TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Siim Liinat IADB193436

E-kirjade loomise veebirakenduse arendamine finantsasutusele

Bakalaureusetöö

Juhendaja: Einar Kivisalu
Magistrikraad

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Siim Liinat

05.01.2023

Annotatsioon

Lõputöö eesmärgiks on arendada veebirakendus pangandusettevõttele Swedbank AB, et võimaldada ettevõtte klientidele saadetavate e-kirjade jaoks vajaliku HTML koodi genereerimist.

Ettevõttes puudub olemasolev lahendus, mis võimaldaks ettevõtte kirju saatvatel töötajatel neid ise koostada, mistõttu teeb seda tellimustöö põhjal ettevõtte veebiarendusosakond käsitsi HTMLi kirjutades. Protsess võtab arendajatel palju aega, sisaldab palju korduvat tööd ning takistab nendel põhitööga tegelemast. On olemas teenuseid, mis pakuvad e-kirjade koostamist lihtsustavaid tarkvarasid, kuid ühegagi neist ei ole võimalik koostada ettevõtte disainijuhistele vastavaid kirju.

Rakendus võimaldab kõigil ettevõtte e-kirju saatvatel töötajatel neid ise koostada, isegi kui neil puuduvad vastavad tehnilised oskused koodi enda kirjutamiseks, ning kiirendab ja lihtsustab kogu protsessi. Kasutajatel on võimalik e-kiri selle osadest kokku lohistada, igat kirja osa kujundada ning lõpuks osade põhjal kirjale vastavat korrektset HTML koodi genereerida.

Rakendus valmis kasutades JetBrains IntelliJ IDEA rakendust koos Vue.js raamistiku, TypeScripti, HTMLi, CSSi ja Node.js-iga.

Rakendus testiti ning laeti üles ettevõtte serverisse ja on saadaval töötajatele läbi ettevõtte sisevõrgu.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 5 peatükki, 31 joonist, 0 tabelit.

Abstract

Development of E-mail Generation Web Application for a Financial Institution

The goal of this thesis is to develop a web application for Swedbank AB, a banking group, that can generate the HTML code necessary for the e-mails sent by the company to its clients.

The company is missing a solution that would enable its e-mail sending employees to create and send e-mails, as the current system requires technical expertise that the employees do not have. Because of this, the task has been delegated to the web development team, who possess the required skills and knowledge. The e-mail creation process also takes a lot of time, contains a lot of repetitive work, and stops developer's from working on their main tasks. Though there are services that provide easier ways for creating e-mails, none of them can create e-mails matching the company's design guidelines.

The developed web application allows anyone in the company to make e-mails, even if they lack the technical expertise; it also expedites the process, reduces errors, and allows for greater e-mail re-usability. Users can create quickly create e-mails by drag and dropping pre-made elements into the e-mail canvas and then customizing the elements per the user's needs, whether by changing colours, adding text or more. When the e-mail is done, the HTML code for the e-mail is generated automatically.

The application was developed using JetBrains IntelliJ IDEA with Vue.js and HTML, CSS, TypeScript, and Node.js.

The project was tested and published on the company's server and is available to the company's employees through its internal network.

The thesis is in Estonian and contains 42 pages of text, 5 chapters, 31 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides, ühendab üht tarkvara teisega
CSS	*Cascading Style Sheets*, stiilireeglite keel HTMLi kujundamiseks
Dark mode	Graafiliste kasutajaliideste valik või disain, mis kasutab heledat sisu tumedal taustal
HTML	*Hypertext Markup Language*, märgistuskeel veebilehe sisu Struktureerimiseks
JSON	*JavaScript Object Notation*, lihtsustatud andmevahetusvorming
Kanban	Visuaalne töövoo juhtimise meetod, mis kasutab tahvleid üksikute ülesannete veergudesse organiseerimiseks
Kutse-tegevusele nupp	*Inglise keeles „call to action“* ehk CTA nupp, innustab külastajat kohesele tegevusele, näiteks lehte külastama
localStorage	HTML5 spetsifikatsioonis määratletud mehhanism, mis võimaldab veebisaidil salvestada andmeid kliendi arvutis jäädavalt.
Sass	*Syntactically Awesome Style Sheets*, märgendkeel, mis kompileeritakse CSSiks
SCSS	*Sassy CSS*, *Sassi* uuem süntaks, mis käitub laiendina CSSile
Store	Keskne asukoht veebirakenduses, kus hoitakse rakenduse olekut
TypeScript	Tugevalt tüübitud programmeerimiskeel, mis kompileeritakse JavaScriptiks ja on mõeldud selle edasiarendusena
Vue.js	JavaScripti raamistik kasutajaliideste loomiseks
WYSIWYG	*What You See Is What You Get*, süsteem, kus muudetav sisu näeb välja nii nagu lõplikus vormistuses
XSS	*Cross-site scripting*, Murdskriptimine, rünnaku tüüp arvutiturbes

Sisukord

1 Sissejuhatus 11
2 Probleemi analüüs 13
3 Lahendusmeetodi valik 17
3.1 Nõuded lahendusele 17
3.2 Olemasolevad lahendused 18
3.3 Töövahendite kirjeldus 23
3.4 Tööprotsessi kirjeldus 25
4 Lahenduse analüüs 27
4.1 Lahenduse kasutamine 27
4.1.1 E-kirja alustamine 27
4.1.2 Uue e-kirja alustamine 28
4.1.3 Malli valimine 29
4.1.4 Rakenduse põhivaade 30
4.1.5 E-kirja eksportimise menüü 35
4.2 Lahenduse kood 37
4.2.1 Kasutajaliidese arhitektuur 37
4.2.2 Andmete hoidmine 38
4.2.3 Kirja salvestamine 40
4.2.4 HTMLi genereerimine 41
4.2.5 Baidijärjestuse märk 43
4.3 Testimine 45
4.3.1 Koodi testimine 45
4.3.2 Kasutatavuse testimine 46
4.4 Tulemused 49
4.5 Edaspidine tegevus 50
5 Kokkuvõte 52
Kasutatud kirjandus 53
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 55

Lisa 2 – Ettevõtte arendajatele saadetud küsimustik 56

Jooniste loetelu

Joonis 1. Näidis õigesti vormistatud ettevõttesisesest lingist 14
Joonis 2. Näide õigesti vormistatud ümbersuunavast lingist 15
Joonis 3. Ekraanitõmmis näidiskirjast 20
Joonis 4. Näidiskood CSSis 24
Joonis 5. Sama näidiskood SCSSis 24
Joonis 6. E-kirja alustamise menüü 28
Joonis 7. E-kirja valikute menüü 29
Joonis 8. Malli valimise menüü 30
Joonis 9. „*Text-only*“ mall 30
Joonis 10. Rakenduse põhivaade tühja e-kirjaga 31
Joonis 11. Rakendus *dark mode* valikuga 31
Joonis 12. „*Logo*" pärast lohistamist 32
Joonis 13. Elemendi valikute menüü paremal ääres 33
Joonis 14. „*Logo*“ pärast kuldkliendi variandi valimist 33
Joonis 15. Tekstiredaktor 34
Joonis 16. Logo ja värviline kast 34
Joonis 17. Tavaline eksportimise menüü 36
Joonis 18. Perioodilise kirja eksportimise menüü 36
Joonis 19. Ekraanitõmmis rakenduses genereeritud e-kirjast 37
Joonis 20. Projekti kasutajaliidese arhitektuur 38
Joonis 21. IMailOptions liidese kood 39
Joonis 22. Näide: *disclaimer*-elementi genereeriv funktsioon 42
Joonis 23. E-kirja linke vormistav funktsioon 42
Joonis 24. Venekeelne e-kiri ilma baidijärjestuse märgita 44
Joonis 25. Baidijärjestuse märgi lisamine HTML-failile 44
Joonis 26. Venekeelne e-kiri koos baidijärjestuse märgiga 45
Joonis 27. Lingi genereerimise funktsiooni ühiktestid 46
Joonis 28. Endine kirjavaliku aken Business/corporate märkeruuduga 48
Joonis 29. Uus kirjavaliku aken kahe raadionupuga 48

Joonis 30. Graaf e-kirjade koostamise rahuolu kohta..... 49
Joonis 31. Graaf rakenduse kasutuslihtsuse kohta..... 50

Tabelite loetelu

Tabeleid ei ole.

1 Sissejuhatus

Swedbank AB (edaspidi Swedbank) on Rootsis ja Baltimaades tegutsev rahvusvaheline pangandusettevõte. Oma igapäevase tegevuse käigus on ettevõttel vaja saata suures koguses erinevaid e-kirju klientide informeerimiseks ja pakkumiste saatmiseks. Ettevõttel puudub olemasolev lahendus e-kirjade koostamiseks ja seetõttu koostatakse neid manuaalselt HTMLi koodiga. Kuna kirju saatvatel osakondadel puudub vajalik tehniline oskus e-kirjade koostamiseks, siis tegeleb sellega kõigis Balti riikides veebiarenduse osakond.

Kuigi e-kirjade koostamise süsteem töötab, siis ettevõte ei soovi mitmel põhjusel sellega jätkata. Veebiarendajate peamine tööülesanne on ettevõtte internetipanga arendamine, mitte e-kirjade koostamine. Muu töö tegemine aeglustab veebiprojektide arendamist ning võttes arvesse arendajate kõrgeid palku [1], kulutab suuresti ettevõtte rahalisi ressursse. E-kirjade koostamine on ka arendajatele endile vastumeelt, sest töö on monotoonne, korduv ning mittestimuleeriv.

E-kirjade koostamise protsessil on samuti palju probleeme. Kirjade disain muutub võrdlemisi tihti, mille tõttu vanu kirju pole võimalik uuesti välja saata, isegi kui nende sisu on ajakohane, ilma kirja taasloomise või käsitsi iga aegunud koodi uuendamiseta. Kirjade koostamise käigus on vajalik teha palju väikseid töid, mida saab lihtsasti automatiseerida, kuid kus käsitsi tehes on lihtne teha vigu. Kui viga ei avastata, läheb kiri vigasena välja ja kui avastatakse, siis tuleb nii arendajal kui töö ülevaatajal kulutada veelgi rohkem aega ülesande täitmisele. Kirjadel on tihti ka muutujad näiteks kliendi nimi kirja tervituses või KUI-SIIS-tehetes, mida ei ole hetkel võimalik arendajal efektiivselt kontrollida.

On saadaval mitmeid olemasolevaid lahendusi kirjade koostamiseks, mida ettevõte saaks kasutada, kuid nendes ei ole võimalik luua ettevõtte disainijuhistele vastavaid kirju ega elemente ning kuigi nende kasutamiseks on vaja vähem tehnilisi oskusi, siis on nad keerulised, aeglased kasutada ja tekitavad ettevõttele turvaohte, mida pank kui usaldusel põhinev ja rangete regulatsioonidega ettevõte ei saa lubada.

Selle töö eesmärk on luua veebirakendus, milles on võimalik koostada väljasaatmiseks valmis e-kirju ja automaatselt genereerida kirjale vastav HTML kood. Töö esimestel etappidel valmis kirja elementide visuaalse lohistamise süsteem ning elementide kujundamine ja muutmine. Järgnevalt lisati elementidele vastava HTMLi genereerimine ja selle failina salvestamine ning hiljem lisati ka e-kirja oleku salvestamise ja kirja salvestatud failist laadimise funktsionaalsus. Rakendus arendati vastavalt nõuetele kasutajasõbralikuks, kiiresti kasutatavaks, turvaliseks ning tulevikukindlaks.

Töö elluviimiseks uuritakse olemasoleva protsessi puuduseid, sihtkasutajate vajadusi ning turul olevaid lahendusi. Analüüsi tulemusel leitakse parim viis probleemi lahendava tarkvara loomiseks. Lahendus arendatakse, testitakse ning antakse ettevõttele üle.

Eesmärkide saavutamiseks kasutatakse erinevaid tehnoloogiaid ja vahendeid, mille hulka kuuluvad Node.js, TypeScript, Vue.js ja muud. Tulemuste kontrollimiseks kasutatakse nii automaattestimist kui kasutajate ja kasutatavuse testide tagasisidet.

Antud töö annab ülevaate loodud projektist. Töö koosneb neljast olulisest peatükist: probleemi analüüs, lahendusmeetodi valik, lahenduse analüüs ja kokkuvõte.

Probleemi analüüsis kirjeldatakse põhjalikumalt probleemi olemust ja mõju ettevõttele, lahendusmeetodi valikus täpsustatakse probleemile vastava lahenduse nõudeid, analüüsitakse olemasolevaid lahendusi ning kirjeldatakse tööprotsessi. Lahenduse analüüsis kirjeldatakse loodud lahendust ja selle ülesehitust ning hinnatakse tulemusi.

2 Probleemi analüüs

Swedbank Baltics AS, mille alla kuuluvad Swedbank AB Eesti, Läti ja Leedu filiaalid, on Baltimaade suurim pangandusettevõte umbes 3,3 miljoni privaat- ja 282 000 ärikliendiga [2]. Neile klientidele tuleb igapäevaselt saata suures koguses e-kirju lepingute, meeldetuletuste, hinnakirja muudatuste, fondide tulemuste, kampaaniate ja muuga.

E-kirjad saadetakse välja kahel kujul – HTML ja avatekst. HTML lubab e-kirjade kujundust ja sisu täpselt määrata ning lisaks võimaldab ka pilte, videoid, linke ja muud. Avatekst samas sisaldab ainult kirja teksti ning mitte rohkemat; see piirab kirja võimalusi, kuid samas laeb kiiremini ja on paremini loetav ekraanilugejatele [3].

Kuigi ettevõtte kirjad kasutavad üldiselt korduvaid kujundusi, siis on kirjade nõudmised liiga erinevad, et ei ole otstarbeline luua malle, mida hiljem taaskasutada. Iga kiri tuleb eraldi koostada, lähtudes ettevõtte disainerite loodud e-kirja disainijuhendist, kus on olemas nii iga kirjas kasutatava elemendi kujundus kui ka sellele vastav HTML kood.

Vähestel töötajatel on vajalikud tehnilised oskused ja ka õigused kirja koostamisega hakkama saada, seega kui kellelgi kogu ettevõtte peale on vaja koostada uut e-kirja, siis peavad nad selle ülesande esitama veebiarenduse meeskonnale, kellel on nii teadmised kui õigused. Praegune kirjade koostamise süsteem on järgnev: tellija, kellel on kirja vaja, loob ettevõtte tööülesannete süsteemis uue ülesande, kuhu ta lisab kõik vajaliku info, muuhulgas kirja teksti, kujunduse ja kirjas kasutatavad pildid. Arendaja koostab kirja info põhjal vastava HTML koodi ning kui valmis, loob ettevõtte meediaserveris uue e-kirja, kuhu on võimalik lisada kirja HTML kood kõigis keeltes, nende avateksti versioonid ning ka kirjade pealkirjad ja väljasaatmise emailiaadressid. Arendaja paneb iga keele HTML koodi meiliserverisse, täidab ära avatekstid, pealkirjad ja saatmisaadressid ning saadab töö tellijale kirja järjekorranumbri. Töö tellija vaatab kirja üle ning kui kõik on korras, on arendaja töö lõppenud, kuid kui ei ole, siis arendaja teeb muudatused, annab tellijale teada ja protsess kordub kuni kiri on lõplikult korras.

See ei ole aga pikaajaliselt sobiv lahendus. Veebiarendajatel on tööülesanded, mida neil on vaja täita, ning e-kirjade kirjutamine ei käi selle alla, vaid on ajalooliselt ettevõttes nii jäänud. Kirjade koostamine takistab päris tööde täitmist ning langetab meeleolu, sest üldiselt ei meeldi kellelegi seda praegusel kujul teha. Kirjade koostamine võib ajendada ka töötajat ettevõttest lahkuma, et saada tegeleda rohkem valdkonnaalase tööga. Kirjade HTML kood on ka erinev tavalisest veebiarenduses kasutatavast HTMList, olles palju vanamoodsam [4], mistõttu arendajad saavad küll tööga hakkama, kuid neil puudub vastav spetsialiseerumine, et koodi hallata ning vajadusel tulevikus korrigeerida.

