TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Carl-Robert Reidolf 179484
Triin Tammaru 179731
Lilian Väli 179808

AS SEB PANK EESTI RAHAPESUKAHTLASE TEGEVUSE KOHTA TEATISE EDASTAMISE RAKENDUSE TÄIENDAMINE

Bakalaureusetöö

Juhendaja: Tõnn Talpsepp
Doktorikraad

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: Triin Tammaru

17.05.2020

Annotatsioon

AS SEB Panga äriarenduse initsiatiivil koostati projekt, mille eesmärgiks oli arendada rahapesu andmebüroo rahapesukahtluse teatise veebivormi põhjal AS SEB Panga sisene teatiste edastamise rakendus. Arendustöö valmis kahes etapis. Esimeses etapis valminud rakendus täitis AS SEB Panga vastavuskontrolli poolt seatud põhikriteeriumid, kuid ei lahendanud nõudeid teatise koostamise kiiruse, efektiivsuse ja kasutajasõbralikkuse osas. Antud lõputööga valmis projekti teine etapp, mille eesmärgiks oli tulenevalt eelnimetatust automatiseerida teatise koostamist, parendada esimeses etapis tehtud arendust ning täiendada rakendust vajalike lisadega.

Lõputöö tulemusena valmis AS SEB Panga sisene automatiseeritud rakendus rahapesukahtlaste tehingute edastamiseks rahapesu andmebüroole läbi X-tee. Rakendusele arendati juurde võimalus lisada erinevaid faile, genereerida täidetud teatisest PDF-dokument ning laadida alla teatise raport. Samuti sai töö tulemusena otsida AS SEB Panga kliente ning nendega seotud tehinguid otse panga andmebaasidest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 66 leheküljel, 5 peatükki, 38 joonist, 16 tabelit.

Abstract

Improvements of the Application for Submission of Reports about Suspicion of Money Laundering for AS SEB Bank

Estonian Financial Intelligence Unit (FIU) is an independent structural unit of the Estonian Police and Border Guard Board. The main purpose of FIU is to prevent money laundering and financing of terrorism in Estonia. Anyone who suspects that a transaction may be connected with money laundering or terrorism financing may submit a notification of suspicious transactions via their web-based application. The same online version was also used by the Compliance department of AS SEB Bank. For them, the FIU's form had several shortcomings.

On the initiative of AS SEB Bank's business development, a project was prepared. The aim of said project was to develop AS SEB Bank's internal application based on FIU's existing web form. The project was completed in two stages. The first stage met the main criteria set by AS SEB Bank's Compliance department, but did not address the requirements regarding speed, efficiency and usability. With this thesis, the second stage of the project was completed. The aim of this thesis was to automate the data filling process, add improvements and integrate necessary additional features to the development done in the first stage of the project.

As a result of this thesis, AS SEB Bank's internal automated application for submission of reports about suspicion of money laundering to FIU via X-Road was completed. The developments done include the ability to attach files to the form, generate a PDF document from a completed notification and download a report of the notification. Furthermore, it is now also possible to search the customers of AS SEB Bank and their transactions directly from the bank's databases.

The thesis is in Estonian and contains 66 pages of text, 5 chapters, 38 figures, 16 tables.

Lühendite ja mõistete sõnastik

ABL Scratchpad	*Editor*, millega saab koodi testida
Back-end	Loogika ja andmete salvestamine
CSV	*Comma Separated Values* failivorming
Curl	*Client Uniform Resource Locator*
Front-end	Kuvand, mida näeb kasutaja
IT	Infotehnoloogia
Java Bean	Objekt Java arenduskeeles
MVC	*Model View Controller*
NBSF	Tellerite töökeskkond AS SEB Pangas
NBSX	Andmebaas ja *back-end* AS SEB Pangas
OpenEdge ABL	OpenEdge Advanced Business Language
PDF	*Portable Document Format*
RAB	Rahapesu andmebüroo
Setup	Ärikriitilise informatsiooni andmebaasitabel
UI	*User Interface* ehk kasutajaliides
UX	*User Experience* ehk kasutajakogemus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Rahapesu andmebüroo (edaspidi RAB) on Politsei- ja Piirivalveameti iseseisev struktuuriüksus, mille põhiülesandeks on rahapesu ja terrorismi rahastamise tõkestamine Eestis. Kõik isikud, kellel tekib majandustehingu teostamise käigus kahtlus, et tehing võib olla seotud rahapesu või terrorismi rahastamisega, saavad esitada teate RAB-i veebipõhise rakenduse kaudu. Sama veebiversiooni kasutasid teadete esitamiseks ka AS SEB Panga *Compliance* ehk vastavuskontrolli töötajad, kuid nende jaoks esines RAB-i vormil mitmeid puudusi. (Politsei- ja piirivalveameti kodulehekülg, 2018)

