23]

4.5.4 Testimine

Loodud rakendus on kaetud ühik- ja integratsioonitestidega. Testimisel on kasutatud Mockito[3] ning JUnit[4] tööriistu, et jäljendada testitava objekti sõltuvusi. Integratsioonitestid ja ühiktestid ei kata loodud näitemallide osa, et lihtsustada ja kiirendada mallide muutmise või kustutamise protsessi.

---

[1] https://mybatis.org/mybatis-3/

[2] https://www.liquibase.org/

[3] https://site.mockito.org/

[4] https://junit.org/

Integratsioonitestide jooksutamiseks kasutatakse Testcontainers[1] teeki, mis paneb testimise ajaks tööle eraldi Dockeri[2] konteineri, kus jooksutatakse enne testimist ka kõik Liquibase abil defineeritud andmebaasi muudatused. See garanteerib, et testimise käigus kasutatakse võimalikult sarnast konfiguratsiooni produktsioonikeskkonnale ning ühtlasi testib ka Liquibase'i lisatud skriptide toimimist vastu korrektset andmebaasi. Seejuures ei mõjuta testid päris andmebaasi, seega on välistatud oht, et mõni test võiks päris andmeid rikkuda.

Kiiremaks integratsioonitestide jooksutamiseks oleks võimalik kasutada ka mõnda mälupõhist andmebaasi, nagu H2[3]. See ei garanteeriks aga andmebaasimuudatuste toimimist produktsioonikeskkonnas. Näiteks kui luua SQL uuendamise lause (u*pdate statement*), kus kasutatakse aliast lause *set* osa järel, siis selline skript toimiks H2 andmebaasis, kuid ei toimiks PostgreSQL andmebaasis. [24]

Seejuures iga testi alguses uue Docker'i konteineri loomine ja andmebaasimuudatuste jooksutamine muudaks testimise väga aeglaseks. Seetõttu on rakenduses kasutatud lahendust, mis loob ühe konteineri kõikide integratsioonitestide jaoks. See tähendab, et integratsioonitestide kirjutaja peab arvestama sellega, et tema testi järel kõik testi poolt lisatud andmed saaks kustutatud ning kõik testi poolt kustutatud andmed saaks uuesti lisatud. Siis on garanteeritud ka see, et iga testi alguses on andmed samasugused ja testide järjekorra muutmine ei riku testide loogikat.

4.6 Andmebaas

Andmebaasisüsteemi valimisel lähtuti sellest, et andmebaasisüsteem oleks tasuta saadaval ja toetatud MyBatis, Liquibase ning Testcontainers teekide poolt. Samuti oli oluline, et andmebaasisüsteem võimaldaks määrata piiranguid (*constraint*) ning käitumismustrit kitsendustega konflikti sattumisel. Otsustatud sai PostgreSQL 12 kasuks, mis vastab suures osas SQL 2016 standardile ning vastab püstitatud nõuetele. PostgreSQL suurteks tugevusteks on lisaks ka hea dokumentatsioon ja skaleeritavus,

[1] https://www.testcontainers.org/

[2] https://www.docker.com/

[3] https://www.h2database.com/html/main.html

mida kinnitab fakt, et PostgreSQL klastreid kasutatakse ka suurtes süsteemides, mis haldavad terabaitidesse ulatuvaid andmekoguseid. [25]

4.7 Docker

Docker on 2013 aastal loodud programm, mis võimaldab operatsioonisüsteemi tasemel virtualiseerimist. [26] See on mõeldud arendajatele, et teha tarkvara arendamise ja serverisse paigutamise protsess võimalikult mugavaks, ilma, et oleks vaja muretseda, kuidas tarkvara erinevatel platvormidel käitub. [27] Dockeri konteinerid sobivad loomu poolest väga hästi mikroteenuste arhitektuuri, tuues kaasa viis suurt tugevust:

1. Automatiseerimise kiirendamine igas tarkvara arendamise elutsüklis. Iga konteinerit on võimalik eraldi serveris jooksutada ja konteinerite serverisse üles panemiseks on saadaval mitmeid tööriistu. Kogu see protsess on tehtav skriptide abiga.
2. Iseseisvuse suurendamine erinevate teenuste vahel. Iga konteiner on eraldiseisev üksus, mis võimaldab süsteemi eri osasid arendada erinevates programmeerimiskeeltes ja erinevate tehnoloogiate abiga.
3. Porditavuse suurendamine. Docker pakib konteinerisse kaasa kõik vajalikud sõltuvused, mis on saadaval igas keskkonnas kus konteinerit käivitada soovitakse.
4. Vähene ressursside kasutus. Docker'i konteinerid sisaldavad endas ideaaljuhul vaid rakendust ning rakenduse sõltuvusi.
5. Turvalisuse suurendamine. Sõltuvuse vähendamine süsteemi eri osade vahel võimaldab mugavamalt testida iga osa turvalisust. [28]

PDF'ide genereerimise rakenduses on iga süsteemi osa jaoks loodud Docker'i konteiner. Klientrakenduse ja serverirakenduse kokku panemise protsesside kirjeldamiseks on loodud *Dockerfile* skriptid ning kogu rakenduse tööle panemise mugandamise eesmärgil on loodud ka *docker-compose.yml* skript, mis automaatselt kõik kolm rakenduse osa Docker'i konteineritena käivitab. Rakenduses kasutusel olev *docker-compose.yml* skript on näha töö Lisas 2.