HTMLis kirjade koostamine on väga aeglane protsess: töötajal tuleb uut kirja alustades kas varasemate kirjade seast üles leida võimalikult sarnane kiri või siis alustada uut kirja nullist. Seejärel tuleb iga elemendi jaoks kokku kopeerida sellele vastav HTML kood ning elementi koodis edasi kujundada vastavalt olukorrale. Kas või väikeste muudatuste tegemine võib võtta palju aega, sest tihti tuleb teha palju käsitsi tööd. Näiteks kõik kirjades kasutatavad täpitähed ja erisümbolid tuleb arendajatel käsitsi kodeerida; teksti rasvaseks, kursiivi või värviliseks tegemiseks tuleb kasutada vastavalt HTMLi *,<i> ja * märgendeid; värvilise bloki taustavärvi muutmiseks tuleb värvi muuta neljas erinevas kohas, sest kast on jagatud neljaks kastiks, millel igaühel on oma taustavärv, ning muud taolist. Lisaks üldisele koodile tuleb ka igas kirjas kindlatesse kohtadesse sisestada kirja järjekorranumber ettevõtte meiliserveris, kirja riik ning keel. Sama info tuleb ka lisada kõikidesse kirjas olevatesse linkidesse koos lingi järjekorranumbriga, mis samuti tuleb käsitsi sisestada. Lingid jagunevad veel omakorda kaheks: ettevõttesisesteks linkideks (Joonis 1) ning ümbersuunavateks linkideks (Joonis 2). Olukorrale lisab keerukust, et mitte kõik ettevõtte lingid ei lähe esimese kategooria alla, vaid ettevõtte dokumendid, blogipostitused ja mõned muud veebiaadressid saavad samuti ümber suunatud. Pikemates kirjades võib linke olla üle kümne, mille vormistamine võtab tihti väga palju aega. Kuna kirja koostamisega kaasnevaid ülesandeid on palju, siis on väga lihtne midagi unustada või kahe silma vahele jätta.

```
$feedbackHandlerUrl?mailId=$mailId&url=https://www.swedbank.ee/?language=EST&
WT.seg_4=MCR_40000_est&SW.link=link_1&WT.dcsvid=$mailId
```

Joonis 1. Näidis õigesti vormistatud ettevõttesisesest lingist

```
$feedbackHandlerUrl?mailId=$mailId&url=https://www.swedbank.ee/rdt?WT.seg_4=M
CR_40000_language=EST&SW.link=1&WT.dcsvid=$mailId&redirectlink=https://www.ta
ltech.ee/
```

Joonis 2. Näide õigesti vormistatud ümbersuunavast lingist

Kirjades on tihti sees muutujad, mis e-kirjade süsteemis asendatakse kindlate väärtustega nagu kliendi nimi, ja KUI-SIIS-tehted, mille puhul sama kiri saadetakse paljudele klientidele, kuid e-kirjade süsteem määrab enne saatmist, milline tehtesisene osa on konkreetsele kliendile nähtav või mitte. Neid on arendajatel väga raske enne saatmist testida ─ettevõtte e-kirjade süsteemis on küll muutujate ja KUI-SIIS-tehete tugi, kuid see on väga piiratud ja arendajad on lõpuks üldiselt sunnitud ise kõiki olukordi manuaalselt läbi testima, et veenduda, et kiri on õigesti koostatud.

Probleemiks on samuti kirjade taaskasutatavus. Tihti soovivad e-kirjade tellijad, et arendaja võtaks vana kirja ja saadaks selle otse või väikeste muudatustega välja, sest kirja sisu on jätkuvalt täiesti õige. Ettevõtte e-kirjade disain muutub aga regulaarselt, mistõttu ei ole võimalik vanu kirju otse kasutada, vaid arendajal tuleb kogu kiri käsitsi uuendada, mis võib võtta rohkem aega kui uue kirja loomine. Oleks vaja, et vanade kirjade sisu jääks samaks, kuid disain uueneks automaatselt.

Praegusel lahendusel on probleemiks ka turvalisus. Arendajate loodud e-kirju vaatavad üle ainult töö tellijad. Kuid nemad hindavad ainult seda, kas kirja välimus vastab ettevõte disainistandardile ning kas kirja funktsionaalsus on nii nagu olema peaks – lingid ja nupud on korrektsed, tekst on loetav, kirjas kasutatavad muutujad korrigeeruvad e-kirja baasi poolt etteantutega. Kontrollijatel puudub nii vajalik tehnoloogiline oskus kui ka võimalus e-kirjade koodi kontrollida, mistõttu on pahatahtlikul arendajal võimalik kirjadesse panna skripte, mis teostavad näiteks XSS-i rünnakuid [5]. Kui ettevõtte nime alt saadetaks välja pahavaraline e-kiri, siis sellest tekkiks ettevõttele väga suur maine- ja rahakahju ning kindlasti tooks see kaasa ka riigipoolseid karistusi, sest pankadele seatud nõudmised on väga kõrged. Kuigi modernsed e-kirja teenused rakendavad meetmeid, mis blokeerivad automaatselt enamik XSS ohte [6], siis pankade laia kliendibaasi tõttu on võimalik, et on kliente, kes uuemeelseid teenuseid ei kasuta või avavad kirju veebisirvikutes, kus taolisi meetmeid ei rakendata.

Lisaks arendajatele on praegune kirjasüsteem probleemne ka tellijatele. Kirjade koostamine ootab peamiselt arendajate ressurssi järel ning selle tõttu kirja saatmine

hilineda. Vaeva tekitab ka kirjades muudatuste tegemine, sest koodis muudatusi teha on õigus ainult arendajatel, seega isegi kui muudatus on väga väike ja töötaja poolt tehtav, tuleb neil arendajale juhised anda ja uuesti oodata kirja täitmist.

Ettevõte soovib, et edaspidi saaksid kirju koostada kirju saatvad töötajad ise, vabastades arendajad sellest ülesandest ning andes kirja saatjatele rohkem kontrolli oma töö üle.

3 Lahendusmeetodi valik

Käesolevas peatükis uurib autor kasutajate ja ettevõtte nõudeid lahendusele ning analüüsib, kas mõni turul olemasolev lahendus vastab nendele. Lisaks kirjeldatakse lahenduse töövahendeid ja -protsessi.

3.1 Nõuded lahendusele

Ettevõte soovib, et e-kirju oleks võimalik koostada mitte ainult arendajatel, vaid ka teistel töötajatel. See tähendab, et lahendust peab olema lihtne kasutada ka nendel, kes ei ole HTMLiga varasemalt kokku puutunud ja kõik HTMLiga seotu peab olema kasutaja eest peidetud või ligipääsetav ainult siis, kui kasutajal on vajalikud tehnilised oskused.

Lahenduses peab olema võimalik luua võimalikult suure hulga ettevõtte igapäevastest e-kirjadest ning olema kasutatav võimalikult suurel hulgal ettevõtte disaininimekirjas välja toodud e-kirja elementidest ja nende erinevatest võimalustest. Igat elementi peab olema võimalik konfigureerida vastavalt vajadusele, näiteks pildil määrata selle linki ja alternatiivset teksti; logol millist variant kasutatakse; pealkirjadel ja värvilistel blokkidel peab saama valida värvi. Kasutaja peab saama sisestada teksti ning nendes lihtsasti kasutada erinevaid fondistiile nagu rasvane tekst, kursiiv, värvitud või allajoonitud tekst ja ka teksti sisestada linke.

Lahendus peab e-kirja sisestatud lingid automaatselt töötlema vastavalt ettevõtte juhistele. Lingid peab saama viia ümber ettevõttes kasutusel olevale kujule, kus lisaks muule infole, on ka sees e-kirja keel, riik, lingi järjekorranumber ning e-kirja enda järjekorranumber. Kui link suunab ettevõtte välisele leheküljele, siis tuleb ka see ära määrata. Ka peaks automaatselt kodeeritama ja asendatama kõik erisümbolid.

Lahenduses genereeritud HTML peab vastama firma disainijuhistele, olema vigadeta ning korrektselt vormistatud, et seda oleks vajadusel võimalik ka arendajal üle kontrollida. Rakendus peab võimaldama ka muutujate ja KUI-SIIS-tehete toetust, selleks, et kindlat sisu saaks näidata ainult kindlatele klientidele ja oleks võimalik sisestada näiteks kliendi nime kirja päisesse. HTMLi sees peaks olema ka element, mis kogub statistikat saadetud kirjade kohta [7]. Kui muidu loetakse seda privaatsusriskiks, siis ettevõte saadab kirju ainult omaenda klientidele ja/või nõusoleku andnud isikutele.

Lahendus peab olema tulevikukindel, et oleks võimalik kasutada vanu kirju ilma neid käsitsi uuendamata või taasloomata. See tähendab, et kui kasutaja on salvestanud vana kirja, siis selle lahendusse laadides peaks disain uuenema automaatselt kõige viimasele variandile, kasutades vana kirja sees olevat informatsiooni.

Lahendus peaks olema suuteline genereerima ka e-kirja veebiversiooni. Veebiversioon on e-kirja veebile kohandatud versioon, mida kliendil on võimalik avada juhul kui kiri tal e-kirja teenuses õigesti ei avane. Seal on eemaldatud kõik muutujad ja KUI-SIIS-tehete vahelised elemendid, lingid on tavakujul ehk töötlemata ning kirjas olev andmeanalüüsi võimaldav element on asendatud sama ülesannet täitva skriptiga.

Lisaks peab genereeritud e-kiri olema turvaline. Kuna kasutajatel on võimalik panna kirjadesse ükskõik millist teksti, siis sellega kaasneb ka risk, et nad kasutavad seda pahatahtlikult ja teksti asemel sisestavad koodi, et korda viia erinevaid rünnakuid. Selle tõttu on väga tähtis kasutajate sisend üle töödelda nii, et isegi kui nad kirjutavad teksti sisse mingisuguse skripti, siis seda ei jooksutata, vaid kuvatakse kui tavalist teksti [8].

3.2 Olemasolevad lahendused

E-kirjade saatmine on ettevõtetele tavaline eesmärk ning kuna kõigil neil ei ole ressursse või oskusi, et kirju firmasiseselt koostada, siis on turul olemas palju teenuseid, mille eesmärk on teha see töö lihtsamaks, kiiremaks ja kasutajasõbralikumaks.

Selleks, et leida, kas mõni olemasolevatest lahendustest sobib probleemi lahendamiseks, analüüsiti ja võrreldi neid põhjalikumalt. Turul olevad e-kirja koostamise lahendused, mida uuriti, olid: Unlayer, Smaily, Mailchimp, Klayivo, Moosend, CoffeeCup, BEE, Designmodo, Mosaico ja Stripo. Kõik eelpool nimetatud teenused pakuvad lihtsustatud e-kirjade koostamise võimalust, kus ei ole vaja e-kirja koodi kirjutada, vaid kirja saab eelloodud elementidest hiirega kokku lohistada ja siis igat elementi täpsemalt edasi kujundada.

Lahenduste hindamiseks kasutati eelnevas alapeatükis välja toodud nõudeid. Nõuetele vastavuse testimiseks kasutati näidiskirja (Joonis 3), mis on sarnane tüüpilisele ettevõtte poolt klientidele välja saadetavale e-kirjale ning sisaldab kõige tavalisemaid kirja elemente: ülevalt alla järjekorras ettevõtte logo; pilt; pealkiri; tekst; värviline blokk koos teksti ja muutujaga; värviline blokk koos pildi, punktide ja nupuga; tekstikujul signatuur

ning kirjaalune täpsustav tekst. Testi käigus prooviti näidiskirja teenustes taasluua ning samal ajal vaadati ka kui lihtne oleks edaspidi sarnast ülesannet täita – kas peab iga kord kirja uuesti looma või saab järgnevatel kordadel juba koostatud elemente uuesti kasutada. Paljud teenustest hõlmasid endas lisaks e-kirja koostamisele ka muud e-kirjadega seonduvat nagu kirjade väljasaatmine, andmeanalüütika ja statistika kogumine kirja avamise ja linkidele vajutamiste kohta ning ka kirjade automaatne saatmine näiteks kliendi esimese ostu puhul. Kuna ettevõttel on vastavad nõudmised juba kaetud, siis neid arvesse ei võetud.

Joonis 3. Ekraanitõmmis näidiskirjast

Suurim probleem, mis esines, oli see, et mitte üheski testitud lahenduses ei olnud võimalik järgi teha isegi kergelt keerulisemaid elemente, mida ettevõte oma kirjades regulaarselt kasutab. Käesolevad teenused on pigem suunatud e-kaubanduse ja väikeettevõttetele,

mistõttu on nendes kasutatavad kujundused ka teistsugused panga pigem konservatiivsest stiilist. Näidiskirja koostades sai enamiku olemasolevate lahendustega muuta kirja taustavärvi ning adekvaatselt sisestada ettevõte logo, pealkirja ja teksti, kuid üheski peale Stripo ei suutnud autor luua värvilise taustaga kaste, isegi kui nendes oli sees ainult tekst ja mitte pilt. Paljud olemasolevad lahendused lubavad küll lõpuks tulemuse saavutada sisestades kirja sisse HTML koodi, millega saaks asendada mittevõimalikud elemendid, kuid see ei lähe kokku eesmärgiga asendada manuaalset HTML kirjutamist. Ka peaks seda HTMLi kirjutama arendajad, mistõttu ei ole see sobilik tavakasutajatele. Veel oli ettevõtte disainijuhiste järgimine väga vaevarikkas ja aegavõttev, isegi kui sellega lõpuks hakkama saadi, mis ei vasta seatud lihtsus- ja kiirusnõuetele. See tulenes peamiselt sellest, et olemasolevad lahendused, kui müüdavad teenused, üritavad pakkuda suurt hulka võimalusi võimalikult erinevateks olukordadeks, kuid sellega suureneb märgatavalt rakenduse keerukus ja ajaline kulu, eriti menüüdes navigeerides. Kuigi oleks võimalik kohandada ettevõte disainieeskirju kasutusele võetava lahenduse võimalustega, siis see piiraks sügavalt ettevõtte võimalusi ja saadetavate kirjade varieeruvust.

Turul olevad lahendused ei võimalda ka täielikult automatiseerida kirjadega seotud käsitsi tööd. Kuigi nad vormistavad kirju nii, et täpitähti ja erisümboleid pole enam vaja kodeerida, siis ei ole üheski neis võimalik vormistada linke nii, et need sobiksid kokku ettevõtte andmeanalüüsisüsteemidega.

Paljudel lahendustel osutub probleemiks ka hind. Enamik analüüsitud teenustest esitavad arve ettevõtte klientide hulga järgi ja kuigi ühegi nende hinnakirjad 3 miljoni kliendi hinda ei näidanud ilma ettevõttega kontakti võtmist, siis kõige lähemal oli Moosend, kes 1 miljoni kliendi eest soovis 5600€/kuu. Samas on ka lahendusi, mis võtavad hinda kasutajate arvu järgi. Nende hinnad on üldiselt märgatavalt odavamad, näiteks Stripo soovib 10 kasutaja eest umbes 91€/kuus ja 7€ juurde iga lisakasutaja eest [9].

Lahendustes nagu Smaily oli küll võimalik kirja koostada, kuid ei olnud võimalik seda HTML-failina salvestada või muud meetodit pidi eksportida, vaid ainult läbi sama lahenduse kaudu kliendibaasile välja saata. Need lahendused ei ole ettevõttele sobivad, isegi kui kirja loomise osa oleks nõuetele vastanud, sest Swedbankil on juba olemas omaenda tööriistad kirjade välja saatmiseks ning kliendibaaside usaldamine ettevõttevälisele teenusepakkujale oleks panga jaoks väga suur turvarisk, eriti kuna mitmed testitud lahendused on hiljuti jäänud rünnakute ohvriks. Nii Mailchimp kui

Klaviyo langesid 2022. augustis kalastamisründe ohvriks, kus Mailchimpi puhul oli ründajal ligipääs 214 kontole [10] ning Klaviyo puhul 44 kontole [11]. Lisaks kliendibaaside lekkimise ohule, tekib väliste teenustega risk, et kui ründaja saab ligipääsu ettevõtte disainidele ja mallidele, siis on võimalik kasutada neid petterünnakutes, kus ründaja saadab välja Swedbanki kujundusega kirja, kuid küsib kliendilt nende salasõnu või muud infot. Taolised rünnakud on ettevõttel varasemalt juba juhtunud [12] ning kui ründajad saaksid enda kätte e-kirjade koostamise tööriistad või meetodid, siis oleks neil piiramatu võimalus uute rünnakute läbiviimiseks.

Kokkuvõttes, kuigi mõned testitud lahendused on adekvaatsed, siis ei vasta ükski neist kirjeldatud nõudmistele. Lahendused on aeglased ja keerulised kasutada, neis ei ole võimalik koostada ettevõtte disainistandarditele vastavaid kirju ning nende kasutamine oleks väga kallis ja/või ebaturvaline. Kui ettevõte siiski sooviks mõnda neist lahendustest kasutada, siis kõige sarnasem tulemus saavutati Stripo e-kirja koostajaga, kus on ka võimalik kiri lihtsasti HTML kujul eksportida.

Lisaks uuriti teenust Parcel.io, mis erinevalt eelnimetatud lahendustest on puhtalt e-kirjade HTMLi kirjutamiseks mõeldud integreeritud arenduskeskkond. Kuigi see ei sobi probleemi lahenduseks, siis võiks ettevõte seda kaaluda, kui otsustab jääda praeguse käsitsi HTMLi kirjutamise meetodi juurde. See võimaldab märgatavalt kiirendada e-kirjade HTMLi kirjutamise protsessi sisaldades funktsionaalsust, mida üldistes arenduskeskkondades pole, peamiselt kooditükkide salvestamine taaskasutatavate komponentidena ja e-kirjade spetsiifiline koodianalüüs. Kahjuks jookseb Parcel.io samuti hinnaprobleemi otsa, sest iga litsents maksab 560€/aastas, mis ka väikse meeskonnaga läheb üpris kalliks, arvestades, et tööd on siiski võimalik teha ka juba ettevõttes olemasolevate arenduskeskkondadega.