Tulenevalt vastavuskontrolli töötajate ettekirjutustest otsustati AS SEB Panga äriarenduse initsiatiivil koostada projekt ning arendada AS SEB Panga sisene teatiste edastamise rakendus. Projekt valmis Tallinna Tehnikaülikooli tudengite Lilian Väli, Triin Tammaru, Marie Elise Soomre ning Carl-Robert Reidolfi ja AS SEB Panga koostöös kahes etapis, meeskonnaprojekti ja lõputööna. Esimeses etapis arendasid tudengid valmis rakenduse põhja, kus oli võimalik algatada teatise koostamist, täita see käsitsi ning edastada RAB-ile. Esimese etapi arendustegevus dokumenteeriti ning kaitsti meeskonnaprojekti aine läbimiseks. Teise etapi jooksul teostati rahapesu kahtlusega teatiste edastamise rakendusele edasiarendused ning lisati projekti esimese etapi jooksul tekkinud ideede põhjal täiendused.

Meeskonnaprojekti käigus valminud rakendus täitis projekti alguses AS SEB Panga vastavuskontrolli poolt seatud põhikriteeriumid, milleks olid võimalus pangasiseses süsteemis teatiseid koostada, täita ning RAB-ile saata. Need aga ei lahendanud kogu projekti koostamisel püstitatud nõudeid teatise koostamise kiiruse, efektiivsuse ja kasutajasõbralikkuse osas. Esimeses etapis valminud arenduse peamised kitsaskohad olid teatise väljade vähene eeltäidetus ning failide lisamise võimaluse, teatisest kompaktse visuaalse ülevaate ja statistika puudumine. Seetõttu olid lõputöö eesmärgid automatiseerida teatise koostamist, parendada esimeses etapis tehtud arendust ning täiendada rakendust vajalike lisadega.

Projektil oli mitmeid funktsionaalseid ja mittefunktsionaalseid nõudeid. Neist tähtsaimad funktsionaalsed näiteks isikute ja tehingute andmebaasist otsimine ning nende põhjal väljade automaatne eeltäitmine. Lisaks, teatise põhjal genereeritud PDF dokumendi ja raporti alla laadimine. Vastavuskontrolli töötajate jaoks oli väga oluline aspekt see, et kellegi poolt kasutuses olevat teatist ei saaks teised samal ajal vaadata ning et teatised oleksid staatuste põhjal jaotatud erinevatesse tabelitesse. Olulised funktsionaalsed nõuded olid veel lisakontrollid teatise täitmisel ning RAB-i poolse vastuse kuvamine saadetud teatise andmete korrektsuse kohta.

Mittefunktsionaalsetest nõuetest põhilised olid kasutajasõbralikkus, äripoole otsustest tulenevate sisuliste muudatuste lihtne jõustamine ja turvalisus. Kogu süsteemi efektiivseks toimimiseks oli ka oluline minimeerida andmebaasile tehtavate päringute arvu.

Peamised kasutusjuhud:

1. Vastavuskontrolli töötajana soovin AS SEB Panga kliente otsida andmebaasist, et ei peaks juba olemasolevaid andmeid käsitsi täitma.
2. Vastavuskontrolli töötajana soovin, et võimalikult paljud väljad oleksid automaatselt eeltäidetud, et vormi koostamisele kuluks vähem aega.
3. Vastavuskontrolli töötajana soovin kliendi kontode tehinguid otsida andmebaasist, et ei peaks juba olemasolevaid andmeid käsitsi sisestama.
4. Pangana soovin näha statistikat rahapesu kahtlase tegevuse kohta, et tulevikus teadlikumalt tegeleda konkreetsete rahapesu suundadega ning sellist tegevust ennetada.
5. Vastavuskontrolli töötajana soovin valesti täidetud vormi korral näha veateadet ebakorrektsete andmeväljade kohta, et saaksin vigased väljad parandada.
6. Vastavuskontrolli töötajana soovin ma näha lisakontrolle väljade täitmisel, et mul ei ununeks olulised väljad.
7. Vastavuskontrolli töötajana soovin alla laadida teatise kohta PDF dokumenti, et mul oleks võimalik esitada teatisest kompaktne visuaalne ülevaade.
8. Vastavuskontrolli töötajana soovin alla laadida teatiste kohta raportit, et näha statistilist ülevaadet tehtud teatiste kohta.
9. Vastavuskontrolli töötajana soovin, et teised töötajad ei saaks minu kasutuses olevat teatist muuta, et teate koostamisel ei tekiks konflikti.

10. Pangana soovin, et AML teatise saatmise menüüpunktile oleksid seatud kasutajapiirangud, et teatist saaksid koostada vaid kvalifitseeritud ja volitatud töötajad.

Käesolev lõputöö annab detailse ülevaate sooritatud meeskondlikust projektist. Lõputöö koosneb viiest põhilisest osast: metoodika, töö tulemused, analüüs ja järeldused, kokkuvõte ning lisad.

Metoodika kirjelduses on välja toodud kasutatud tööriistad, tehnoloogiad ja meetodid ning kirjeldatud tööprotsessi. Samuti on antud projekti kohta täiendavat informatsiooni. Töö tulemuste all on detailselt lahti kirjutatud projekti tehniline dokumentatsioon. Analüüsi ja järelduste all on põhjendatud sügavuti töö tulemusi ja lisatud hinnang projekti teostuse ja protsessi kohta. Lisades on välja toodud meeskonnaliikme isiklikud panused ning eneseanalüüs.