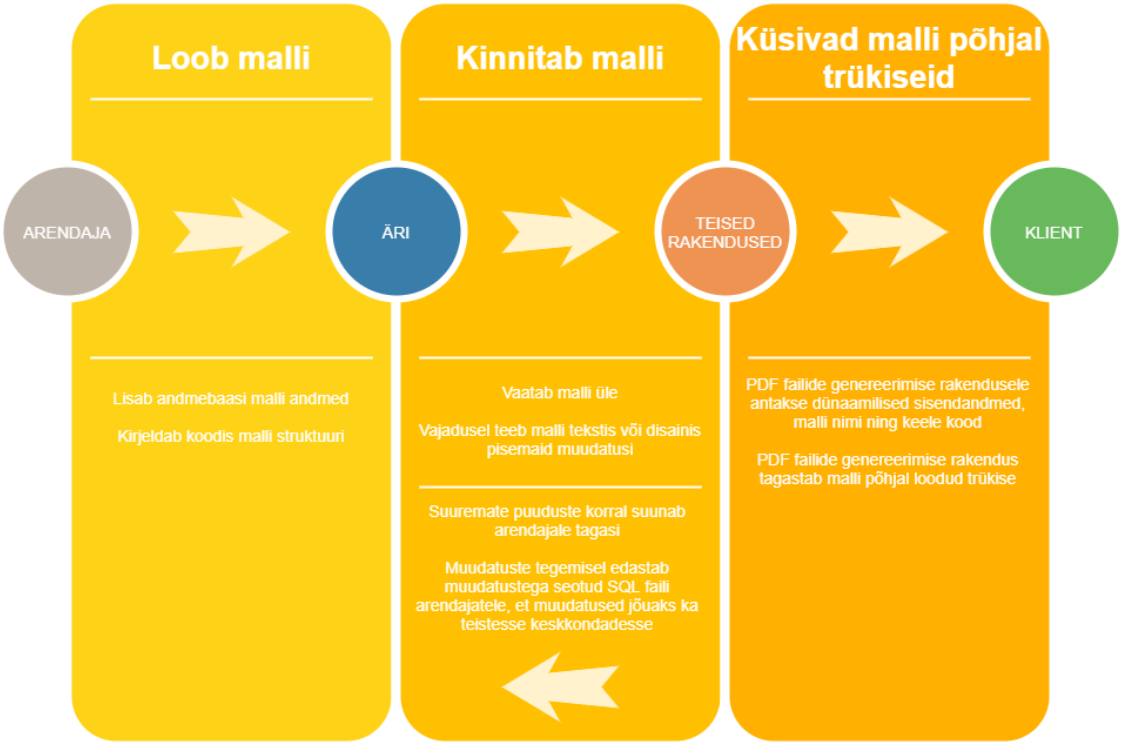
5 Tulemus

Töö tulemuseks on PDF failide genereerimise rakendus koos nelja näitemalliga. Rakenduse käivitamiseks on vaja Docker'it, Java 14 ning Gradle 6. Nende olemasolul on võimalik rakendus kahe lihtsa käsuga käivitada. Rakendus on vabalt alla laetav ja kõigile saadaval GitHub keskkonnas[1].

5.1 Rakenduse kasutamine

Rakenduse kasutamine algab sellest, et arendaja saab info malli andmete kohta ning loob esialgse versiooni mallist. Seejärel saab äri pool malli üle vaadata. Äri poolelt kinnituse saamise korral võetakse mall kasutusse ning seda on võimalik teistel rakendustel kasutada trükiste loomiseks. Kui rakendus vajab suuremaid struktuurseid muudatusi, tuleb ülesanne suunata arendajale. Kui aga vajatakse väiksemaid muudatusi, on võimalik äri poolel need muudatused kasutajaliidese kaudu teha ning saadud SQL skript arendajatele edastada. Nii jääb arendajate ülesandeks vaid saadud SQL faili Liquibase'i migratsiooni lisamine. Malli loomise protsessi kirjeldab joonis 7.

[1] https://github.com/katamb/pdf-generator



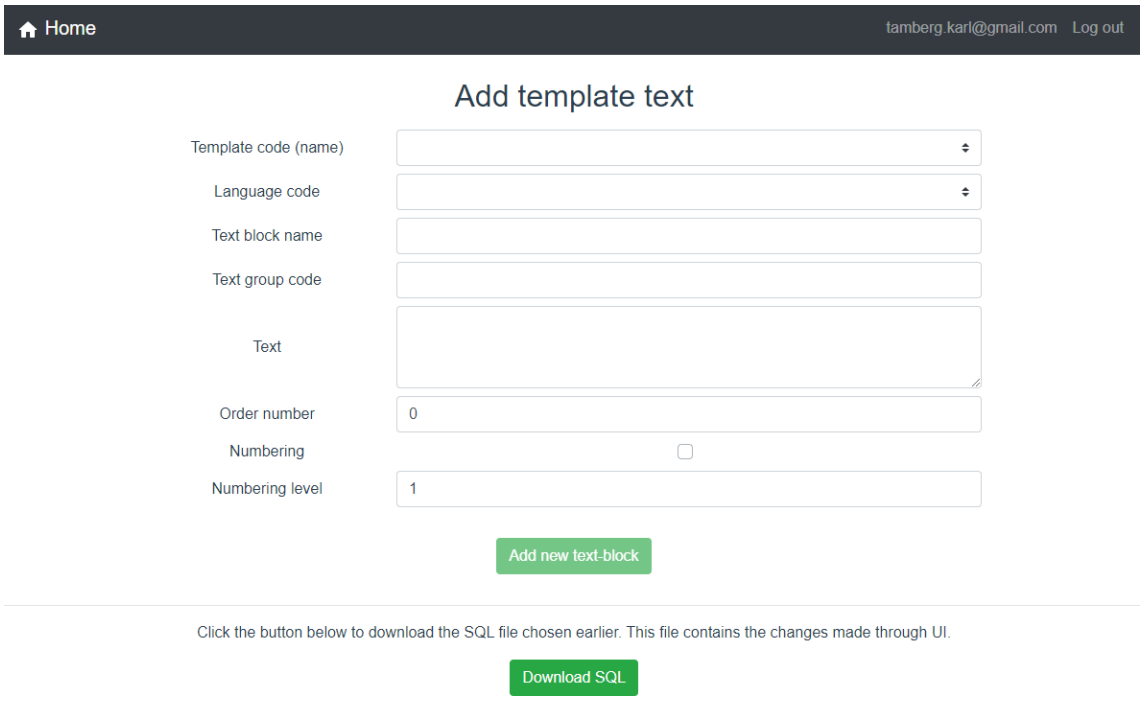
Joonis 7. Malli loomise ning kasutamise protsess.