Kuigi olemasolevad lahendused ei sobinud probleemi lahendamiseks, siis oli mõningail neis funktsionaalsused, mida töö käigus loodavasse lahendusse kaaluda lisada. Paljud lahendused pakkusid kirja eelvaadet, mis on kasulik, et näha kuidas kiri välja näeb nii arvutiekraanil kui ka telefonis. Mõningad teenused, näiteks Parcel.io, võimaldavad lisaks sellele näha ka täpselt, kuidas kiri näeb välja erinevates e-kirja teenustes. See on vajalik, sest erinevad e-kirja teenused kuvavad samu e-kirju erinevat moodi, isegi kui kirja HTML-kood on sama [13]. Pankade kasutajaskond on väga erinev ning samuti on ka nende seadmete valik, millest saadavaid kirju avatakse. On väga tähtis, et loodavad kirjad

töötavad adekvaatselt võimalikult suurel hulgal seadmetel. Veel oli enamikul teenustel suur mallide ehk varasemalt disainerite poolt valmis loodud näidiskirjade album, mis on hea selleks, et kasutaja ei peaks kirju iga kord nullist looma, vaid saaks võtta aluseks mõne juba varasemalt loodud kirja.

3.3 Töövahendite kirjeldus

Rakenduse arendamiseks kasutati mitmeid takvaraarenduse platvorme ja erinevaid programmeerimiskeeli. Kuna valik jäeti peamiselt töö autorile endale, siis valis ta need, millega oli juba varasemast kogemusest tuttavam kuna ettevõttes sooviti rakendust võimalikult ruttu kasutusele võtta. Lisaks on autor juba tuttav nende parimate praktikatega.

Rakenduse arendamiseks kasutati peamiselt JetBrainsi IntelliJ IDEA arenduskeskkonda ning Vue.js raamistikku, projekti arendamisel kasutati HTML, SCSS ja TypeScript programmeerimiskeeli.

Vue.js on vabavaraline JavaScripti raamistik kasutajaliideste ehitamiseks. See on üks kõige laialdaselt kasutatud JavaScripti raamistikke ning kasutatud paljude suurettevõtete poolt nagu Google, Microsoft ja NASA [14]. Selle kasuks otsustati, sest võrreldes teiste laialtlevinud veebiraamistikega nagu React või Angular, on autoril Vue.js-iga palju rohkem kogemust, eriti uute projektide alustamisega. Vue.js on lihtne ja populaarne, mis tähendab, et seda peaksid oskama kasutada ka tulevased rakenduse arendajad ning selle võimsus tähendab, et rakendust peaks olema sujuv kasutada. Lisaks on Vue.js-is HTML, SCSS ja TypeScript kõik ühes .Vue failivormingus, mis teeb projekti haldamise väga lihtsaks.

SCSS ehk *Sassy CSS* on laialdaselt kasutatud CSS märgendkeele täiendus, mis kompileerub tagasi CSSiks. Samuti kui CSSiga, on sellega võimalik kujundada HTML elemente. SCSS sisaldab ka muid edasiarendusi, kuid peamine on see, et kui CSSis peab iga koodiblokk olema eraldi (Joonis 4), siis SCSSis on neid võimalik loogika alusel üksteise sisse panna (Joonis 5), mis lühendab koodi ning teeb selle paremini organiseeritavaks.

```
.class {
	background-color: black;
}
.class .class2 {
	color: white
}
```

Joonis 4. Näidiskood CSSis

```
.class {
	background-color: black;
	.class2 {
		color: white;
	}
}
```

Joonis 5. Sama näidiskood SCSSis

TypeScript on JavaScripti peale ehitatud tugevalt tüübitud programmeerimiskeel, mis ka kompileerub JavaScriptiks [15]. JavaScript on üks maailma populaarsemaid programmeerimiskeeli [16] ning üks levinumaid viise, kuidas veebilehtedele lisada interaktiivsust. TypeScriptil on palju eeliseid JavaScripti ees: staatiline tüüpimine aitab varem vigu püüda ja teeb koodi lihtsamini kontrollitavaks, liidesed aitavad koodi paremini struktureerida ning sellel on parem arenduskeskkondade tugi [16].

Rakenduse vaated, komponendid ning navigeerimised kirjutati Vue.js raamistikku kasutades TypeScripti ja HTMLiga ning kujundati SCSSiga. E-kirja HTMLi genereerimine, ettevõtte standarditele vastavate linkide genereerimine ning genereeritud HTMLi erisümbolite kodeerimine kirjutati TypeScriptis, et neid saaks vajadusel lihtsasti uutes projektides kasutada või ümber tõsta, kui ettevõttes soovitakse tulevikus raamistikku vahetada.

Kasutajaliidese loomiseks kasutati Vue.js jaoks loodud Vuetify-nimelist Material Design kasutajaliideseraamistikku [13]. See võimaldas lihtsasti rakendusele luua minimalistliku ja kokkusobiva kujunduse kasutades raamistiku erinevaid eelloodud disainielemente. Rakenduse tekstiredaktoriks valiti Tiptap, sest see võimaldab täielikku kontrolli stiili ja funktsionaalsuse üle ning lisaks sobitub hästi kokku nii Vue.js-i kui TypeScriptiga [14]. Elementide lohistamiseks kasutati vue.draggable.next raamistikku [15], sest olemasoleva lahenduse kasutamine oli palju lihtsam ja veakindlam kui alternatiivina olnud HTMLi sisseehitatud Drag and Drop API kaudu lohistamissüsteemi ehitamine. Rakenduseks

vajaliku tarkvara installeerimiseks, rakenduse jooksutamiseks ning kompileerimiseks kasutati Node.js keskkonda ning selle NPM paketihaldustarkvara.

Arenduse käigus kasutati ka versioonihaldusteenuseid. Alguses kasutati GitHubi koodi varundamiseks ja muudatuste salvestamiseks ning selle GitHub Pagesi funktsionaalsust, mis võimaldab tasuta rakendust veebiserveris jooksutada. See oli vajalik, sest ettevõtte arvutid on turvakaalutlustel väga kinnised ja interneti kaudu rakenduse jooksutamine oli tollal ainuke viis kuidas seda tööarvutitel käima saada. Hiljem, kui rakendus oli ettevõtte enda serveris, kasutati versioonihalduseks Atlassiani Bitbucketit ettevõtte eelistusel.

Ettevõtte töötajatele rakenduse ligipääsetavaks tegemiseks laeti rakenduse kompileeritud versioon ettevõtte serverisse.

Arenduse käigus kasutati Atlassiani Jirat projekti kanban tabeli loomiseks ja haldamiseks, mis võimaldas projekti jaotada väiksemateks, hallatavamateks tükkideks ja ka ettevõttele aimu anda projekti arengust.

3.4 Tööprotsessi kirjeldus

E-kirju genereeriva rakenduse idee tekkis autoril ettevõttega liitudes. Esimestel kuudel oli e-kirjade koostamine autori peaülesanne ning seetõttu sai ta tuttavaks nii töö enda kui ka selle plusside ja miinustega. Algselt tekkis autoril idee kirjutada ainult kood, mis automatiseeriks töö käigus esinevad lihtsasti automatiseeritavad tööd nagu täpitähtede ja muude erisümbolite kodeerimine, kirja järjekorranumbrite sisestamine, linkide loomine ja muud. Aga saades innustust ülikooliõpingute jooksul läbitud aine „Veebihalduskeskkonnad“ raames kogetud Wordpressi Blocks veebilehtede loomise lahendusest, kus oli võimalik väga lühikese ajaga veebileht eelloodud elementidest kokku lohistada ja siis vastavalt vajadusele elemente täiendada või muuta [16], tekkis tal idee kokku panna täisrakendus, mis kombineeriks nii elementide kokku lohistamise e-kirjade koostamiseks kui koodi, mis taustal käsitsi tööd automatiseerib.

Tööd tutvustati autori vanemkolleegidele ja ka hiljem ülemusele, kes andis kaks nädalat prototüübi loomiseks. Pärast kahte nädalat oli valmis algeline lahendus, kus oli võimalik elemente ringi lohistada, kuid mitte täpsemalt kujundada. Siiski oli see piisav, et rakenduse arendamine heaks kiita. Järgnevad paar kuud töötas autor rakenduse kallal üksi, hallates samal ajal kanbani tabelit, kus ta hoidis kirjas kõik tööd, mis olid tehtud,

tegemisel ning tulevikus plaanis. Projektiga tegeleti iganädalaselt – üldiselt väljaspool tööaega ehk õhtuti ja nädalavahetustel, kuid mõnikord ka tööajal. Projekti arutati regulaarselt nii ülemusega kui kaastöötajatega, aeg-ajalt korraldati ka rakenduse demonstratsioone.

Ettevõte lahenduse töövahenditele ning kasutatud programmeerimiskeeltele täpseid piiranguid ei seatud, kuid sooviti, et kasutataks modernseid lähenemisi ning agiilseid arendusmetoodikaid. Kuna autor pole edaspidi ilmselt ainukene, kes rakendust arendab, siis üritas ta jälgida ettevõttes kasutusel olevaid tavasid ning võimalikult head koodikvaliteeti, rakendades muuhulgas koodi keerukuse madalal hoidmist, süsteemide mitmekordset disainimist ning korduste vältimist [22].

Kõigepealt valmis elementide lohistamise süsteem, siis elementide lisamine ning nende kujundamine ja muutmine, järgnesid rakenduse kujunduse disainimine, elementide üksteise sisse lohistamine, e-kirjade mallid, HTMLi genereerimine ja eksportimine, tekstiredaktori lisamine kindlatele elementidele, e-kirja salvestamine nii veebisirvikus kui eraldi JSON-failina ning ka selle laadimine, lokaliseerimise tugi. Lõpus lisati automaattestid ning viidi läbi küsitlus rakenduse kasutajate seas ja ka kasutatavuse test.

4 Lahenduse analüüs

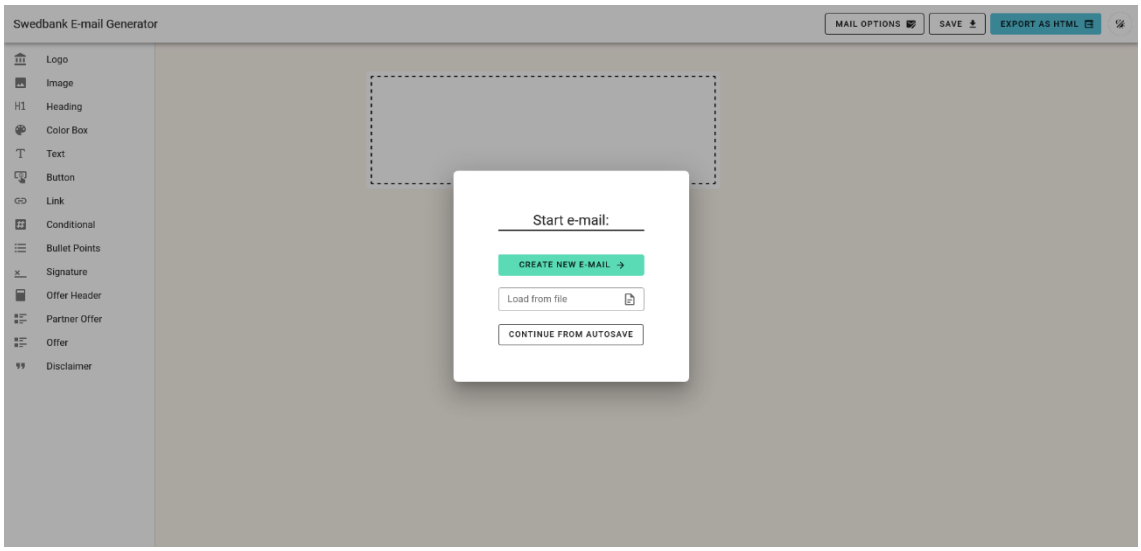
Peatükis kirjeldatakse töö käigus arendatud lahendust ja selle toimimist, uuritakse selle kasu ettevõttele ning analüüsitakse, mida saaks tulevikus juurde arendada.

4.1 Lahenduse kasutamine

Peatükk demonstreerib kuidas kasutajad on mõeldud lahendust kasutama ning toob välja ka selle funktsionaalsuse.

4.1.1 E-kirja alustamine

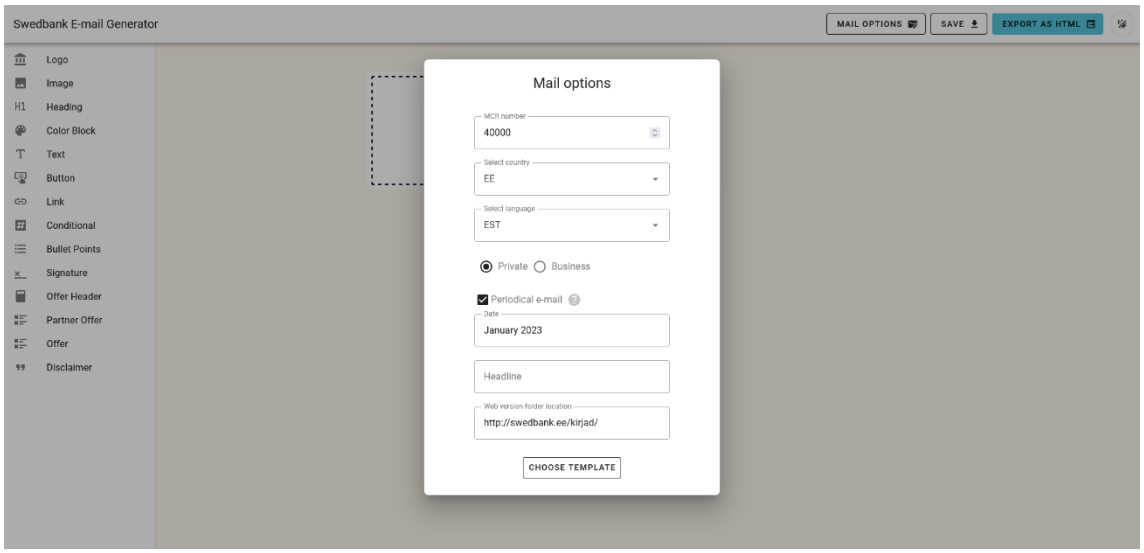
Rakendust alustades avaneb e-kirja alustamise menüü (Joonis 6), kus kasutajal on kolm valikut. Esimene valik viib uue kirja loomise sektsiooni; teine valik võimaldab kasutajal laadida olemasolevat kirja, kui ta on varasemalt selle salvestanud rakenduse siseselt JSON-failina; kolmas valik võimaldab kasutajal laadida olemasoleva kirja automaatsalvestustest, mis toimuvad kirja loomise ajal taustal automaatselt iga 10 sekundi tagant. Esimene valik on kõige tähtsam, mistõttu sellele vastav nupp on tehtud värviliseks ja silmapaistvamaks; teisejärgulised laadimisvalikud on kuvatud selgelt, kuid mitte väljapaistvalt; niimoodi on kasutajale visuaalselt kohe selge, millise valikuga esimene kord jätkata [17].



Joonis 6. E-kirja alustamise menüü

4.1.2 Uue e-kirja alustamine

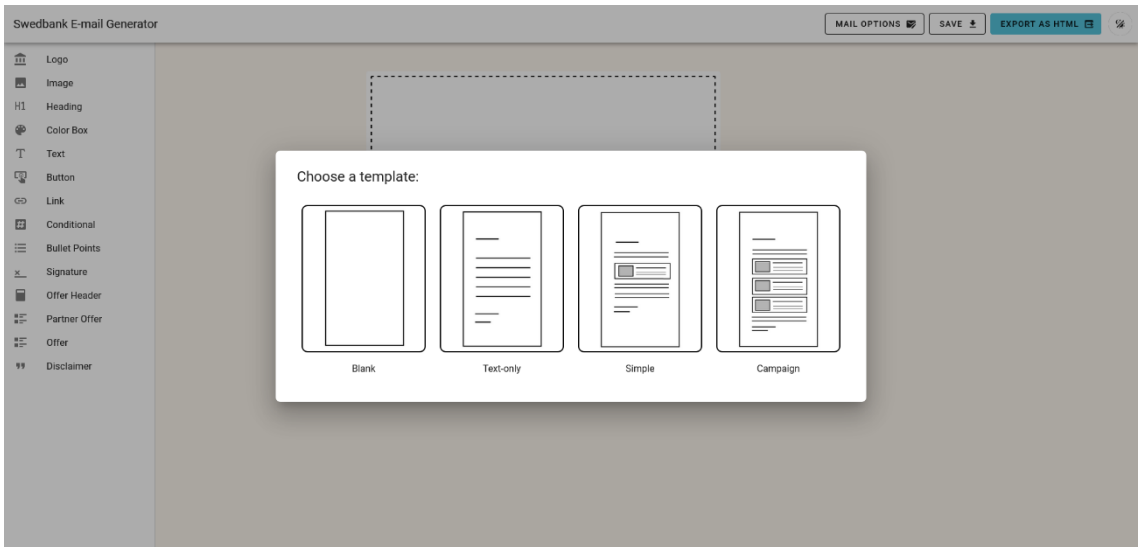
Valides e-kirja alustamise menüül uue kirja valiku, avaneb menüü (Joonis 7), kus kasutajal palutakse sisestada kirja keel, riik ja meiliserveri järjekorranumber ning ka valik, kas kiri on privaat- või ärikliendile. Kui kasutaja valib „*Periodical e-mail*“ valiku, siis avanevad lisaväljad perioodiliste kirjades jaoks. Perioodilised kirjad on taolised, mida saadetakse regulaarselt, nendeks on näiteks uudiskirjad, kuukirjad, pensionikirjad, vanemate- ja noortekirjad. Perioodilistel kirjadel on natuke teistsugune vormistus – kirja ees on saatmise kuu, aasta ja kirja veebiversiooni link. Kui kiri on perioodiline, siis genereeritakse automaatselt rakenduses ka sellele vastav veebiversioon.