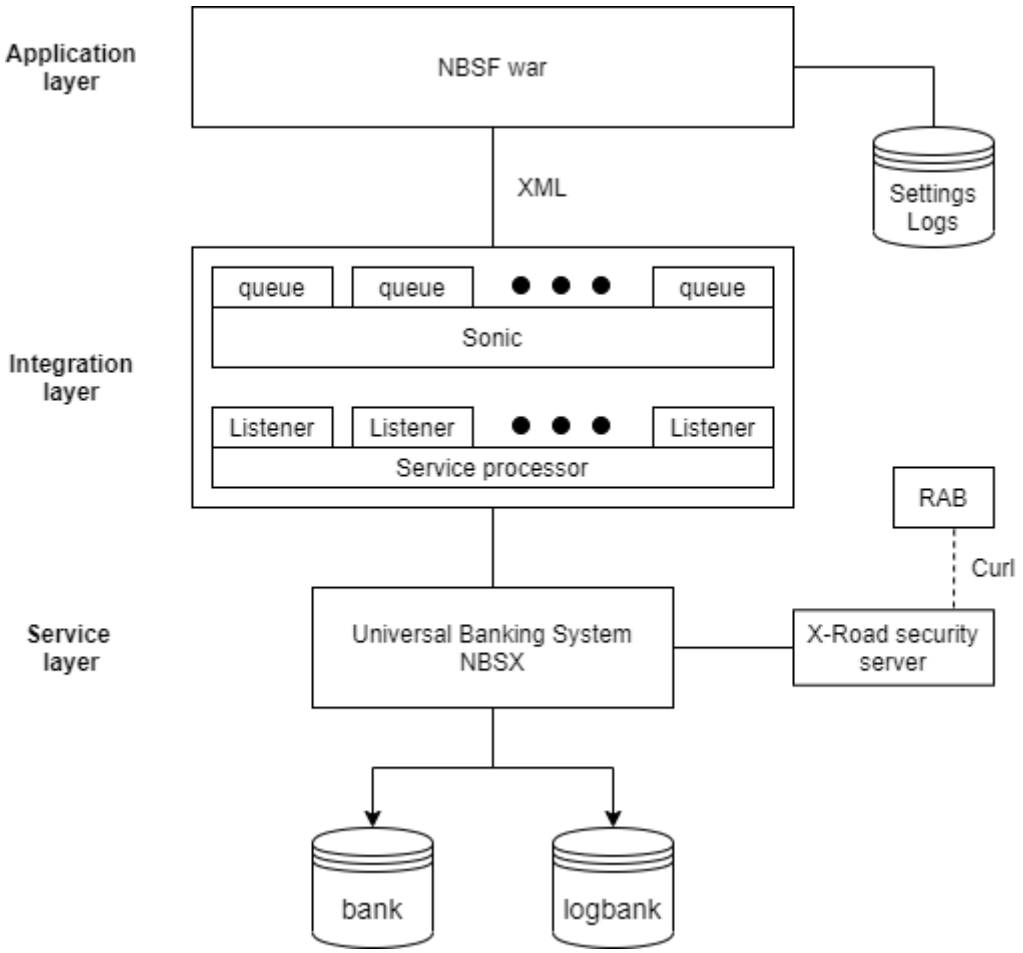
2 Metoodika

2.1 Objekti detailne kirjeldus

Peamiseks põhjuseks, miks AS SEB Pank soovis leida alternatiivi RAB-i teatise esitamise rakendusele oli see, et vastavuskontrolli töötajal võttis veebirakenduse kasutamine ning andmete pangasüsteemidest otsimine ja nende käsitsi sisestamine kaua aega. Arendus, mis valmis projekti esimeses etapis, eemaldas vajaduse kasutada RAB-i veebirakendust, kuid ei kiirendanud tööprotsessi olulisel määral. Projekti teise etapi sisuline eesmärk oligi automatiseerida teatise täitmise protsessi selliselt, et erinevad väljad eeltäidetakse panga andmebaasidest ja infosüsteemidest päritud andmetega klientide, tehingute ja muu vajaliku kohta.

Projekti esimeses etapis valmis meeskonnaprojektina veebirakendus, millega pank sai RAB-ile rahapesu kahtluse korral edastada teatise tehingu osapoolte, sisu, suuruse ja tausta kohta. Valminud veebirakendus oli oma olemuselt lihtne vorm, kuhu kasutaja saab sisestada vajalikud andmed, mida soovib RAB-ile edastada. Süsteemis oli menüüpunkt, mille alt sai näha koostatud, pooleliolevaid ning saadetud teatiseid, koostada uut teatist ning nupuvajutusega see X-tee vahendusel RAB-ile saata. Vormil olid eeltäidetud vaid saatja ehk teatise koostanud vastavuskontrolli töötaja ja AS SEB Panga andmed. Lisaks sai alustatud ja saadetud teatiseid näha teatiste tabelis.

Valminud rakendus integreeriti AS SEB Panga infosüsteemi nimega NBSF, mida kasutavad vastavuskontrolli töötajad ning ka tellerid, haldurid ja nõustajad. NBSF on pangasisene front-end rakendus ning andmed ja vajalikud sisendid saab see erinevatest andmebaasidest ning ka kesksüsteemist NBSX (Joonis 1). Selle arendamiseks kasutatakse Eclipse-i tarkvara ning Java arenduskeelt. NBSFi lisati esimesi etapi käigus uus menüüpunkt, mille alt on võimalik näha kõiki saatmata, saadetud ning pooleliolevaid teatiseid, koostada uusi teatiseid ning neid RAB-ile X-tee vahendusel saata.



Joonis 1. Süsteemi skeem.

Teatise salvestamiseks oli vaja luua ühendus ka NBSXiga, mis on AS SEB Panga kesksüsteem. Erinevate NBSXi protseduuridega töödeldakse panga klientide, dokumentide, lepingute ning maksete andmeid ning vajadusel edastatakse need teistele süsteemidele. NBSXi kasutatakse ka erinevate protsesside haldamiseks, näiteks maksete peatamiseks ja teostamiseks, klientide otsimiseks ja failide loomiseks. Selle arendamiseks kasutatakse OpenEdge platvormi ning OpenEdge ABL arenduskeelt.

NBSX teostab päringuid erinevatele andmebaasidele, mida on AS SEB Pangas kokku ligemale 15. Suuremates neist on ligikaudu 500 tabelit. Tehtud arenduse jaoks olid esmatähtsad kaks andmebaasi - logbank ning bank. Need on andmebaasid, kus hoitakse informatsiooni klientide, tehingute ja töötajate kohta.