Kasutajaliides võimaldab mallide juures teha muudatusi vaid andmebaasis hoitavate andmetega. Need muudatused peavad nõuete kohaselt olema lihtsasti üle kantavad ka teistesse keskkondadesse. Selleks tuleb kasutajal enne muudatuste tegemist valida või luua fail, kuhu andmebaasi pihta tehtavad mallide muutmisega seotud andmebaasi päringud salvestatakse. Peale muudatuste tegemist tuleb alla laadida saadud SQL fail ning lisada saadud fail Liquibase skriptide juurde. Seejärel saab kasutada versioonihaldustarkvara võimekust, et keskkondasid hallata.

Rakenduse sisselogimise leht on näha Lisas 3 ning rakenduse avaleht on näha Lisas 4.

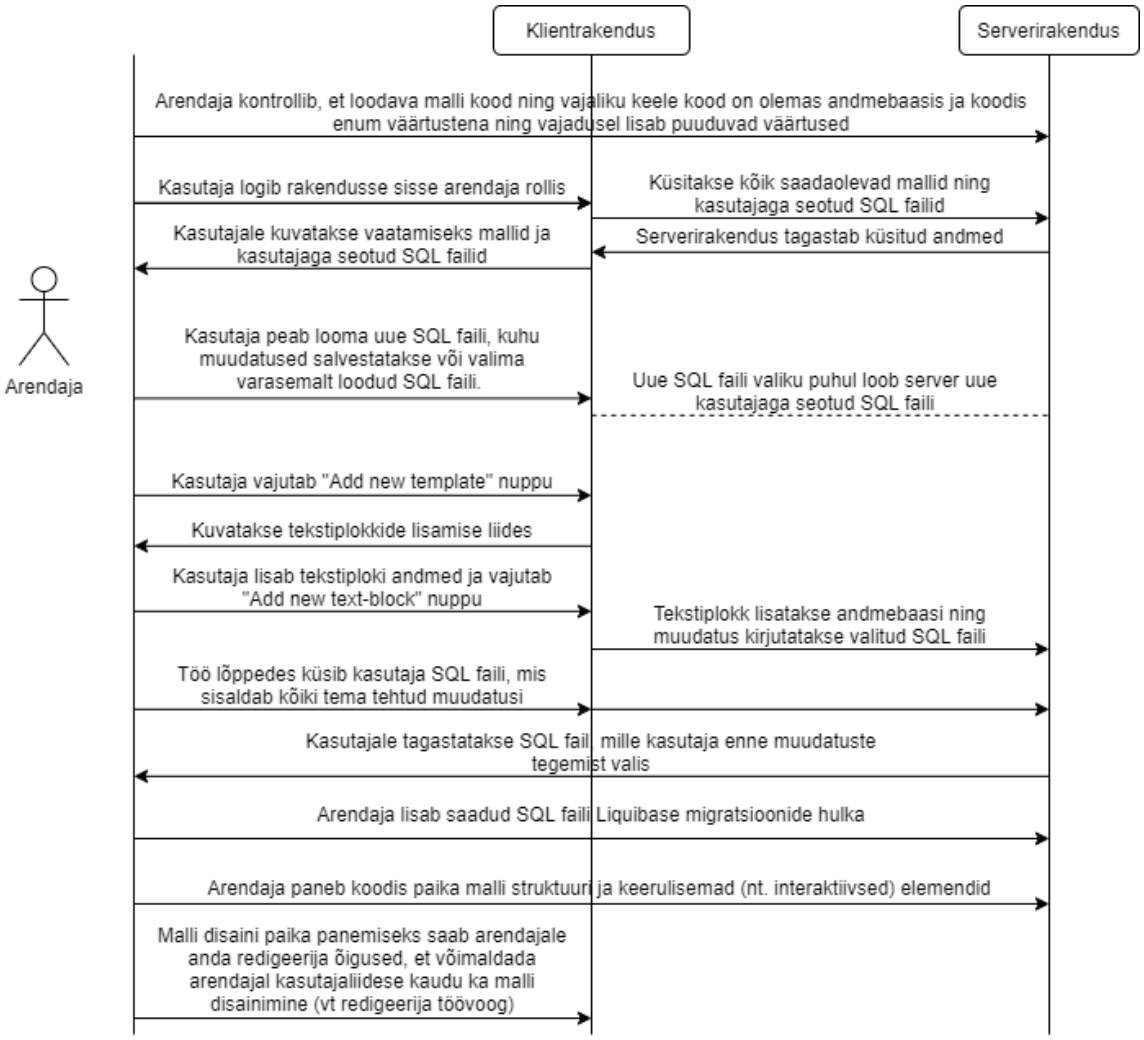
5.1.1 Arendaja töövoog

Malli loomiseks on vaja lisada uue malli kood (nimi) Liquibase skriptide abil andmebaasi ning konstant (*enum*) väärtusena koodi. Seejärel saab arendaja mugavalt kasutajaliidese kaudu lisada andmebaasi malliga seotud tekstiplokid. Töö lõpul saab tekstiplokkide lisamiseks vajalikud SQL laused alla laadida ning selle faili samuti Liquibase skriptidesse lisada. Arendajale mõeldud kasutajaliides on näha joonisel 8.



Joonis 8. Arendaja kasutajaliides.