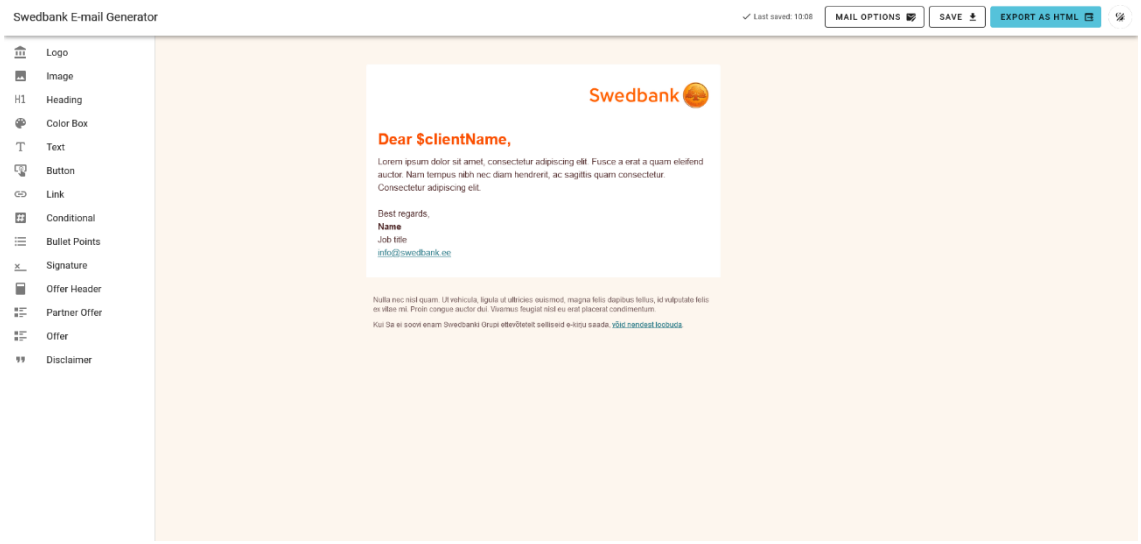
Joonis 7. E-kirja valikute menüü

4.1.3 Malli valimine

Pärast eelmise menüü väljade täitmist avaneb kasutajale mallide valimise menüü (Joonis 8). Kõige vasakpoolsem valik alustab tühja kirjaga, kuid mugavam ning kiirem on valida üks kolmest eelloodud mallist, mille valimisel saab kirja loomist alustada juba pooleldi valmis kirjast. „*Text-only*“ valik (Joonis 9) sisaldab kirja, kus on ettevõtte logo, kirja pealkiri, tekstikast, saatja signatuur ning kirjast loobumise tekstiväli kõige all. „*Simple*“ valik on sarnane, kuid sisaldab kirja üleval pilti ning kirja keskel värvilist blokki, milles on pilt, pealkiri, tekstikast ja kutse-tegevusele nupp. „*Campaign*“ valik on samuti sarnane „*Text-only“* valikule, kuid kirja keskel on kolm värvilist blokki. Need kolm tüüpi valiti, sest neid esineb kõige rohkem, kuid tulevikus on võimalik teisi veel juurde lisada, et katta ka muud juhud.



Joonis 8. Malli valimise menüü

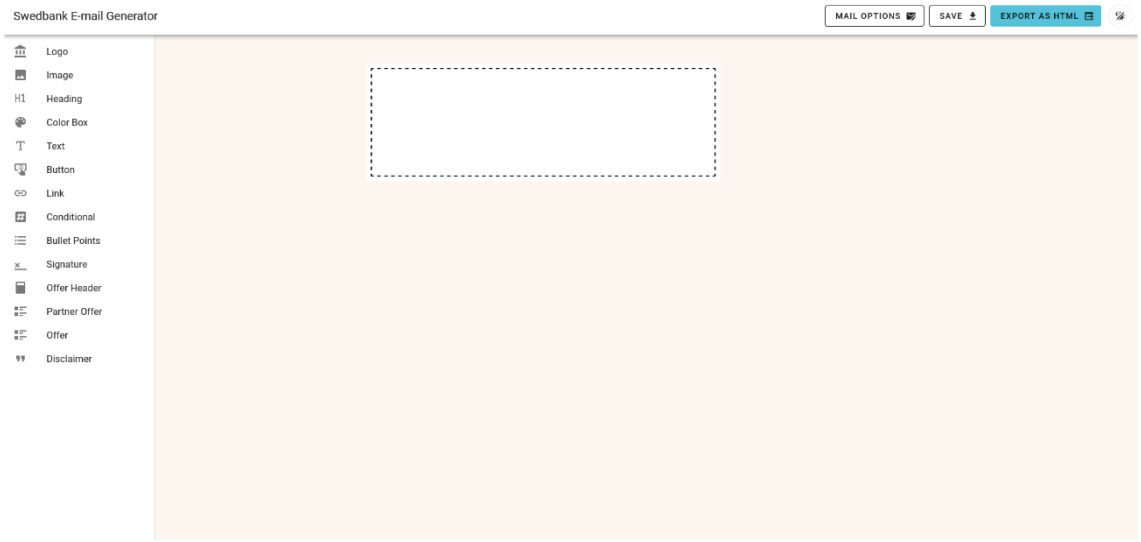


Joonis 9. „*Text-only*“ mall

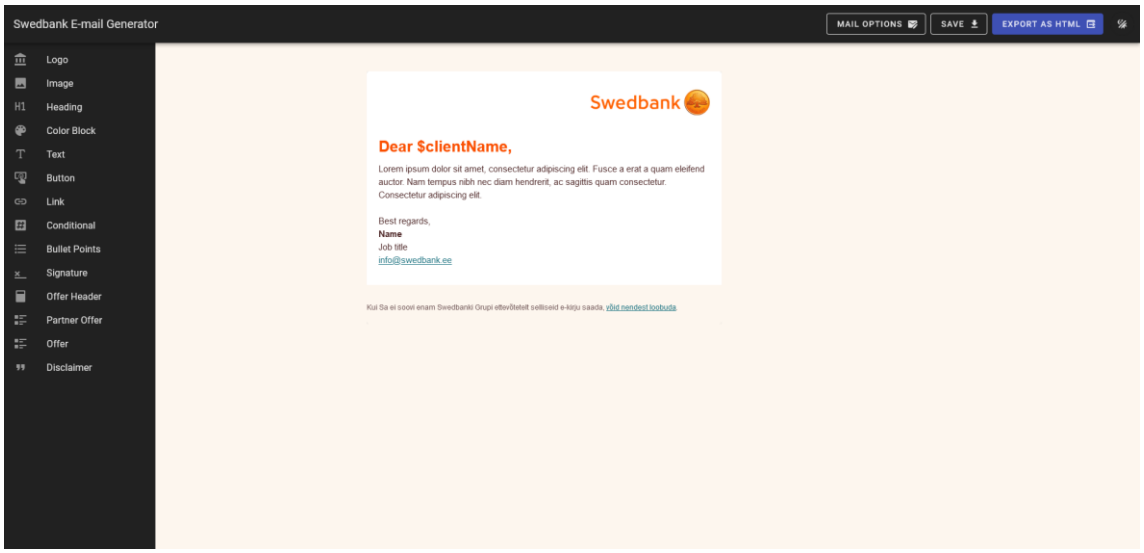
4.1.4 Rakenduse põhivaade

Olles valinud malli või laadides varasemalt salvestatud kirja, avaneb kasutajale rakenduse põhivaade (Joonis 10). Põhivaate vasakus ääres on kirjas kasutatavate elementide sektsioon, mis koosneb lohistavatest elementidest. Kirja keskel, joonisel 10 punktilise kastiga ümbritsetud ala moodustab kirja enda ala. Paremal üleval ääres on *dark mode* nupp, mis muudab mõnede kasutajate eelistusele vastavalt rakenduse kujunduse tumedamaks (Joonis 11), kuid jätab kirja värvid põhivaate keskel ja taustal samaks, et kasutajal ei tekiks kahtlusi, kas kirja lõplik kujundus sõltub ka sellest valikust. *Dark*

mode'is on lähtutud samadest disainilähenemistest kui heleda disainiga ehk kõige tähtsam nupp peaks olema värviga rõhutatud [17].



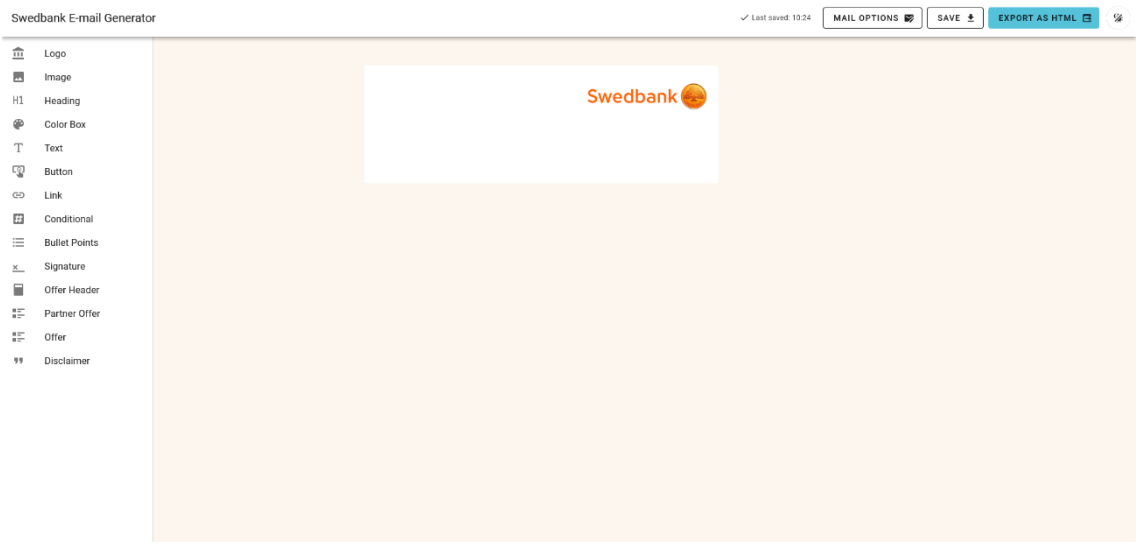
Joonis 10. Rakenduse põhivaade tühja e-kirjaga



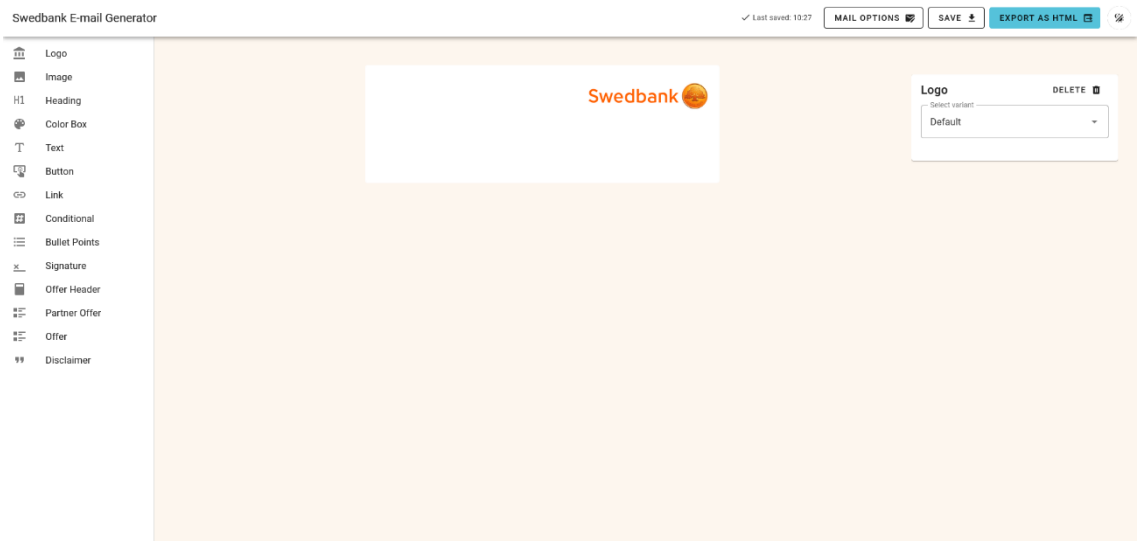
Joonis 11. Rakendus *dark mode* valikuga

Kasutaja saab kirja koostada lohistades elemendi, näiteks „*Logo*" elementide sektsioonist, kirja sisse. Lohistamise ajal on võimalik täpselt valida, kuhu lohistatav element kirja sees paigutub. Ka pärast lohistamist saab igal hetkel elementide järjekorda muuta, lohistades soovitud elemendi mingi teise elemendi ette, järele või sisse. Kindlatele elementidele on rakendatud täiendavad reeglid, näiteks saab „*Logo*" element olla kirjas ainult esimesel positsioonil ning „*Disclaimer"* ainult viimasel.

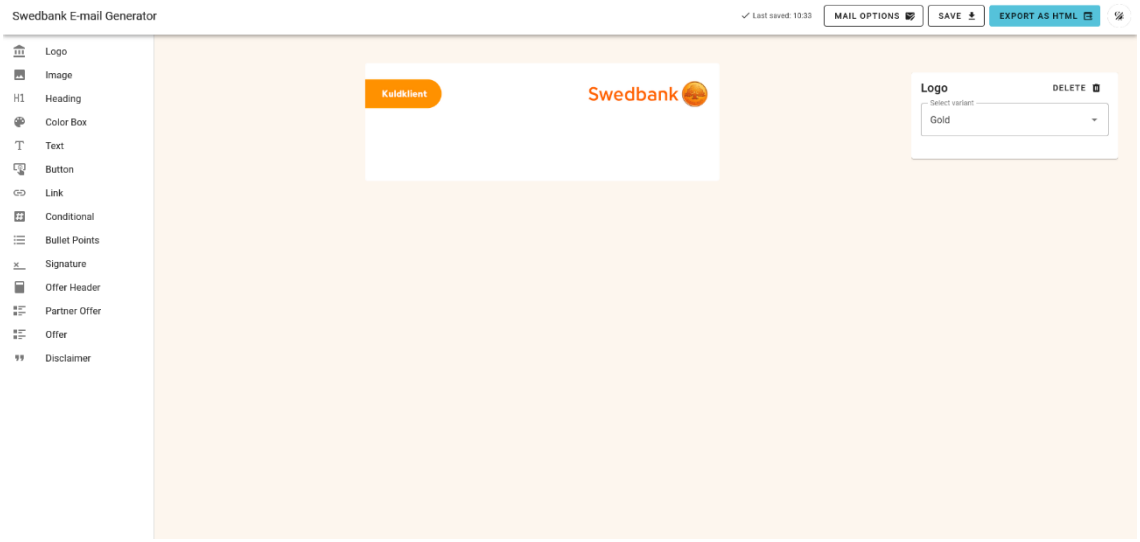
Pärast elemendi lahti laskmist element laetakse ja on näha kuidas see kirja sees päriselt välja näeb (Joonis 12). Elemendile peale vajutades avaneb paremal pool elemendi valikute menüü (Joonis 13). Kõigil elementidel on „*Delete*“ nupp, mis kustutab elemendi; pärast elemendi kustutamist on võimalik paari sekundi jooksul kustutatud element taastada vajutades kustutamise teatises olevale „*Undo*“ nupule. Lisaks „*Delete*“ nupule on iga elemendi valikute menüüs ka ainult sellele elemenditüübile vastavad lisavalikud. Logo-elemendil on ainult tüübivalik, mis vahetab ettevõtte tavalise, kuldkliendi ja privaatkliendi versioonide vahel, kuid teistel elementidel on oma valikud, näiteks pealkirjadel ja värvilistel kastidel saab muuta värvi, signatuurile saab lisada pildi ning pildile saab muuhulgas lisada teksti, mida loevad ainult vaegnägijate ekraanilugejad. Igal elemendil on täpselt nii palju valikuid, et katta kõikvõimalikud kasutusjuhud. Muutes menüüs mingit elemendi valikut, uueneb kohe ka element ise (Joonis 14).