2.2 Tööriistade kirjeldus

Lõputöö arendamiseks kasutati erinevaid programmeerimiskeeli ja tarkvaraarenduse platvorme. *Back-end*'i kirjutamisel rakendati protseduurilist programmeerimist. Selle tehniliseks teostamiseks kasutati Progress OpenEdge platvormi, OpenEdge ABL ehk Progress programmeerimiskeelt ja skriptide jooksutamiseks PuTTY tarkvara. Progress programmeerimiskeele süntaks on üsna lähedane inglise keelele ja võimaldab väga lihtsalt teha päringuid andmebaasile, sinna andmeid salvestada ning neid kustutada.

Front-end'i kirjutamisel lähtuti objektorienteeritud paradigmast. Seda kirjutati Eclipse platvormil Java arenduskeeles ning graafilise kasutajaliidese konstrueerimiseks kasutati ZK raamistikku, kus struktuur luuakse ZUL-failides XHTML keeles. AS SEB Pangas on XHTML kohandatud vastavalt panga arendustegevuse vajadustele. Java objektile vastaval XHTML-failil on välja toodud need elemendid, mida soovitakse vormil kuvada. Kui HTML-faili puhul luuakse kujundus CSS-failis, siis AS SEB Panga XHTML-fail seda ei võimalda. Saab vaid lisada välju, tabeleid ja nuppe vaikimisi kujundusega. See, kas välja kohal on rippmenüü, kuupäeva valik, *checkbox* või midagi muud, tuleb valida NBSFi kasutajaliideses vastavat tööriista kasutades. Kogu funktsionaalsus on koos ühes kontrolleris, kus väärtustatakse meetodites Java objekte ja nende elemente ning kutsutakse välja Progressi teenuseid näiteks teate salvestamiseks ja kustutamiseks. Java arenduse lokaalseks jooksutamiseks kasutatakse Apache Tomcat lokaalset veebiserverit.

Koodimuudatused laaditi üles Giti, kuid vigade haldamiseks ja koodi laadimisprotsessi jälgimiseks kasutati Jenkins tarkvara. Jenkins on pideva integreerimisega ja avatud lähtekoodiga automatiseerimisserver, mis aitab automatiseerida tarkvara arendamise protsessi seda osa, mis arendajast ei sõltu. See on serveripõhine süsteem, mis töötab servleti konteinerites, näiteks Apache Tomcatis. See toetab versioonikontrolli tööriistu ning suudab käivitada erinevatel automatiseerimistarkvaradel (nt Apache Ant ja Apache Maven) põhinevaid projekte, samuti erinevaid keskskripte ja Windowsi pakettkäske. Kui Progressis või Eclipse'is programmeeriti juurde uut loogikat, siis seda, kas koodimuudatus jõudis ka serveritesse, kas esines vigu ning milline oli vigade sisu, sai näha Jenkinsi veebirakendusest. (Muli ja Okoth, 2018)

Kogu projekti haldamine toimus Jiras. Jira on tarkvara, kus on võimalik arendusprojekt jagada väiksemateks tükkideks ning neid meeskonnasiseselt ära jagada. Jira on osa

tooteperekonnast, mis on loodud erinevat tüüpi meeskondade abistamiseks töö juhtimisel. Algselt oli Jira kavandatud vigade jälgimiseks, kuid täna on Jira arenenud võimsaks tööhaldusvahendiks igasuguste kasutusjuhtude jaoks, alates nõuetest ja testjuhtumite haldamisest kuni agiilse tarkvaraarenduseni. (Harned, 2018)

Antud projekti kirjelduse ja nõuete kohta koostati üks suurem Jira task, mille alla lisati juhendid äriarenduse poolt ning CSV-failid, mis sisaldasid endas erinevate *dropdown* menüüde valikute sisusid. Projekti peamise *task*'i alla lisati enda poolt koostatud väiksemad arendus-*task*'id, mis olid kõik suure projekti osad. Need jagati protsessi käigus omavahel laiali.