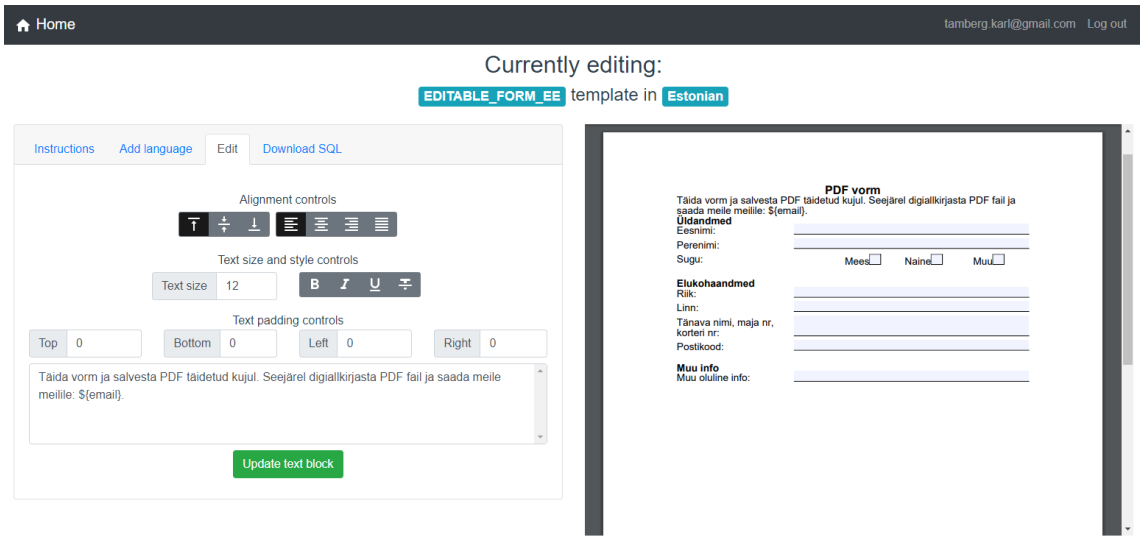
Kõik arendaja töö juures vajalikud sammud on võimalik läbi teha ka kasutajaliidest kasutamata, kuid ilma kasutajaliideseta on malli loomise protsess aeglane ning ebamugav. Peale tekstiplokkide lisamist tuleb koodis luua *PdfGenerator* klassi laiendav klass, mis kirjeldab malli struktuuri ning keerulisemaid (näiteks interaktiivseid) elemente. Seejärel on mõistlik avada kasutajaliides ning kasutada redigeerija rollile mõeldud võimalusi malli lõpliku disaini paika panemiseks. Arendaja töövoogu kirjeldab joonis 9.



Joonis 9. Arendaja töövoog.

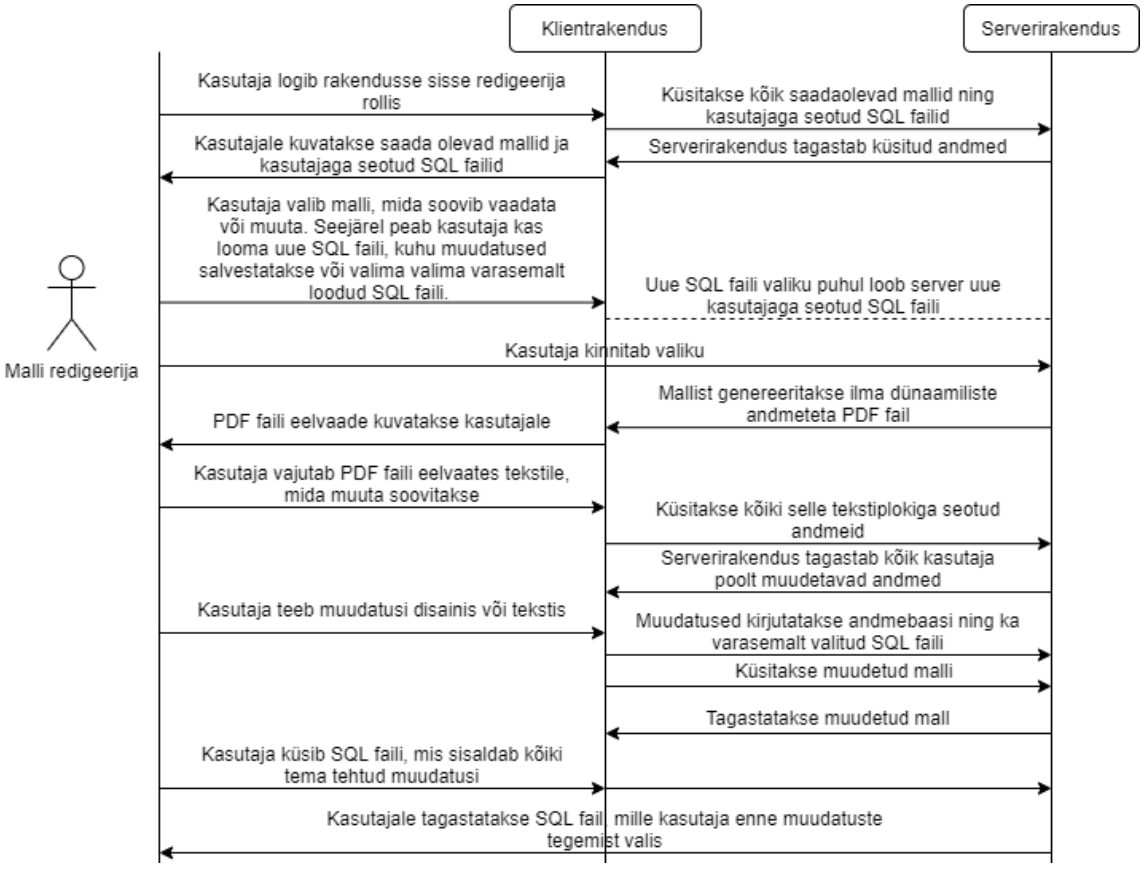
5.1.2 Redigeerija töövoog

Redigeerija logib rakendusse sisse, valib malli, keele ning SQL faili. Seejärel kuvatakse redigeerija vaade, mille vasakul küljel on malli muutmiseks vajalikud tööriistad ning paremal küljel malli eelvaade. Vajutades malli eelvaates tekstiplokile, kuvatakse vasakul selle tekstiplokiga seotud andmed ning lubatakse neid andmeid muuta. Joonisel 10 on näha redigeerija kasutajaliides.



Joonis 10. Redigeerija kasutajaliides.

Kasutajaliides võimaldab redigeerijal lisaks malli muutmisele ka avatud malli põhjal muukeelse malli loomist. Selleks tehakse koopia käesolevast mallist ning seejärel tuleb käsitsi kõikide tekstiplokkide väärtused ära tõlkida. Tekstiplokkide tõlkimiseks on töövoog sama nagu ka muude teksti muudatuste tegemiseks. Joonisel 11 on kuvatud redigeerija töövoog teksti või disaini muutmiseks.