Joonis 12. „*Logo*" pärast lohistamist



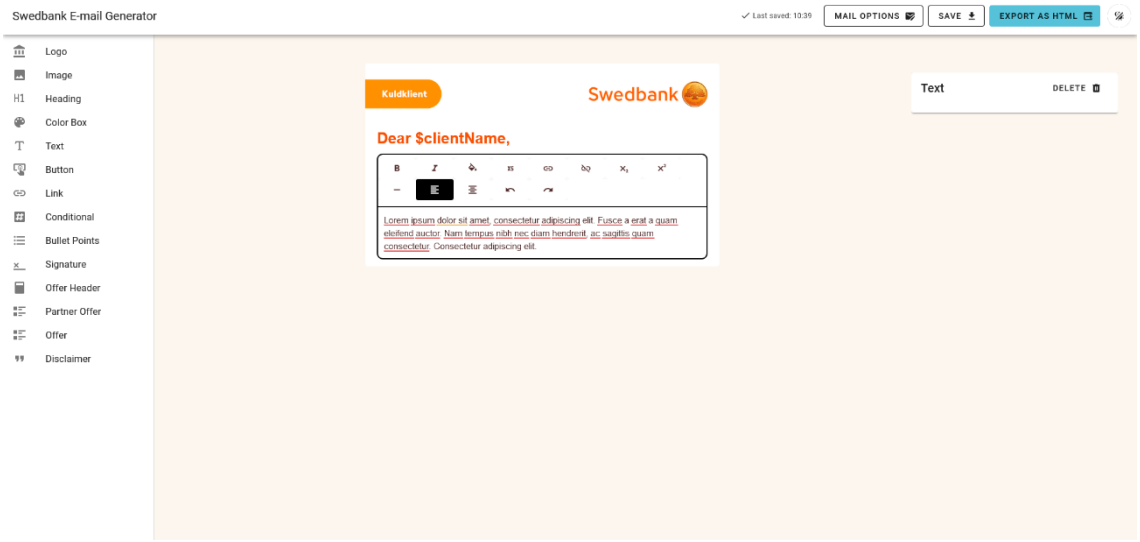
Joonis 13. Elemendi valikute menüü paremal ääres



Joonis 14. „*Logo*“ pärast kuldkliendi variandi valimist

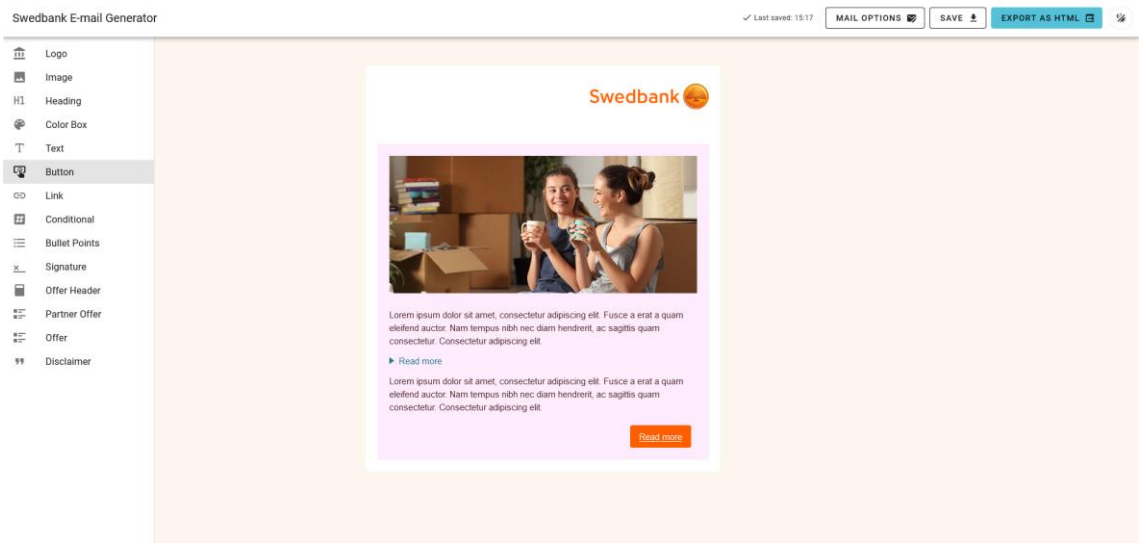
Mitmed elemendid sõltuvad kirja alguses valitud keelest, et näidata keelele vastavat sisu. Näiteks ettevõtte logod erinevad sõltuvalt riigist ja valitud keelest; kui on valitud eesti keel, siis kuldkliendi logol on tekst eesti keeles, kuid kui valida leedu keel, siis on sama tekst logol leedu keeles. Mõnede elementide sees on kasutusel tekstiredaktor – kui element on aktiivne, siis avaneb menüü, kus on võimalik lihtsasti kasutada kõiki tavalisi tekstimuutmise võimalusi nagu rasvane tekst, kursiivis tekst, lingid ja muu (Joonis 15). Kui element pole enam aktiivne, siis tekstiredaktor peidetakse ja tekstikast kuvab ainult vormistatud teksti. Tekstiredaktori jaoks valiti WYSIWIG-tüüpi lahendus, sest see võimaldab kliendil kohe näha ja muuta teksti kujundust ning seda on lihtsam kasutada kui HTML- või Markdown-märgendeid [18]. Tekstiredaktori sees töötavad ka erinevad

õigekirjakontrolli tööriistad, mis näitavad välja ja aitavad parandada trüki- ja muid õigekirjavigu.



Joonis 15. Tekstiredaktor

Teistest elementidest erinevad on värvilise kasti ja pakkumise elemendid, mille sisse on võimalik lohistada kindlaid teisi elemente. Joonisel 16 on kujutatud värviline kast, mille sisse on lohistatud pildi, kahe tekstikasti ja nupu elemendid.



Joonis 16. Logo ja värviline kast

Kirja loomise ajal on igal hetkel võimalik lehe üleval asuvast „*Mail options*“ nupust avada e-kirja valikute menüü (Joonis 7) ja muuta seal olevaid valikuid. See on vajalik, sest mõnikord soovitakse, et arendaja teeks mitu väga sarnast kirja, kus ainuke erinevus

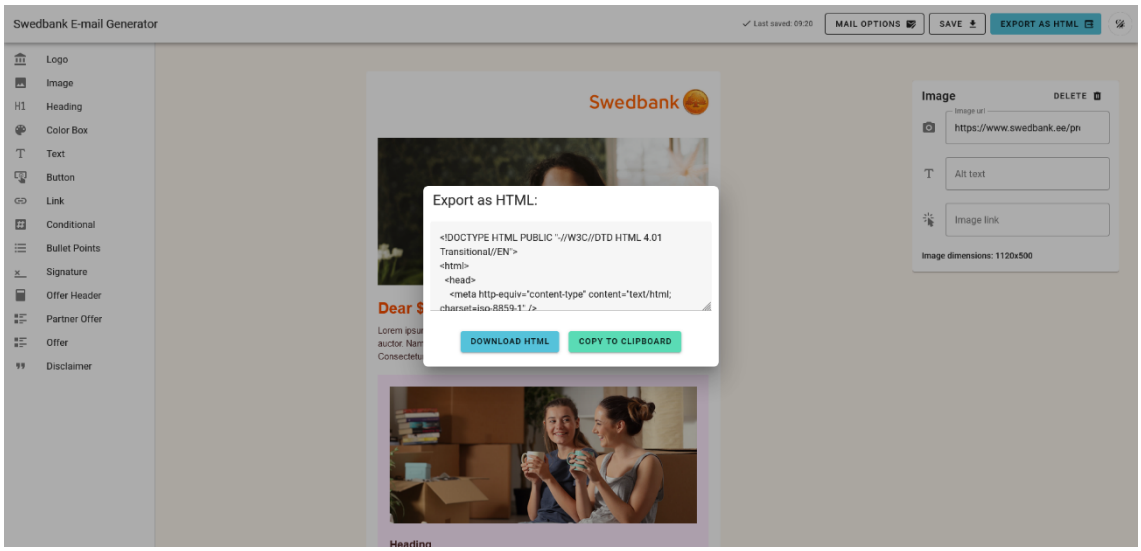
on üks-kaks lauset. Sellisel juhul ei pea iga kord looma uue e-kirja, vaid saab muuta kirja järjekorranumbrit ja siis uuesti eksportida.

E-kirja on võimalik loomise ajal salvestada JSON-failina. See võimaldab kasutajal hiljem e-kirja alustamise menüü kaudu (Joonis 6) otse jätkata kirja loomist varasemalt salvestatud kirjast, ilma, et peaks kõike uuesti tegema. JSON-faili kasutamine võimaldab ka kirjade disainil automaatselt uueneda, kuna seal on salvestatud ainult kirja ja elementide andmed, mitte kirja kujundus nagu HTMLis. Seega kui rakenduse disaini hoitakse uuendatud, on andmete põhjal alati võimalik genereerida kõige uuem disain. JSON-vorming valiti, sest see sobitub hästi kokku TypeScriptiga ja on lihtne kasutada, HTML-faili oleks palju raskem uuesti sisse lugeda ja selle sisu tõlgendada.

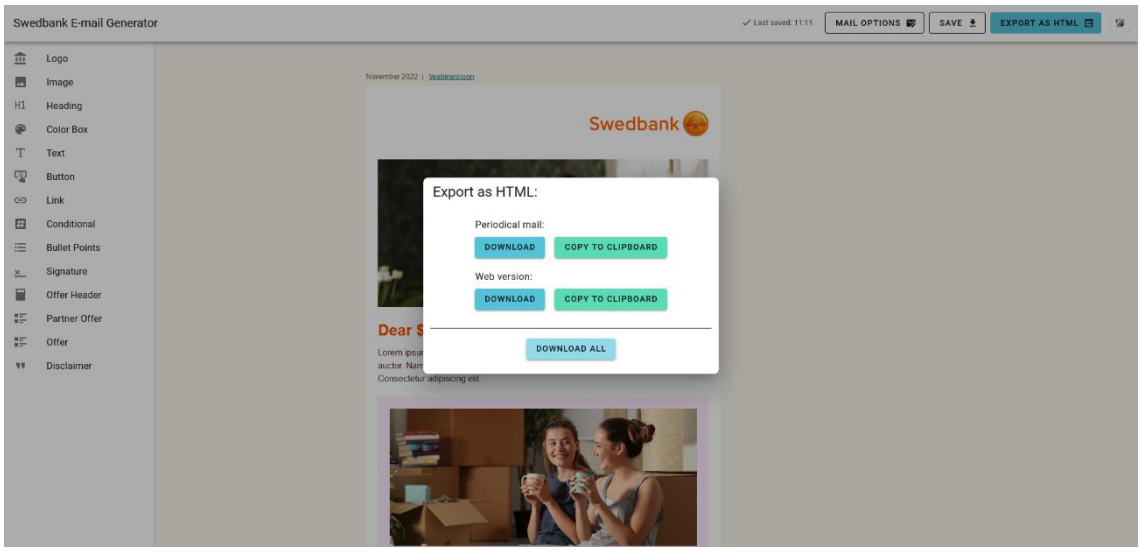
4.1.5 E-kirja eksportimise menüü

Kui e-kiri on kasutaja poolt valmis loodud, on vaja see eksportida HTML kujule. Vajutades rakenduse põhivaates paremal üleval „*Export as HTML“* nupule, mis on värviline, et see kasutajale kõige paremini silma paistaks [17], avaneb vastav eksportimise menüü. Menüü erineb vastavalt sellele, kas e-kirja valikutes valiti perioodiline kiri või mitte. Tavalise kirja puhul avaneb menüü (Joonis 17), kus on võimalik näha genereeritud HTMLi koodi enne salvestamist, salvestada genereeritud HTML failina või kopeerida lõikelauale. Kopeerimine on kasulik, sest ettevõtte meiliserverisse ei ole võimalik HTMLi failina üles laadida, vaid tuleb kopeerida selle sisu. Perioodilise kirja puhul on eksportimise menüü (Joonis 18) natuke teistsugune – kuna perioodilisel kirjal on vaja juurde ka veebiversiooni, siis on mõlemal versioonil allalaadimise ja kopeerimise valikud ning ka lisaks võimalus alla laadida mõlemad versioonid korraga.

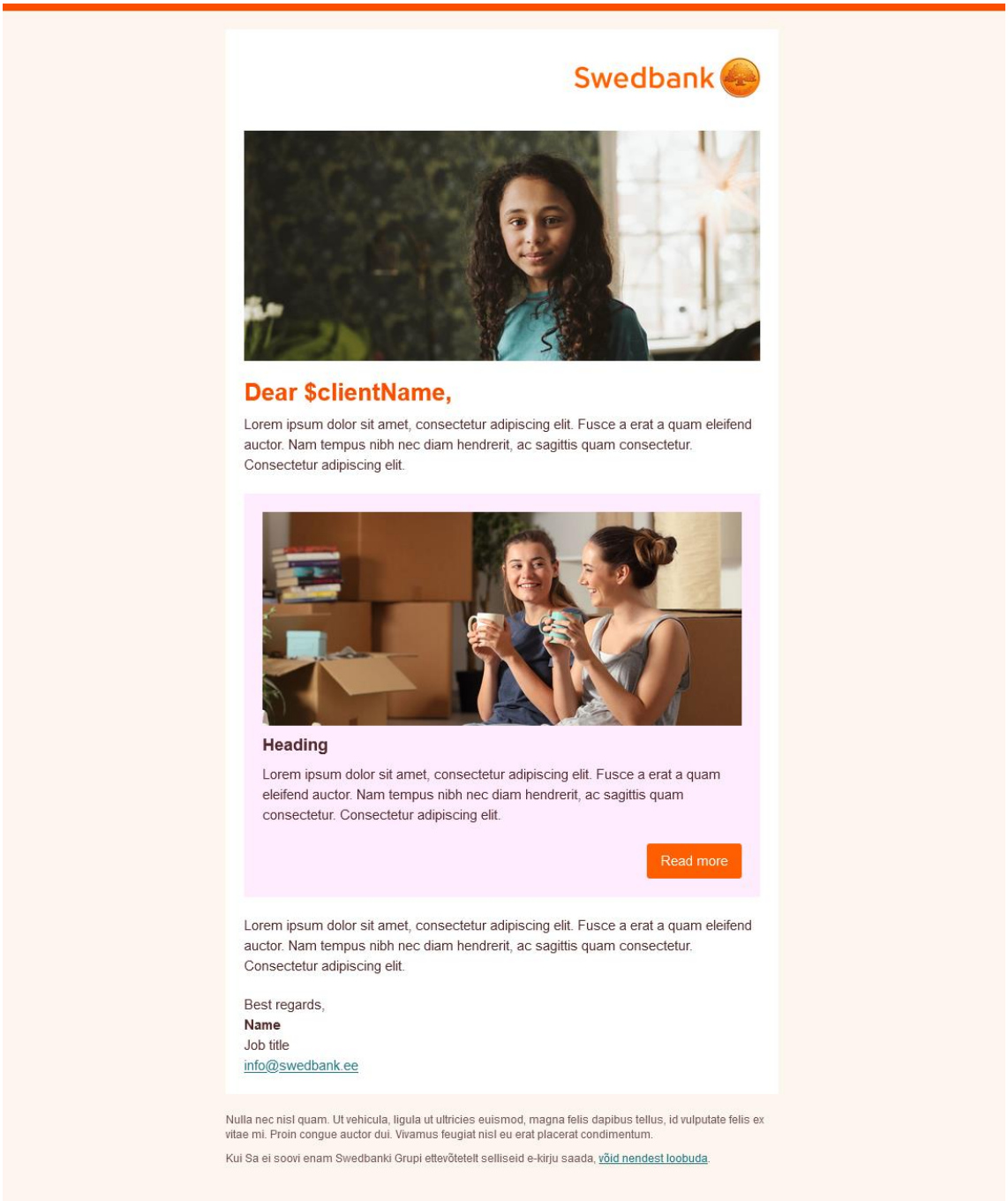
Pärast kirja koodi allalaadimist (Joonis 19) või kopeerimist saab arendaja HTMLi meiliserverisse panna ja sellega on kirja koostamine lõpetatud.



Joonis 17. Tavaline eksportimise menüü



Joonis 18. Perioodilise kirja eksportimise menüü



Joonis 19. Ekraanitõmmis rakenduses genereeritud e-kirjast

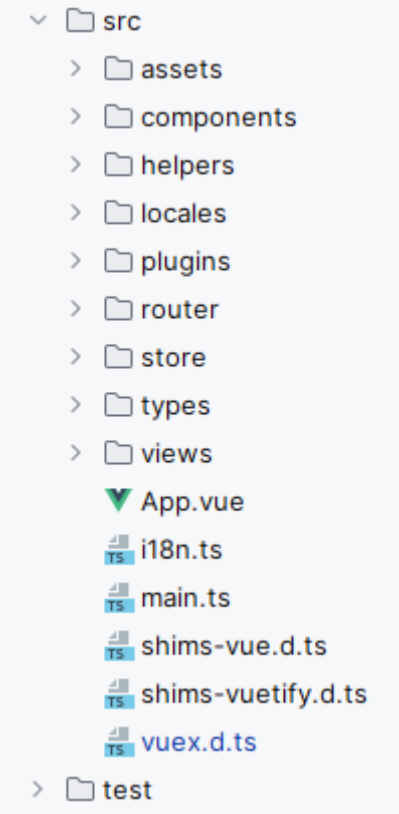
4.2 Lahenduse kood

Peatükis kirjeldatakse rakenduse koodi ülesehitust ning seletatakse selle üldiseid tööpõhimõtteid.

4.2.1 Kasutajaliidese arhitektuur

Joonisel 20 on kajastatud rakenduse arhitektuur. Rakenduses kasutati ühte põhivaadet, mis asub *src/views* kaustas, koos mitme taaskasutatava komponendiga, mis asuvad

src/components kaustas. Taaskasutatavateks komponentideks on näiteks e-kirja valikute menüü (Joonis 7) ning tekstiredaktor. *Helpers* kausta sisse pandi rakenduse poolt kasutatavad TypeScripti funktsioonid nagu linke vormistav funktsioon ning *types* kausta sisse pandi kõik rakenduse poolt kasutatavad klassid ja liidesed. *Src/assets* kaustas on rakenduse poolt kasutatav pildid ning *src/locales* kaustas on rakenduses kasutatavad tõlked.



Joonis 20. Projekti kasutajaliidese arhitektuur

Vue.js kasutab omaenda failivormingut .Vue, mille sees on <template> märgendi vahel HTML kood, <script> märgendi vahel JavaScripti või TypeScripti kood ning <style> märgendi vahel rakenduse stiilikujunduse kood CSS, SCSS ja muude alternatiivide kujul.

4.2.2 Andmete hoidmine

Uue kirja alustamisel luuakse kasutaja kirjavalikutele (Joonis 7) vastav MailOptions-objekt, mis järgib IMailOptions liidest (Joonis 21). Pärast loomist objekt salvestatakse rakenduseülesesse *store*'i, et kõigil rakenduse osadel oleks võimalik sealt andmeid lugeda, ilma, et seda peaks manuaalselt kõigile komponentidele edasi andma.

MailOptionsi väärtuseid loeb sisse rakenduse põhivaade ja enamik komponente, kus neid kasutatakse nii kirja õigesti kuvamiseks kui hiljem kirjale vastava HTMLi genereerimisel.

```
interface IMailOptions {
    MCR: string,
    country: COUNTRY,
    language: LANGUAGE,
    business: boolean,
    regular: boolean,
    date: string | null,
    headline: string | null,
    web_location: string | null,
}
```

Joonis 21. IMailOptions liidese kood

Kirja sisu moodustavaid elemente hoitakse elementListis, mis on elemendile vastavate Element-objektide loend. Seda hoitakse samuti *store'is*. elementListi kasutab peamiselt elemente kirja sees kuvav komponent, kuid ka elemendi valikute komponent ning kirja lõpus HTMLi genereeriv funktsioon. Element-klass rakendab IElement liidest, millel on väljad:

- id - elemendile unikaalne järjekorranumber, mis on vajalik, et lohistamisel elementidel järge hoida;
- *type* - määrab elemendi tüübi, näiteks logo, värviline blokk, tekstikast;
- *subElements* - loend elemendi lapselementidest; seda kasutatakse ainult nende elementide puhul, mille sisse saab teisi elemente panna, näiteks värviline blokk;
- *options* - sisaldab kõiki valikuid, mida kasutaja saab elemendi kohta muuta, sõltub tüübist.

Kui kasutaja elemendi kirja lohistab, siis luuakse uus Element-objekt. Objektile antakse järjekorranumber ning vaikeväärtused, mis sõltuvad elemendi tüübist ja kirja keelest. Näiteks kui kasutaja lohistab *Heading*-elemendi eestikeelsesse kirja, siis on selle tekstivälja vaikeväärtus „Tere, $clientName", kuid inglisekeelses kirjas „Dear $clientName". Pärast järjekorranumbri ja vaikeväärtuste määramist lisatakse objekt elementListi, rakendus uuendab automaatselt kirja sisu uue elemendiga ning kuvab selle kasutajale vastavalt elemendi tüübile ja selle valikutele.

4.2.3 Kirja salvestamine

Iga 10 sekundi tagant toimub rakenduses automaatne e-kirja salvestamine *localStorage*'isse ehk veebisirviku kohalikku andmebaasi. Selle käigus salvestatakse MailOptions, elementList, salvestamise kuupäev ja kellaaeg ning rakenduse versioon ühtsesse SaveFormat objekti, mis *localStorage*'isse salvestamiseks tuleb muuta JSON-kujule. Korraga hoitakse *localStorage*'is kuni viit kirja salvestust. Kui salvestuste hulk on täis, siis käiakse ükshaaval läbi kõik varasemalt tehtud automaatsalvestused ja neid võrreldakse uue salvestusega. Vanade salvestuste lugemiseks tuleb need ajutiselt JSON-kujult muuta tagasi JavaScripti kujule. Kui ühegi varasema salvestatud kirja MailOptions on identne olemasoleva kirjaga ja elementListid on erinevad ehk vahepeal on tehtud muudatusi, siis varasem salvestus asendatakse uuega. Kuigi kiirem oleks salvestusi võrrelda ainult kirja järjekorranumbri väärtuse järgi, siis kogu kirja valikute võrdlemine tähendab, et samal ajal võivad olla salvestatud sama kirja erinevad versioonid. Kui ükski varasem salvestus ei kattu ja salvestuste arv on juba täis, siis salvestatakse üle kõige vanem salvestus.

Kasutajal on võimalik kirja ka JSON-failina salvestada. Sellisel juhul pannakse samuti kirja MailOptions, elementList, kuupäev ja kellaaeg ning versioon SaveFormat objekti, kuid pärast JSON-kujule muutmist tehakse andmed kasutajale allalaetavaks Filesaver.js-nimelise failisalvestusteegiga.

Kui kasutaja laeb salvestatud kirja automaatsalvestusest või failist, siis kõigepealt tõlgendatakse need JSON-kujult JavaScripti kujule, pärast mida loetakse sisse MailOptions ja elementList, et kasutaja saaks kirja koostamist jätkata. Kuupäeva ja kellaaega kirja sees ei kasutata, vaid seda näidatakse ainult kasutajale enne laadimist, et too teaks täpsemalt millist kirja ja versiooni ta jätkamas on. Rakenduse arenduse käigus võivad andmed kuju muuta ehk varasemalt salvestatud kirjal võib olla element, mida rakenduses enam ei kasutata või hoopis puudu elemendiväli, mida salvestamise ajal ei olnud. Kui võimalik kasutatakse kirjaga kaasa salvestatud rakenduse versiooni, et vanad andmed tõlgendada uuteks andmeteks. Kuid kui see pole võimalik, siis mittetunnustatud elemendid eemaldatakse laadimisel elementide loendist ning puudu olevatele elemendiväljadele antakse vaikeväärtused.

4.2.4 HTMLi genereerimine

Rakenduse poolt genereeritud HTML-fail koostatakse kasutaja lohistatud ja kujundatud elementide ning nende järjestuse põhjal. HTML pannakse rakenduses kokku tekstikujul ning alles lõpus salvestatakse HTML-failina.

Alguses lisatakse HTMLi dokumenditüübi deklaratsioon, mis peab olema iga HTML-faili alguses, ning <meta> märgendi sees olevad andmed nagu faili kodeering, mis sõltub kirja riigist ja keelest ning määrab kirjas kasutatavad karakterid. Sellele järgneb <*style*> märgendite sees kirja stiili määrav CSS-kood.

E-kirjade HTMLi kirjutatakse peamiselt tabelite kaudu ehk <*table*>, <*tr*> ja <*td*> märgenditega, sest need on toetatud kõikide e-kirja teenuste poolt [19]. See tähendab, et e-kirjade koodis on sisu üks suur tabel ning kõik selle sees olevad elemendid on samuti vormistatud tabelitena.

Iga rakenduses kasutatava elemendi kohta on rakendusse salvestatud selle tüübile vastav HTML-kood, mis võeti ettevõtte e-kirja disainijuhisest ning on muudetud, et arvestada ka elemendile vastavaid muutujaid (Joonis 22). Rakendus käib ükshaaval läbi kõik kasutaja kokku lohistatud kirja elemendid, edastades elemendi funktsioonile, mis tagastab elemendi tüübile ja andmetele vastava HTML-koodi, lisades selle koostatavale HTMLile. Kui elemendil on sees kasutaja poolt sisestatud teksti, siis see kodeeritakse, et vältida probleeme erisümbolitega ning potentsiaalsete skriptidega. Kui elemendil on olemas alamelemendid, mis on elemendi sees, siis samamoodi käiakse ka need kõik läbi, kuid funktsioonile antakse juurde parameeter, et tegu on alamelemendiga – kui element on teise elemendi sees, siis tulenevalt väiksemast ruumist ning põhielemendi värvilisest taustast, on selle vormistus erinev. Elementide vahele lisatakse tühja ruumi <*div*> märgenditega, mille kõrgus sõltub vastavalt sellele eelneva ja/või järgneva elemendi tüübist.

```
function generateDisclaimer(element: IElement) {
  return `<table border="0" cellpadding="0" cellspacing="0" style="max-width:560px;font-family:Arial,sans-serif;
           font-size:15px;color:#512b2b;background-color:#ffffff;">
           <tr>
             ${element.options.imageUrl ?
               `<td width="90" height="90" valign="top">
                  <img src="${element.options.imageUrl}" border="0" width="90" height="90"
                       alt="${element.options.altText}" />
                </td>
                <td width="10"> </td>`
             : ''}
             <td width="${element.options.imageUrl ? 460 : 560}" style="font-family:Arial,sans-serif;font-size:15px;color:#512b2b;line-height:22px;">
               ${element.options.text}
             </td>
           </tr>
         </table>`;
}
```

Joonis 22. Näide: *disclaimer*-elementi genereeriv funktsioon

Kui elemendi sees on link, siis edastatakse see link linke genereerivale funktsioonile *generateUrl()* (Joonis 23), mis loeb parameetritena sisse lingi aadressi ning kirja keele, riigi ja meiliserveri järjekorranumbri. Lingid jagatakse kaheks ümbersuunatavate (Joonis 1) ja ettevõtte oma linkide (Joonis 2) vahel, mõlemate puhul link vormistatakse vastavalt ettevõtte standardile kasutades etteantud parameetreid.

```
export default function generateUrl(url: string, country: string, language: string, mcr: string, link_id: string) {
    if (isRedirect(url)) {
        return `$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.${country.toLowerCase()}/rdt?WT.seg_4=
                MCR_${mcr}_${language.toLowerCase()}&amp;SW.link=${link_id}&WT.dcsvid=$mailId&amp;redirectlink=${url}`
    }
    else {
        const splitUrl = url.split( splitter: /(?=#)/g);
        const strippedUrl = splitUrl[0];
        let tag = splitUrl[1];
        if (tag === undefined) tag = "";
        return `$feedbackHandlerUrl?mailId=$mailId&amp;url=${strippedUrl}${url.includes("?language=") ? '' : '?language=' + language}
                &amp;WT.seg_4=MCR_${mcr}_${language.toLowerCase()}&amp;SW.link=link_${link_id}&amp;WT.dcsvid=$mailId${tag}`;
    }
}
```

Joonis 23. E-kirja linke vormistav funktsioon

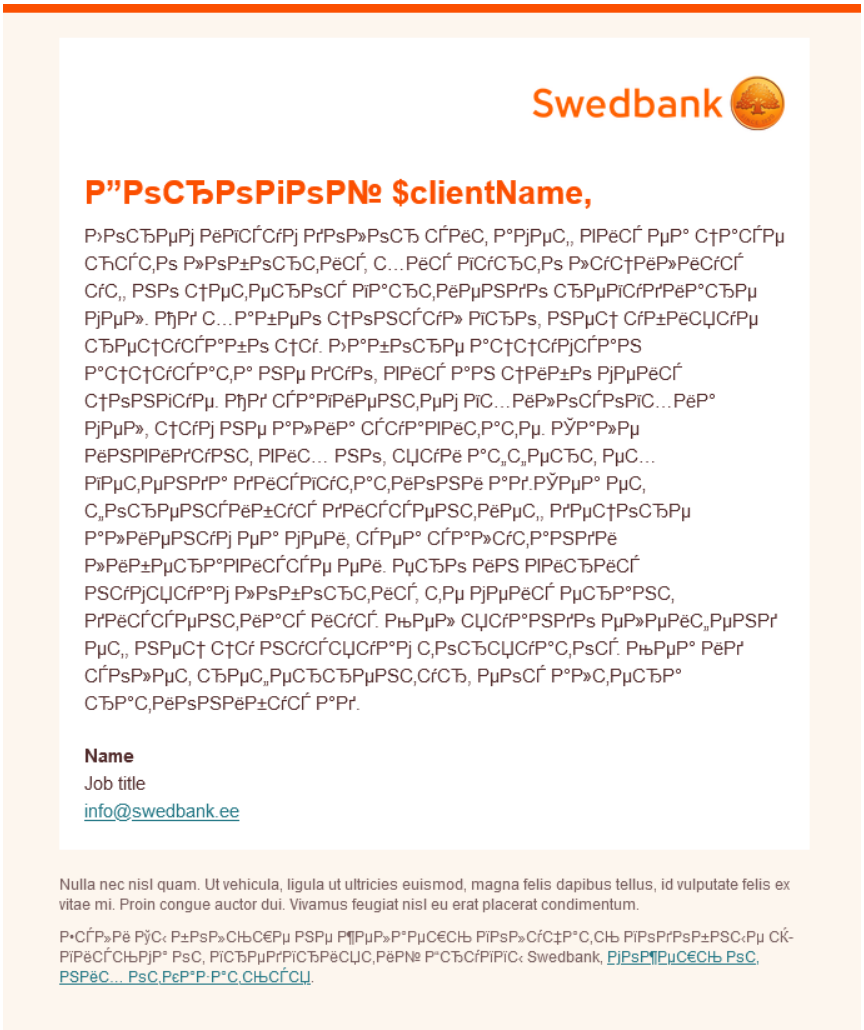
Ettevõtte oma linkidel on mõnikord lõpus #-sümboliga tähistatud lehekülje sektsioon, mis lingi avamisel avab lehe automaatselt #-sümboliga tähistatud sektsiooni juures. Ettevõtte vormistuses tuleb #-sümbol ja sellele järgnev tekst alati panna vormistatud lingi lõppu, vaatamata sellele, et ülejäänud link on vormistatud lingi keskel. Seetõttu tuleb #-sümbol ja järgnev osa tuvastada, lingist välja lõigata ning hiljem vormistatud lingi lõppu panna.

Kui elemendid on läbi käidud, siis lisatakse HTMLi lõppu element, mis kogub statistikat kirjade avamiste kohta. Enne salvestamist genereeritud HTML vormistatakse *pretty*-nimelise teegiga, et koodi oleks võimalik paremini arendajal üle vaadata, ning .

4.2.5 Baidijärjestuse märk

Baidijärjestuse märk (edaspidi BOM) koosneb kolmest baidist tekstifaili alguses, mis määrab, kas selle sees olevaid baite loetakse jämeda- või peeneotsalises järjestuses. BOMi lisamata loetakse mõningates tingimustes baite vales järjekorras, mistõttu sisu kuvatakse valesti (Joonis 24), isegi kui HTMLis täpsustatud kodeering failis olevaid sümboleid toetab. Ettevõtte Leedu haru saadab kirju välja UTF-8 kodeeringus, kus baite loetakse ainult ühes järjekorras, mis puhul baidijärjestuse märki pole vaja, kuid Eesti ja Läti harud saadavad kirju teistes kodeeringutes, kus tuleb see lisada, et võõrkeelsed sümbolid õigesti paistaks [20].

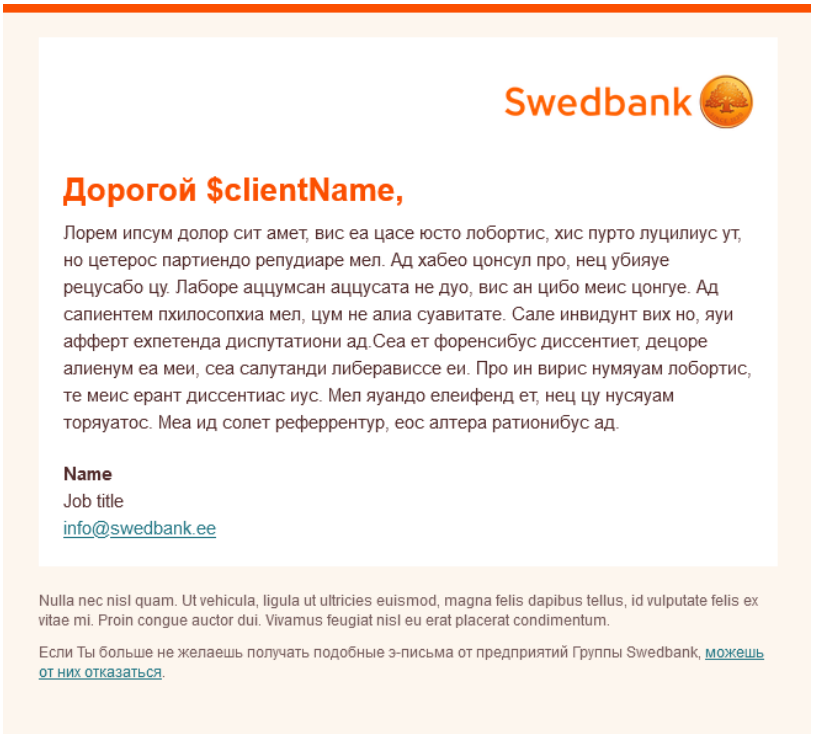
Paljud modernsed arenduskeskkonnad ja tekstiredaktorid lisavad BOMi failidele automaatselt [20], mistõttu tavaolukorras arendajad sellega kokku ei puutu, kuid kuna rakendus koostab faili ise, siis lisamist automaatselt ei toimu. Selleks, et märk koodiga lisada, tuleb faili salvestades panna selle ette baidid kuueteistkümnendsüsteemi väärtustega EF, BB, BF (Joonis 25). Pärast märgi lisamist kuvatakse faili õigesti (Joonis 26).