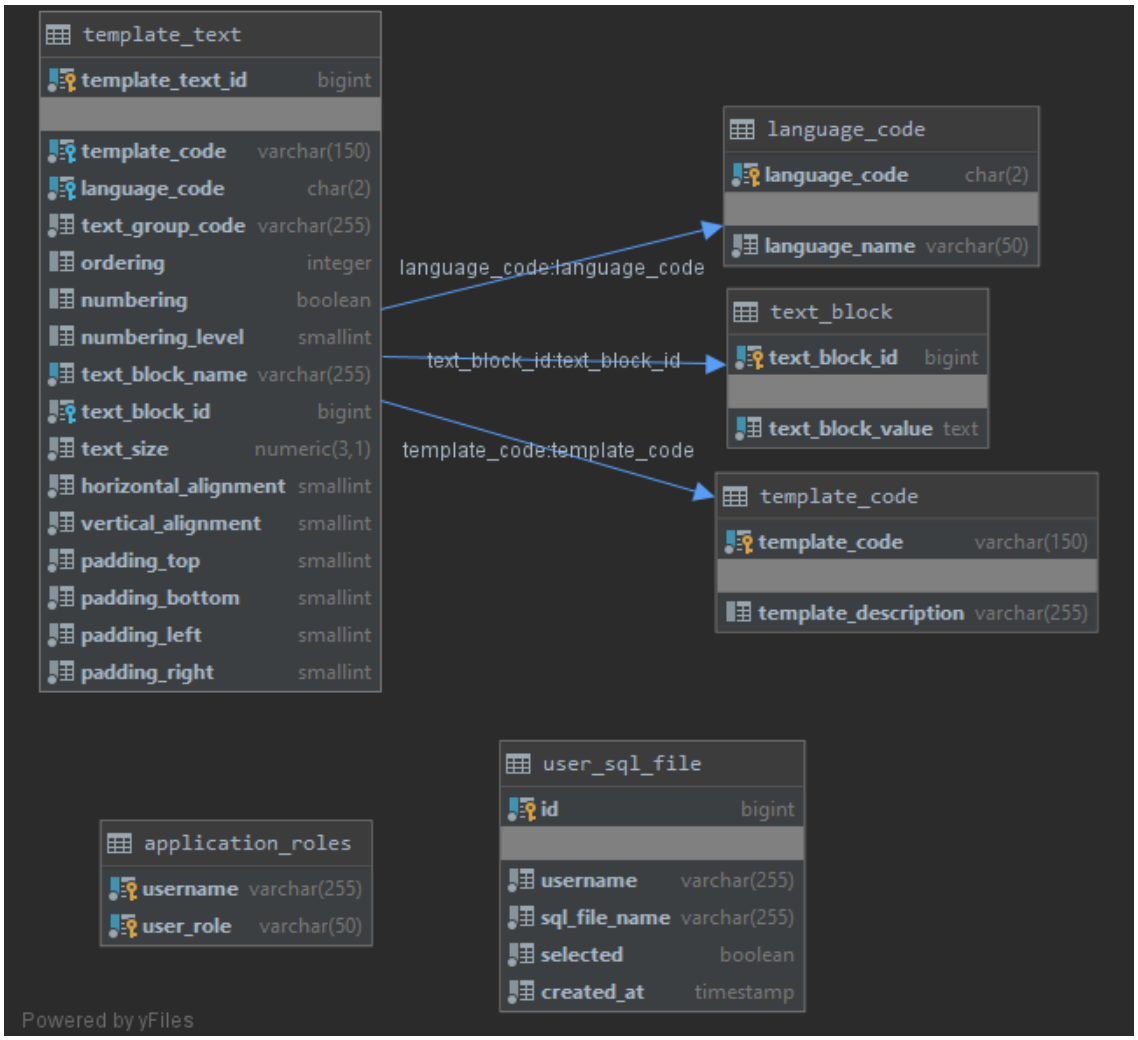


Joonis 11. Malli redigeerimine kasutajaliidese kaudu.

5.2 Andmebaas

Kõiki kasutajaliidese kaudu muudetavaid malliga seotud andmeid hoitakse andmebaasis. Neid malli osasid kirjeldab *template_text* tabel. Malli nime ning keelt hoitakse eraldi klassifikaatorite tabelis. Staatilise teksti jaoks on samuti eraldi tabel, sest osa tekste võivad olla väga pikad ning korduda. Selline lähenemine eemaldab vajaduse korduvaid tekste mitu korda andmebaasi lisada, hoides kokku salvestusruumi ning parandades töökiirust.

Automaatselt genereeritud SQL failide metaandmete haldamiseks on kasutusel *user_sql_file* tabel. Nendesse SQL failidesse kirjutatakse kasutajaliidese kaudu mallis tehtud muudatused ning failid salvestatakse konfiguratsioonifailis (*application.properties*) märgitud kataloogi. Rollide haldamiseks kasutatakse *application_roles* tabelit. Loodud andmebaasi skeem on näha joonisel 12.



Joonis 12. Rakenduse andmebaasi skeem.

Andmebaasi haldamise teeb mugavaks MyBatis ning Liquibase teekide koos kasutamine. MyBatis pakub päringute pealt kuulamiseks liidest (*interceptor*), mis lubab mugavalt tehtavaid andmebaasipäringuid filtreerida ning olulised muudatused faili kirjutada. Liquibase omakorda teeb mugavaks loodud SQL failide haldamise.

5.3 Muudatused rakenduse päriselt kasutusse võtmisel

Saadud rakendus on küll koheselt kasutatav, kuid rakenduse päris süsteemi integreerimisel tuleks teha mõningaid muudatusi. Suur osa vajalikest muudatustest tuleb teha serverirakenduse konfiguratsioonifailis (*application.properties*) ning klientrakenduse konstantide failis (*constants.ts*).