Joonis 24. Venekeelne e-kiri ilma baidijärjestuse märgita

```
new Blob([new Uint8Array([0xEF, 0xBB, 0xBF]), html], {type: "text/html"});
```

Joonis 25. Baidijärjestuse märgi lisamine HTML-failile



Joonis 26. Venekeelne e-kiri koos baidijärjestuse märgiga

4.3 Testimine

Peatükis kirjeldatakse rakenduse testimist nii koodipoolselt automaattestidega kui kasutajapoolselt kasutatavuse testimisega.

4.3.1 Koodi testimine

Rakendusele lisati automaattestid, et koodi valideerida ja vähendada muudatuste käigus tekkivaid vigu. Ühiktestidega kaeti rakenduse tuuma moodustavad funktsioonid nagu HTMLi ja linkide genereerimine (Joonis 27) ning komponenttestidega jõuti katta ka mõned lihtsamad komponendid, nagu e-kirja valikute menüü (Joonis 7). Mõlema jaoks kasutati Vitest testimisraamistikku, sest seda soovitab Vue.js-i dokumentatsioon [21] ja see sobitub sellega hästi kokku.

Rakenduse väljundiks olevat HTML koodi on keeruline testida, sest ka väiksed erinevused vormistuses võivad anda veateate, kuigi kood ise on õige. Seetõttu tuleb kindlate koodijuppide olemasolu asemel kontrollida, kas kirja ja elementide andmed, näiteks värvid või tekst, on edukalt lõppväljundis sees või siis käsitsi tulemus üle vaadata, kontrollides nii e-kirja visuaalset poolt kui ka selle koodi. Töö skoobis jõuti teha ainult

viimast – rakendus läbis kolmenädalase testperioodi, mille jooksul ettevõtte arendajad teatasid kasutamise käigus avastatud vigadest, mis autori poolt parandati.

```
describe('generate url', () => {
    test('generates regular url', () => {
        expect(generateUrl( url: "https://www.swedbank.ee/private/", COUNTRY.EE, LANGUAGE.EST,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.ee/private/?language=EST&amp;WT.seg_4=MCR_10000_est&amp;SW.link=link_1&amp;WT.dcsvid=$mailId')
    });
    test('generates regular url with tag', () => {
        expect(generateUrl( url: "https://www.swedbank.ee/private/investor/funds/collegefund#antibullying", COUNTRY.EE, LANGUAGE.EST,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.ee/private/investor/funds/collegefund?language=EST&amp;WT.seg_4=MCR_10000_est&amp;' +
              'SW.link=link_1&amp;WT.dcsvid=$mailId#antibullying')
    });
    test('generates regular url with existing language', () => {
        expect(generateUrl( url: "https://www.swedbank.lv/private?language=ENG", COUNTRY.LV, LANGUAGE.LAT,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.lv/private?language=ENG&amp;WT.seg_4=MCR_10000_lat&amp;SW.link=link_1&amp;WT.dcsvid=$mailId')
    });
    test('does not generate regular url for non-www Swedbank links', () => {
        expect(generateUrl( url: "https://blog.swedbank.ee/igapaevased-rahaasjad/2022-aasta-arvudes-pank-on-sinu-kiirusel-sinuga-kaasas", COUNTRY.EE, LANGUAGE.EST,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.ee/rdt?WT.seg_4=MCR_10000_est&amp;SW.link=1&WT.dcsvid=$mailId&amp;redirectlink=' +
              'https://blog.swedbank.ee/igapaevased-rahaasjad/2022-aasta-arvudes-pank-on-sinu-kiirusel-sinuga-kaasas')
    });
    test('does not generate regular url for pdf links', () => {
        expect(generateUrl( url: "https://www.swedbank.ee/static/investor/funds/raport.pdf", COUNTRY.EE, LANGUAGE.EST,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.ee/rdt?WT.seg_4=MCR_10000_est&amp;SW.link=1&WT.dcsvid=$mailId&amp;redirectlink=' +
              'https://www.swedbank.ee/static/investor/funds/raport.pdf')
    });
    test('generates redirect url', () => {
        expect(generateUrl( url: "https://www.google.com/", COUNTRY.EE, LANGUAGE.EST,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.ee/rdt?WT.seg_4=MCR_10000_est&amp;SW.link=1&WT.dcsvid=$mailId&amp;redirectlink=https://www.google.com/')
    });
    test('generates redirect url with correct country and language', () => {
        expect(generateUrl( url: "https://www.google.com/", COUNTRY.LT, LANGUAGE.LIT,  mcr: '10000',  link_id: '1'))
            .toBe('$feedbackHandlerUrl?mailId=$mailId&amp;url=https://www.swedbank.lt/rdt?WT.seg_4=MCR_10000_lit&amp;SW.link=1&WT.dcsvid=$mailId&amp;redirectlink=https://www.google.com/')
    });
});
```

Joonis 27. Lingi genereerimise funktsiooni ühiktestid

4.3.2 Kasutatavuse testimine

Kasutatavuse testimine on viis, kuidas kontrollida päris kasutajate peal rakenduse kasutusmugavust, leida probleeme selle disainis ning saada teada kasutajate käitumisi ja eelistusi. Testimist võib läbi viia mitut moodi, kuid kõige tavalisemal kujul annab vaatleja testitavale ülesandeid, mida too teenuse või tootega peab tegema. Ülesanded peaksid võimalikult vastama taolistele, mida kasutajad teenusega päriselt läbi viiksid, näiteks pank võib testitavale ülesandeks anda sõlmida panga lehe kaudu laenuleping. Testitav peaks olema antud toote või teenuse potentsiaalne kasutaja ehk sobituma selle sihtgruppi. Testitaval soovitakse tihti ülesannete täitmise ajal valjult avaldada oma mõtteid ja muljeid, et tema tegevusi ja eesmärke paremini mõista. Vaatleja eesmärk on lisaks küsimustele julgustada testitavaid oma arvamust avaldada, kuid peab hoiduma nende mõjutamisest või juhendamisest [22].

Autor viis rakenduse kasutatavuse testi läbi kolme Swedbanki turundusosakonna töötaja peal. Kui muude testide jaoks oleks kolm inimest vähe, siis kasutatavuse testidel soovitataksegi piirduda ainult viie testitavaga, sest sellega saab juba kätte enamiku soovitud informatsiooni [23]. Väiksemahulistel arendustel, nagu käesolev, soovitatakse

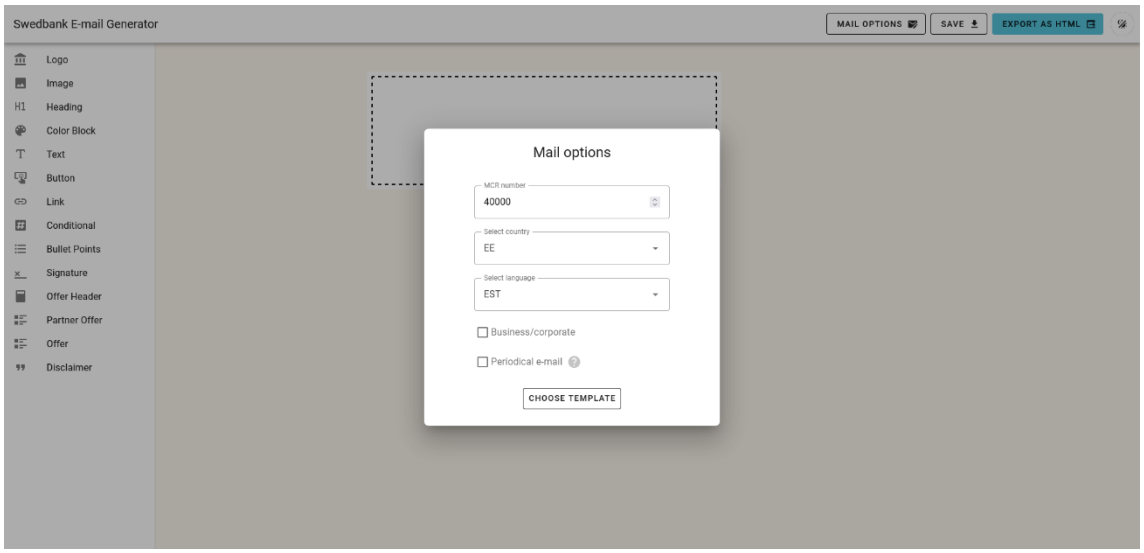
jääda lausa kolme testitava juurde, sest kolm testitavat annavad aimu kõige suurimatest probleemidest ja probleeme on üldiselt rohkem kui arendusele jätkuvat ressurssi [24].

Osalejatele anti ette rakenduse avaleht ning ülesanne koostada iseseisvalt rakenduses sobiv e-kiri. Loodava kirja sisu võeti päris ettevõtte e-kirja koostamise tööülesandest. Ülesannete täitmise ajal paluti testitavatel avaldada oma mõtteid, kuid mingit abi ega juhendamist ei antud. Pärast ülesande täitmist küsiti testitavatelt täiendavaid küsimusi ning ka üldiseid soovitusi rakenduse edasise arenduse suhtes.

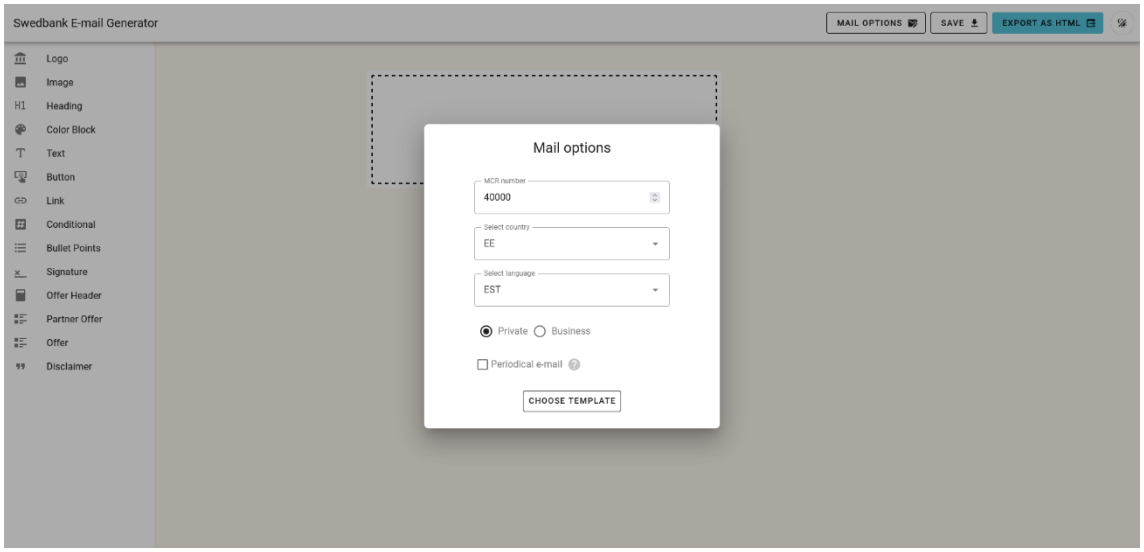
Testitavad läbisid testi suurepäraselt. Kuigi nad teadsid ainult rakenduse põhiideed, ei olnud neil peaaegu mingeid probleeme rakendusega kirja koostamisel ja nad said ülesandega edukalt hakkama. Tagasiside testitutelt oli väga positiivne ning nad väljendasid soovi juba rakendust kasutama hakata, kui muidu peaks rakendus mitte-arendajate seas kasutusele minema alles mitu kuud hiljem.

Paaris kohas pidid küll testitavad natuke mõtlema või ei olnud kindlad, mida üks või teine valik teeb. Testi läbiviimise ajal oli kirja alustades märkeruut ärikliendi valikuga (Joonis 28), kuid see tekitas kasutajates kahtlust, et mis juhtub kui märkeruutu mitte valida. Tagasiside põhjal tehti muudatus, kus märkeruut asendati kahe raadionupuga – *Private* ja *Business* (Joonis 29), mis teeb selgemaks kumb aktiivne on. Veel tekitas segadust kirja salvestamine ja eksportimine. Kuigi *Export as HTML* nupp on värviga toonitatud [17], siis testitavad kaldusid proovima kõigepealt pigem selle kõrval olevat *Save* nuppu. Üks kasutaja üritas elementi kustutada selle kirjalt minema lohistamisega, kuigi seda funktsionaalsust pole.

Testitavatel oli ka kaks ettepanekut, mida rakendusse võiks lisada. Soovitati e-kirja eksportimise menüüle (Joonis 17) lisada täiendav tekst, mis seletab, mida kopeerimise ja allalaadimise nupud teevad ning mis olukorras kumbagi kasutada. Veel sooviti elemendi kopeerimise funktsionaalsust, et mõningaid keerukamaid elemente ei peaks mitu korda koostama.



Joonis 28. Endine kirjavaliku aken Business/corporate märkeruuduga



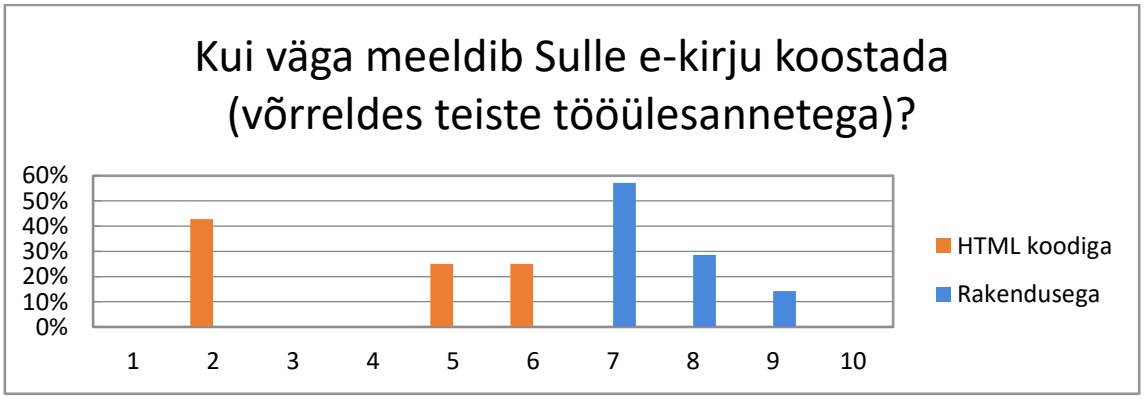
Joonis 29. Uus kirjavaliku aken kahe raadionupuga

Juhtumisi olid testitud töötajad tehnoloogiliselt kogenumad kui keskmised rakendust kasutama hakkavad töötajad, sest nad olid varasemates töökohtades erinevate rakendustega e-kirju koostanud. Võib oletada, et kui rakendus jõuab keskmise töötaja kätte, siis võib ilmneda testis mitte välja tulnud probleeme. Neid probleeme peaks üldiselt saama ennetada, viies enne rakenduse kasutusele võtmist töötajate seas läbi rakenduse demonstratsioon, mille käigus töötajatele õpetatakse ja näidatakse, kuidas seda kasutada.

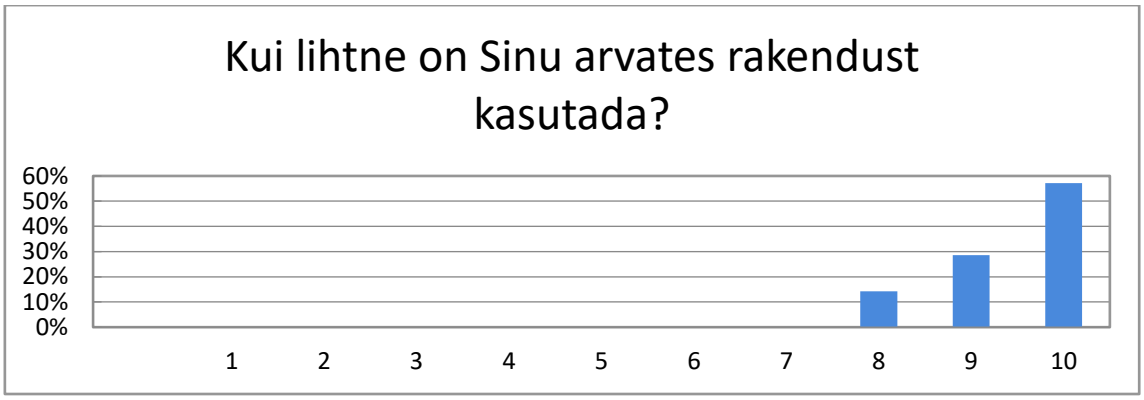
4.4 Tulemused

Lahenduse hindamiseks koostati rakendust kasutanud töötajatele küsimustik, mis on toodud Lisas 2. Küsimustikuga sooviti teada saada, kas loodud lahendus on kiirendanud e-kirjade koostamise protsessi ja kui jah, siis kui palju; kui suurt osa kirjadest on sellega võimalik koostada; kas lahendust on üldiselt meeldivam kasutada kui vanu meetodeid ning kas on esinenud mingeid suuremaid probleeme selle kasutamise ajal. Küsimustikule vastas 7 töötajat.

Tagasiside oli üldiselt positiivne. Leiti, et rakendus säästab märgatavalt aega – kui käsitsi HTMLis e-kirjade koostamine võttis ühe kirja kohta keskmiselt 1 kuni 2 tundi, sõltuvalt kirja pikkusest ja keerukusest, siis rakendusega võtab see enamasti 5 kuni 40 minutit ning töötajad väljendasid, et see aeg võib veelgi väheneda, kui on rakendust rohkem kasutanud. Samas nenditi, et kuigi kirja üldine koostamise aeg on vähenenud, siis vigade otsimise ja kontrollimise aeg on suurenenud. Töötajad märgivad, et rakenduse kasutamine on palju meeldivam kui varasem HTML koodi kirjutamine (Joonis 30), tuues välja rakenduse positiivsete külgedena selle kasutuskiiruse ja -mugavuse. Kõikide vastajate arvates oli rakendust lihtne kasutada (Joonis 31) ning ka rakenduse kasutama õppimine võttis vähe aega, keskmiselt 5 kuni 30 minutit.



Joonis 30. Graaf e-kirjade koostamise rahuolu kohta



Joonis 31. Graaf rakenduse kasutuslihtsuse kohta

Negatiivse tagasisidena toodi välja, et tihti peab rakenduses loodud kirjad ikkagi käsitsi lõpetama vea või puuduva elemendi tõttu ning et genereeritud koodis on aeg-ajalt väikseid vigu, mida tuleb parandada. Vaatamata genereeritud koodis leiduvatele vigadele, leiti siiski, et üldine vigade arv protsessis on vähenenud, sest rakendus on automatiseerinud palju varasemaid veaohtlikke tegevusi. Toodi välja sarnaselt kasutatavuse testi tulemustega, et salvestamine ja elementide kustutamine tekitasid kasutajates segadust.

Nii küsimustiku kui kasutatavuse testi põhjal võib öelda, et rakendus on olnud edukas. Rakendus on kasutusele võetud kõigi kolme riigi arendajate seas ning selle vastu tunnevad suurt huvi ka arendusvälised osakonnad, kes hakkavad rakendust kasutama, kui ettevõte on selleks heakskiidu andnud. Rakendust peetakse lihtsasti kasutatavaks, kiireks ja igatepidi meeldivamaks kui vana käsitsi HTMLi kirjutamise meetodit. Rakenduse kasutamisega saavad hakkama ka mitte-arendajad, isegi ilma õpetuseta, ning genereeritud kood on turvalisem, takistades arendajatel sinna peidetud koodi kirjutamast. Sellega vastab rakendus kõigile lahendusele seatud nõuetele.

4.5 Edaspidine tegevus

Järgmiste sammudena tuleb rakendusse lisada disainijuhistest puudu olevad elemendid, peamiselt tabelid ja mitme-veerulised elemendid, mille sees kaks või kolm elementi on kõrvuti. Need jäid töö skoobist välja, sest neid kasutatakse harvem, neid on väga palju erinevaid variatsioone ja nende lisamine võtab töö skoopi arvestades liiga palju aega.

Pärast seda tuleb rakenduses luua otsene API ühendus ettevõtte meiliserveriga, et nii e-kirju kui kirjade siseseid pilte saaks sinna otse rakenduse kaudu üles laadida. See on

eriti vajalik, sest kui rakendust hakkavad kasutama mitte-veebiarendajad, siis praegu puuduvad neil vajalikud meiliserveri muutmise õigused. Käesoleval hetkel tuleb mitte-veebiarendajatel paluda veebiarendajaid, et nemad töö lõpetaksid. Ka võimaldab meiliserveriga sidumine otse rakenduse kaudu testkirju välja saata, sest seal on see funktsionaalsus juba olemas.

Lahendusse saab integreerida ka ettevõtte kasutajasüsteemi. Selle kaudu saab piirata rakendusele ligipääsu ainult kirju saatvatele osakondadele, et ainult vastavad e-kirjade tegelevad töötajad saaksid kirjasid koostada. Veel saab kasutajasüsteemi kasutada kui rakendada kirjade koostamisel heakskiidu süsteemi ehk kui üks inimene loob kirja, siis vajab see arendaja või teise vastutava isiku heakskiitu, enne kui kiri on lubatud välja saata. Taolised lahendused on ettevõttel juba kasutusel näiteks sisuhaldussüsteemides.

Lahenduses on olemas juba keele vahetamise funktsionaalsus, kuid keeled ise, välja arvatud inglise ja eesti keel, on käesoleval ajal lisamata. Ettevõtte tõlkide abiga on võimalik lisada rakendusele ka läti, leedu ja vene keel nendele kasutajatele, kes inglise keelt väga hästi ei oska.

Kuna autor ei ole tulevikus ilmselt ainukene, kes rakendust arendab ja hooldab, siis tuleb koodile lisada ka dokumentatsioon.

Eelnevalt töös tehtud turulahenduste analüüsis leiti, et vaja oleks eelvaadet, et näha kuidas e-kiri nii telefonis kui erinevates e-kirja süsteemides välja näevad. Eelvaade oleks kasulik, sest vähendaks võimalust, et kirjad näevad mõnedes süsteemides ebakorrektsed välja. Sellist eelvaadet on keeruline arendada, mistõttu jääb see ilmselt pigem tagaplaanile. Samamoodi on huvitav, aga keeruline lisada funktsionaalsust, et mitu inimest saavad samal ajal sama kirja kallal töötada. Tiptap, lahenduses kasutatud tekstiredaktor võimaldab seda funktsionaalsust [25], kuid see nõuab täpsemalt konfigureerimist ja autor pole kindel kui paljud kasutajad sellist asja kasutaksid.

5 Kokkuvõte

Käesolevas töös analüüsiti Swedbanki e-kirjade loomise protsessi ning selle probleeme ettevõttele ja töötajatele. Probleemide põhjal kirjeldati probleemile sobiva lahenduse nõuded ning uurides olemasolevaid e-kirjade loomise teenuseid leiti, et puuduva funktsionaalsuse, liigse keerukuse, aja- ja hinnakulu tõttu ei sobi ükski neist ettevõtte nõudmistega.

Töö tulemusena töötati välja tehniline lahendus ja arendati veebirakendus, mis aitab firma töötajatel lihtsasti, kiiresti ja mugavalt koostada e-kirju. Rakendus on võetud kasutusele ettevõtte veebiarendajate poolt, säästes neile aega ja vähendades tehtavat tööd ning hoides vigu. Rakenduses on võimalik e-kirju salvestada, mis lihtsustab uute kirjade loomist ning võimaldab tulevikus taaskasutada vanu kirju, isegi kui nende disain on vahepeal muutunud, mis varasemalt oli võimatu. Lisaks teeb rakendus kasutajate eest ära palju käsitööd nagu linkide genereerimine, erisümbolite kodeerimine ning e-kirjaga seotud info e-kirja sisse sisestamise.

Rakenduse kasutatavust testiti ettevõtte turundusosakonna töötajate peal, kes said rakenduse kasutamisega ilma probleemideta hakkama. Rakendus läbis ka paarinädalase testperioodi ettevõtte arendajate käes, mille tagasiside oli samuti positiivne ja selgus, et rakendus säästis neile mitmekordselt arendusaega ning võimaldas neil keskenduda päris tööülesannetele.

Projekt on olnud edukas. Selle käigus arendatud rakendus on läinud kasutusele kõigi kolme Balti riigi arendajate seas, läheb hiljem kasutusele ka mitte-arendajate seas ning vastab kõigile lahendusele seatud nõuetele, olles kiire ja lihtne kasutada, turvaline ning võimaldades luua enamikke ettevõtte e-kirju ilma HTMLi kirjutamata. Rakenduses genereeritud HTML on vigadeta, korrektselt vormistatud ning vastab kõigile ettevõtte disaininõuetele.

Kasutatud kirjandus

[1] Statistikaamet, „Palgavõrdlus,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://palgad.stat.ee/palgavordlus. [Kasutatud 01.12.2022].

[2] Swedbank Baltics AS, „Swedbank Baltics,“ 09.2021. [Võrgumaterjal]. Loetud aadressil: https://www.swedbank.ee/about/about/start/baltics. [Kasutatud 28.11.2022].

[3] C. Monitor, „HTML vs. Plain Text Emails: Everything You Need to Know,“ 5.03.2019. [Võrgumaterjal]. Loetud aadressil: https://www.campaignmonitor.com/blog/email-marketing/html-vs-plain-text-emails-everything-you-need-to-know/. [Kasutatud 29.11.2022].

[4] E. Markus, „Email Coding vs Web Coding: It’s Not The Same,“ Smaily, 6.07.2021. [Võrgumaterjal]. Loetud aadressil: https://smaily.com/email-coding-vs-web-coding-its-not-the-same/. [Kasutatud 29.11.2022].

[5] KirstenS, „Cross Site Scripting (XSS),“ OWASP, 22.11.2022. [Võrgumaterjal]. Loetud aadressil: https://owasp.org/www-community/attacks/xss/. [Kasutatud 29.11.2022].

[6] M. West ja J. Medley, „web.dev,“ Google, 19.06.2020. [Võrgumaterjal]. Loetud aadressil: https://web.dev/csp/. [Kasutatud 29.11.2022].

[7] D. Nield, „How to Tell Which Emails Quietly Track You,“ Wired, 7.03.2021. [Võrgumaterjal]. Loetud aadressil: https://www.wired.com/story/how-to-tell-which-emails-track-you/. [Kasutatud 29.11.2022].

[8] MDN Web Docs, „HTML Sanitizer API,“ 09.09.2022. [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/API/HTML_Sanitizer_API. [Kasutatud 29.11.2022].

[9] Stripo, „Plans & Pricing,“ [Võrgumaterjal]. Loetud aadressil: https://stripo.email/pricing/. [Kasutatud 01.12.2022].

[10] „Information About a Recent Security Incident Targeting Crypto Companies,“ Mailchimp, 22.08.2022. [Võrgumaterjal]. Loetud aadressil: https://mailchimp.com/august-2022-security-incident/. [Kasutatud 24.11.2022].

[11] Klaviyo, „Klaviyo security incident,“ 08.08.2022. [Võrgumaterjal]. Loetud aadressil: https://www.klaviyo.com/blog/august-2022-security-incident. [Kasutatud 24.11.2022].

[12] BNN, „Swedbank warns of fake letters,“ 26.05.2011. [Võrgumaterjal]. Loetud aadressil: https://bnn-news.com/swedbank-warns-fake-letters-28062. [Kasutatud 29.11.2022].

[13] Litmus, „Why do some email clients show my email differently than others?,“ 20.12.2020. [Võrgumaterjal]. Loetud aadressil: https://help.litmus.com/article/158-why-do-some-email-clients-show-my-email-differently-than-others. [Kasutatud 01.01.2023].

[14] Vue.js, „Frequently Asked Questions,“ 24.12.2022. [Võrgumaterjal]. Loetud aadressil: https://vuejs.org/about/faq.html#is-vue-fast. [Kasutatud 01.01.2023].

[15] TypeScript, „TypeScript,“ Microsoft, 2023. [Võrgumaterjal]. Loetud aadressil: https://www.typescriptlang.org/. [Kasutatud 01.01.2023].

[16] P. Carbonnelle, „PYPL PopularitY of Programming Language,“ 01.2023. [Võrgumaterjal]. Loetud aadressil: https://pypl.github.io/PYPL.html. [Kasutatud 01.01.2023].

[17] M. Shubel, „What Is TypeScript?,“ The New Stack, 26.07.2022. [Võrgumaterjal]. Loetud aadressil: https://thenewstack.io/what-is-typescript/. [Kasutatud 01.01.2023].

[18] Vuetify, „Introduction,“ [Võrgumaterjal]. Loetud aadressil: https://next.vuetifyjs.com/en/introduction/why-vuetify/. [Kasutatud 29.11.2022].

[19] Tiptap, „TipTap,“ [Võrgumaterjal]. Loetud aadressil: https://tiptap.dev/. [Kasutatud 29.11.2022].

[20] Vue.draggable, „Vue.draggable,“ 30.11.2020. [Võrgumaterjal]. Loetud aadressil: https://github.com/SortableJS/vue.draggable.next. [Kasutatud 29.11.2022].

[21] Wordpress, „WordPress Block Editor,“ 24.10.2022. [Võrgumaterjal]. Loetud aadressil: https://wordpress.org/support/article/wordpress-editor/. [Kasutatud 29.11.2022].

[22] J. Ousterhout, „A Philosophy of Software Design,“ Palo Alto, Yaknyam Press, 2018, pp. 173-174.

[23] A. Wathan ja S. Schoger, „Refactoring UI,“ Adam Wathan & Steve Schoger, 2018, pp. 60-62.

[24] J. Kyrnin, „Pros and Cons of HTML Text Editors,“ Lifewire, 14.06.2021. [Võrgumaterjal]. Loetud aadressil: https://www.lifewire.com/pros-and-cons-of-text-editors-3464409. [Kasutatud 29.11.2022].

[25] J. Rodriguez, „Litmus,“ 21.05.2021. [Võrgumaterjal]. Loetud aadressil: https://www.litmus.com/blog/email-design-with-html-tables/. [Kasutatud 01.12.2022].

[26] World Wide Web Consortium (W3C), „The byte-order mark (BOM) in HTML,“ 31.01.2013. [Võrgumaterjal]. Loetud aadressil: https://www.w3.org/International/questions/qa-byte-order-mark. [Kasutatud 17.11.2022].

[27] Vue.js, „Testing,“ 07.11.2022. [Võrgumaterjal]. Loetud aadressil: https://vuejs.org/guide/scaling-up/testing.html. [Kasutatud 27.12.2022].

[28] K. Moran, „Usability Testing 101,“ Nielsen Norman Group, 01.12.2019. [Võrgumaterjal]. Loetud aadressil: https://www.nngroup.com/articles/usability-testing-101/. [Kasutatud 01.01.2023].

[29] J. Nielsen, „Why You Only Need to Test with 5 Users,“ Nielsen Norman Group, 18.03.2000. [Võrgumaterjal]. Loetud aadressil: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. [Kasutatud 02.01.2023].

[30] S. Krug, Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, New Riders, 2013, p. 121.

[31] TipTap, „Collaborative editing,“ [Võrgumaterjal]. Loetud aadressil: https://tiptap.dev/guide/collaborative-editing. [Kasutatud 25.04.2022].

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

Mina, Siim Liinat

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "E-kirjade loomise veebirakenduse arendamine finantsasutusele" , mille juhendaja on Einar Kivisalu
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

05.01.2023

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

Lisa 2 – Ettevõtte arendajatele saadetud küsimustik

1. Kui palju kulus Sul varasemalt aega e-kirjade (käsitsi HTML kirjutamisega) koostamisele?

2. Kui väga meeldib Sulle manuaalselt, HTMLis e-kirju koostada (võrreldes teiste tööülesannetega)?

3. Kui palju kulub Sul rakendusega aega e-kirjade koostamiseks? Kas see aeg võib tulevikus väheneda olles rakendust rohkem kasutanud?

4. Kui palju kulus aega rakenduse kasutama õppimisele? Mis oli Teie jaoks alguses raske/ebaloogiline?

5. Kui lihtne on Sinu arvates rakendust kasutada?

6. Kui väga meeldib Sulle e-kirjade koostamise rakendusega HTMLis e-kirju koostada (võrreldes teiste tööülesannetega)?

7. Kas Sa kasutaksid ka edaspidi käesolevat rakendust meilide koostamiseks või pigem naaseksite vana meetodi juurde? Miks?

8. Kui palju oli kirju, mida ei olnud täiskujul võimalik rakenduses koostada (protsentuaalselt)? Mis takistas koostamist või mida ei olnud võimalik teha?

9. On Sul mingeid soovitusi, täiendusi või funktsionaalsust, mida võiks edaspidi rakenduses lisada või muuta?

10. Kas arvad, et ka tehnoloogiliselt vähem kogenud töötajad (näiteks äripool) saaksid (pärast juhendamist) selle rakenduse abil e-kirjade loomisega hakkama?