- Kui ei kasutata Google autentimise teenust, siis tuleks autentimise osa oma autentimise teenuse peale ümber teha. Arendamisel sai arvestatud, et see võib vajalik olla, ning autentimise osa sai jäetud võimalikult lihtsalt muudetavaks.
- Kui jäädakse Google autentimist kasutama, siis tuleks:
    - Google Cloud[1] platvormil luua oma rakenduse jaoks uus OAuth 2.0 Client ID.
    - Panna Google Cloud platvormil paika õiged Authorized JavaScript Origin väärtused.
    - Muuta konfiguratsioonifailis ja konstantide failis olev kliendi identifikaatori (*clientId*) väli eelmises punktis saadud kliendi identifikaatori vastu.
    - Hetkel kasutatakse JWT kontrollimiseks Google enda teenust, mis tähendab ühte lisapäringut. Kiiruse tõstmiseks võib automaatselt laadida Google avalikud võtmed serverisse ning JWT kontrolli ilma lisapäringuta serveris teha.[2]
- Docker'i konteiner, kuhu pannakse püsti klientrakendus, kasutab väga lihtsat serverit. Kui rakendust kasutada avalikus veebis, oleks mõistlik kasutada mõnda paremini konfigureeritavat serverit.[3]

[1] https://console.cloud.google.com/apis/credentials

[2] https://developers.google.com/identity/sign-in/web/backend-auth

[3] https://vuejs.org/v2/cookbook/dockerize-vuejs-app.html

- Konfiguratioonifailis tuleks muuta andmebaasi salasõna.
- Muuta konfiguratsioonifailis klientrakenduse URL ning konstantide failis serverirakenduse URL.

5.4 Edasiarenduse võimalusi

Loodud rakenduse edasiarendusena saaks lisada kasutajaliidesesse võimalusi malli disainimuudatuste tegemiseks. Kuna iga tekstilõik on oma kastikese sees, siis selle kastikese taustavärvi, piirjooni ja ka näiteks fonti võiks saada muuta läbi kasutajaliidese. Samuti võiks kasutajaliides lubada lehe ääriste suuruse muutmist. Nii jääks lõpuks koodi ainult teksti paigutuse ning loogika kirjeldus, mis kiirendaks mallide loomise ja muutmise protsessi.

Mitmekeelsete dokumentide lihtsamaks loomiseks võib kasutusele võtta näiteks Google Translation API[1] teenuse. See võimaldaks suure osa tõlketööst automatiseerida. Seejuures tuleb arvestada, et oluliste lepingute puhul tuleks tõlgitud tekst kindlasti üle vaadata, sest automaatselt tõlgitud tekst võib olla ebausaldusväärne.

Töö käigus loodud rakenduse põhjal võiks kirjutada ka uue rakenduse, mis hoiaks kogu malli infot andmebaasis ning võimaldaks kasutajaliidese kaudu kogu malli loomise protsessi ära teha. Selline lahendus võistleks paljude tasuliste lahendustega, kuid kaotaks kindlasti osa võimalusi, mida võimaldab malli koodis hoidmine.

Nii oleks potentsiaalsel rakenduse kasutajal valik. Esimese lahenduse puhul on vaja panustada rohkem aega malli loomiseks, kuid mallide disainile piiranguid ei seata. Teine lahendus pakuks kiiremat ja mugavamat mallide loomist, kuid seaks loodavatele mallidele piiranguid.

[1] https://cloud.google.com/translate

6 Valideerimine

Saadud tulemuste valideerimiseks hinnati loodud rakenduse vastavust analüüsi käigus püstitatud nõuetele. Töö juurde sai loodud neli malli näidetena, mis võimaldavad rakenduse käivitamise järgselt kohe rakendust testida. Mallide näidiste olemasolu lihtsustab ka uute mallide loomist.

6.1 Koodi valideerimine

Koodi toimimise kindlustamiseks on suur osa koodist kaetud ühik– ning integratsioonitestidega. Mallide koodile pole teste lisatud, sest mallide muutmine peab olema mugav ning kiire ja sellele osale testide lisamine teeks mallide muutmise aeganõudvamaks. Seejuures mallide koodis hoidmise üks eelis on justnimelt see, et kui mõni mall sisaldab endas tõeliselt keerulist loogikat, siis ka sellele on võimalik lihtsasti teste lisada.

6.2 Funktsionaalsetele nõuetele vastavus

Kõik kasutajaliidesele püstitatud nõuded said täidetud. Kasutaja näeb rakendusse sisse logides nimekirja kõikidest mallidest ning malli valiku tegemisel näeb kõiki keeli, milles antud mall saadaval on. Valides ka keele, näeb kasutaja malli sisu ilma dünaamiliste andmeteta. Redigeerija rollis saab kasutaja teha malli tekstis ning disainis kasutajaliidese kaudu muudatusi ning arendaja rollis lisada tekstiplokke. Olemasoleva malli põhjal muukeelse malli loomine on samuti tehtav kasutajaliidese kaudu.

Ka serverirakendusele esitatud nõuded said suures osas täidetud. Mallide modulaarsuse, mugava refaktoreeritavuse ja interaktiivsete failide loomise võimaluse nõuded said täidetud PDF failide genereerimise lahenduse valikuga. Loodud serverirakendus on klientrakendusest eraldiseisev ja sobib kasutamiseks mikroteenuste arhitektuuris.

Erinevate keskkondade toe tagamiseks hoitakse kõiki kasutajaliidese kaudu muudetavaid parameetreid andmebaasis. MyBatis teegi abil kirjutatakse kõik mallis tehtud muudatused SQL faili, mille kasutaja saab peale muudatuste tegemist alla laadida

ning andmebaasi migratsioonidele lisada. Malli muudatuste osa loomisel on kasutatud kaitsvat programmeerimisstiili, et tagada migratsioonide jooksmine erinevates keskkondades. Näiteks ei kasutata selles osas automaatselt andmebaasi poolt genereeritud identifikaatoreid, mis võivad keskkondade vahel erineda.

Kõikvõimalikud disainivariandid peaksid samuti olema toetatud. Näiteks laenulepingu näitemallil on kasutatud lahendust, mis kuvab vastavalt laenugraafiku aastate arvule kahes tulbas järjest iga aasta kohta graafiku. Testitud tasuta ning tasulised lahendused, mis võimaldavad graafilise liidese kaudu malle luua, sellist lahendust mõistlikult teostada ei võimaldanud.

Mallide kiire ja mugava loomise nõue täielikult täidetud ei saanud. Arendaja, kes on rakendusega tuttav, peaks olema küll suuteline uue lihtsa malli looma kiiremini kui ühe tööpäevaga, kuid keerulisemate disainielementide puhul võtab kindlasti malli loomine ka kogenud arendajal rohkem kui ühe tööpäeva.

6.3 Mittefunktsionaalsetele nõutele vastavus

Arendustöös pole kasutatud ühtegi tugeva *Copyleft* litsentsiga teeki, mis võimaldab antud töö avalikustada MIT litsentsiga. Tööd on testitud 01.05.2020 seisuga värskeimates Google Chrome, Mozilla Firefox ning Microsoft Edge brauserites Windowsi operatsioonisüsteemil. Nendes tingimustes testimisel ei tuvastatud ühtegi probleemi.

Dokumendi genereerimise kiiruse mõõtmiseks sai mõõdetud aega, mis kulub laenulepingu näitemallist trükise genereerimiseks koos sisendandmetega. Keskmiselt kulus sellele umbes 50 ms, mis on poole kiirem, kui nõuetes kirjeldatud maksimaalne lubatud aeg.

Seega vastab rakendus kõikidele püstitatud mittefunktsionaalsetele nõuetele.

7 Kokkuvõte

Töö eesmärgiks oli luua PDF failide genereerimise rakendus, mis võimaldaks mallide kirjeldamise järgselt teha lihtsamaid muudatusi mallis kasutajaliidese kaudu. Trükiste genereerimise teenus pidi olema eraldisesisvalt ja lihtsasti kasutatav.

Eesmärgi täitmiseks sai analüüsitud nõudeid, millele loodav tarkvara vastama peaks ning valiti välja PDF failide loomiseks mõeldud teek, mis kõige paremini soovitud tulemust saavutada võimaldaks. Kõikide rakenduse loomise juures kasutatud tehnoloogiate puhul sai valik langetatud tööks kõige sobilikuma teegi kasuks, mis ei läheks töö nõuetega vastuollu.

Töö tulemusena valmis andmebaasist, serverirakendusest ning klientrakendusest koosnev mikroteenusena kasutatav PDF failide genereerimiseks mõeldud rakendus. Valminud rakendust on põhjalikult testitud. Rakenduse valideerimisel tehti kindlaks, et rakendus vastaks kõikidele püstitatud nõuetele. Valideerimisel selgus, et kuigi tehniliselt vastab rakendus nõuetele, siis uute mallide loomise protsess võiks olla lihtsam.

Loodud rakenduse edasiarendusena oleks võimalik luua lahendus, mis hoiaks kogu malli andmebaasis ning võimaldaks kogu mallide loomise ning muutmise loogikat kirjeldada kasutajaliidese kaudu. Sellise lähenemise suureks eeliseks oleks mallide loomise töö lihtsustamine, kuid sellega kaoks osa võimalustest, mida pakub malli koodis hoidmine.

Kasutatud kirjandus

[1] I. S. „Jaspersoft Studio Review,“ [Võrgumaterjal]. Available: http://www.innoventsolutions.com/jasper-review.html. [Kasutatud 18 05 2020].

[2] J. Salter, “Open source licenses: What, which, and why,” 24 02 2020. [Online]. Available: https://arstechnica.com/gadgets/2020/02/how-to-choose-an-open-source-license/. [Accessed 24 04 2020].

[3] M. A. Mayr, „Potential pitfalls when using Dompdf and how to fix them,“ 09 09 2017. [Võrgumaterjal]. Available: https://www.agoradesign.at/blog/potential-pitfalls-when-using-dompdf-and-how-fix-them. [Kasutatud 18 05 2020].

[4] B. Lowagie, „What is the difference between iText, JasperReports and Adobe LC?,“ 21 03 2013. [Võrgumaterjal]. Available: https://stackoverflow.com/questions/15492738/what-is-the-difference-between-itext-jasperreports-and-adobe-lc. [Kasutatud 12 04 2020].

[5] R. Fielding, „Architectural Styles and the Design of Network-based Software Architectures,“ 2000. [Võrgumaterjal]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/abstract.htm. [Kasutatud 15 05 2020].

[6] „Wikipedia,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Kasutatud 24 04 2020].

[7] E. Huang, „RESTful API vs Microservice,“ 05 09 2016. [Võrgumaterjal]. Available: https://medium.com/@ericjwhuang/restful-api-vs-microservice-eea903ac3e73. [Kasutatud 02 05 2020].

[8] M. Rouse, C. Bedell, E. Hannan ja S. Wilson, „RESTful API (REST API),“ 04 2020. [Võrgumaterjal]. Available: https://searchapparchitecture.techtarget.com/definition/RESTful-API. [Kasutatud 29 04 2020].

[9] Google, „Google Guides,“ 07 05 2020. [Võrgumaterjal]. Available: https://developers.google.com/identity/protocols/oauth2. [Kasutatud 07 05 2020].

[10] J. Vega, „Client-side vs. server-side rendering: why it’s not all black and white,“ 28 02 2017. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d/. [Kasutatud 24 04 2020].

[11] Chandu, „Are Single Page Applications still relevant in 2020?,“ 08 08 2019. [Võrgumaterjal]. Available: https://medium.com/commutatus/seo-in-single-page-applications-csr-vs-ssr-e342d7cc69b. [Kasutatud 24 04 2020].

[12] Vue, „Introduction,“ [Võrgumaterjal]. Available: https://vuejs.org/v2/guide/. [Kasutatud 07 05 2020].

[13] M. A. M. T. Gavin Bierman, „Understanding TypeScript,“ 2014. [Võrgumaterjal]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-662-44202-9.pdf. [Kasutatud 07 05 2020].

[14] J. U. L. —. DHTMLX, „JavaScript Trends in 2020,“ 23 12 2019. [Võrgumaterjal]. Available: https://codeburst.io/javascript-trends-in-2020-b194bebc5ef8. [Kasutatud 07 05 2020].

[15] J. Zhen, „Integrating Prettier + ESLint + Airbnb Style Guide in VSCode,“ 20 06 2018. [Võrgumaterjal]. Available: https://blog.echobind.com/integrating-prettier-eslint-airbnb-style-guide-in-vscode-47f07b5d7d6a. [Kasutatud 07 05 2020].

[16] „Why Prettier?,“ [Võrgumaterjal]. Available: https://prettier.io/docs/en/why-prettier.html. [Kasutatud 07 05 2020].

[17] S. Daityari, „14 of the Most Interesting Vue UI Component Libraries for 2020,“ 15 04 2020. [Võrgumaterjal]. Available: https://www.codeinwp.com/blog/vue-ui-component-libraries/. [Kasutatud 28 04 2020].

[18] C. Coyier, „What is the difference between CSS variables and preprocessor variables?,“ 25 10 2016. [Võrgumaterjal]. Available: https://css-tricks.com/difference-between-types-of-css-variables/. [Kasutatud 07 05 2020].

[19] Wikipedia, „Java,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Java. [Kasutatud 26 04 2020].

[20] A. MacMurray, „A beginners guide to Gradle,“ 07 06 2018. [Võrgumaterjal]. Available: https://medium.com/@andrewMacmurray/a-beginners-guide-to-gradle-26212ddcafa8. [Kasutatud 05 05 2020].

[21] R. Anderson, F. Hasan ja S. Smith, „Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core,“ 12 05 2019. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-3.1. [Kasutatud 09 05 2020].

[22] Wikipedia, „MyBatis,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/MyBatis. [Kasutatud 07 05 2020].

[23] S. Schmeltzer, „Introduction to Liquibase and Managing Your Database Source Code,“ 23 10 2017. [Võrgumaterjal]. Available: https://dzone.com/articles/introduction-to-liquibase-and-managing-your-databa. [Kasutatud 02 05 2020].

[24] J. Frankowski, „DB Integration Tests with Spring Boot and Testcontainers,“ 18 11 2019. [Võrgumaterjal]. Available: https://www.baeldung.com/spring-boot-testcontainers-integration-test. [Kasutatud 07 05 2020].

[25] PostgreSQL, „About,“ [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 26 04 2020].

[26] Wikipedia, „Docker,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Docker. [Kasutatud 22 04 2020].

[27] „What is Docker?,“ [Võrgumaterjal]. Available: https://opensource.com/resources/what-docker. [Kasutatud 18 04 2020].

[28] D. Jaramillo, R. Smart ja D. V. Nguyen, „Leveraging microservices architecture by using Docker technology,“ 2016. [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7506647&tag=1. [Kasutatud 02 05 2020].

Lisa 1. MyBatis koodinäide.

```
<insert id="batchInsert">
    INSERT INTO
        pdf_generator.template_text (
            template_code,
            language_code,
            text_group_code,
            ordering,
            numbering,
            numbering_level,
            text_block_name,
            text_block_id,
            text_size,
            horizontal_alignment,
            vertical_alignment)
    VALUES
    <foreach collection="templateTextBlocks"
             item="templateTextBlock"
             separator=","
    >
        (#{templateTextBlock.templateCode},
         #{templateTextBlock.languageCode},
         #{templateTextBlock.textGroupCode},
         #{templateTextBlock.ordering},
         #{templateTextBlock.numbering},
         #{templateTextBlock.numberingLevel},
         #{templateTextBlock.textBlockName},
         (SELECT tb.text_block_id
          FROM pdf_generator.text_block tb
          WHERE tb.text_block_value = #{templateTextBlock.textBlockValue}),
         #{templateTextBlock.textSize},
         #{templateTextBlock.horizontalAlignment},
         #{templateTextBlock.verticalAlignment})
    </foreach>
    ON CONFLICT DO NOTHING;
</insert>
```

Lisa 2. Rakenduses kasutatav docker-compose.yml fail.

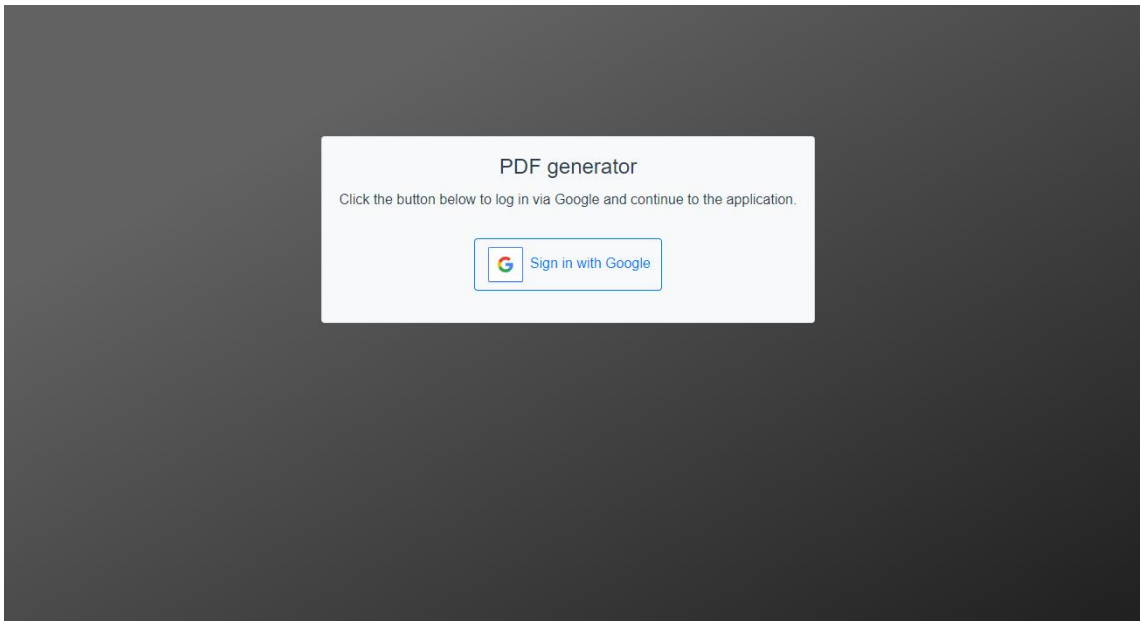
```
version: '3'

services:
  postgres:
    container_name: pdf_generator_db
    image: postgres:12.2
    restart: on-failure
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: pass
      POSTGRES_DB: postgres
    ports:
      - 7702:5432

  spring-app:
    container_name: pdf_generator_app
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 7701:7701
    depends_on:
      - postgres
    environment:
      -spring.datasource.url=jdbc:postgresql://pdf_generator_db:5432/postgres
      -front-end.address=http://dockerhost:8080

  client-app:
    container_name: pdf_generator_client
    build:
      context: client
      dockerfile: Dockerfile
    ports:
      - 8080:8080
```

Lisa 3. Sisselogimise vaade.



Lisa 4. Rakenduse avaleht samaaegselt arendaja ja redigeerija rollis olevale kasutajale.