2.3 Tööprotsessi kirjeldus

Teine etapp algas etteantud projektiga seotud materjalide lugemise ning analüüsimisega, mille järel planeeriti tiimi edasine tegevus. Vaadati üle projekti alguses sätestatud nõuded ning vajadusel kohandati neid ja lisati esimese etapi jooksul tekkinud kasutajapoolsete uute ideede kirjeldused.

Projekti arendamisel kasutati Scrum metoodikat, mis on agiilse tarkvara arendamise raamistik. Meeskonna idee oli jaotada arendus sprintideks ehk iteratsioonideks. Projekti vältel kestis üks sprint kaks nädalat. Esimesel kümnel nädalal viis päeva nädalas ning viimasel kuuel nädalal kahel päeval nädalas. Sprintide planeerimisega tegeleti esimesel nädalal ning esialgu planeeriti kokku kaheksa sprinti. Igal hommikul kogunes tiim 15-minutiliseks *stand-up* koosolekuks, kus osalesid arendajad, tarkvaraarhitekt, projekti tellija ehk arendusjuht, lõppkasutaja ning tiimijuht. Iga arendaja pidi vastama kolmele küsimusele: „Mida tegin eile?“, „Mida plaanin täna teha?“, „Kas on takistusi?“. Kui kellelgi arendajatest oli tekkinud takistusi, siis sai ta pärast koosoleku lõppu abi küsida.

Scrum arendusmetoodikasse kuuluvad ka retrospektiivid, mida viidi läbi pärast igat sprinti. Retrospektiivi eesmärk oli välja tuua, mis eelmises sprindis hästi ja halvasti läks, et järgnevad sprindid sujuksid paremini. Kogu tiim pidi iga kord valima halbadest asjadest 2-3, mida järgneval sprindil parandada.

Tööd olid meeskonnas jaotatud selliselt, et projekti esimesel nädalal tegelesid kõik tiimi liikmed esimese etapi jooksul tekkinud vigade parandamisega. Seejärel juhinduti tööde jagamisel projekti plaanist ning igahommikuste *stand-up* koosolekute käigus otsustati,

kes millega tegeleb. Lisaks, kui kellelgi oli arendamise käigus abi vaja, siis prooviti esialgu omavahel üksteist aidata ning kui probleemile meeskonnasiseselt lahendust ei leitud, paluti abi vanemarendajatelt.

Projekti arenduse käigus testiti jooksvalt nii kasutajaliidese erinevaid funktsionaalsusi, kui ka koodi korrektset töötamist ABL Scratchpadis, kus jooksutati vastavat koodi lõiku ning kontrolliti, kas tagastatud väärtused on õiged. Lisaks kirjutati nii *front-end*'is kui ka *back-end*'is jooksvalt ühikteste.

Koodi refaktoreerimine tehti projekti lõpus enne, kui valmis rakendus testkeskkonda jõustati. Igaüks refaktoreeris eelkõige seda osa koodist, mida tema oli kirjutanud, sest koodiploki autoril on paremad teadmised koodi loogika ning võimalike erijuhtumite kohta. Refaktoreerimise käigus andsid meeskonnaliikmed ka üksteise koodiplokkidele tagasisidet ning võimalusel aidati sisse viia parendavaid lahendusi.

3 Töö tulemused

Projekti arendus põhines esmajoones RAB-i poolt meeskonnale väljastatud juhendil, kus on detailselt lahti kirjeldatud teatise edastamise üldsätted, kõik teatisel paiknevad infoplokid ning tehnilised nõuded teatise saatmise kohta (Lisa 1). Projekti eesmärgist lähtuvalt oli rakenduse arendamise juhtmotiiv see, et teatise plokkide täitmine oleks automatiseeritud. Nõuded ning juhendi automatiseerimise kohta koostasid meeskonnale AS SEB Panga äriarenduse divisjoni arendusjuht Enel Pitk ja *Compliance* osakonna esindaja Õie Õismaa, kes konsulteerisid ka vastavuskontrolli töötajatega. Dokumentatsiooni täiendati esimese etapi lõpus lähtuvalt uutest vajadustest ja kasutajatepoolsetest ideedest. IT-arenduse poolse kirjelduse koostasid IT Team Lead Mihkel Matson ning projekti arendajad Triin Tammaru, Marie Elise Soomre, Lilian Väli ja Carl-Robert Reidolf.

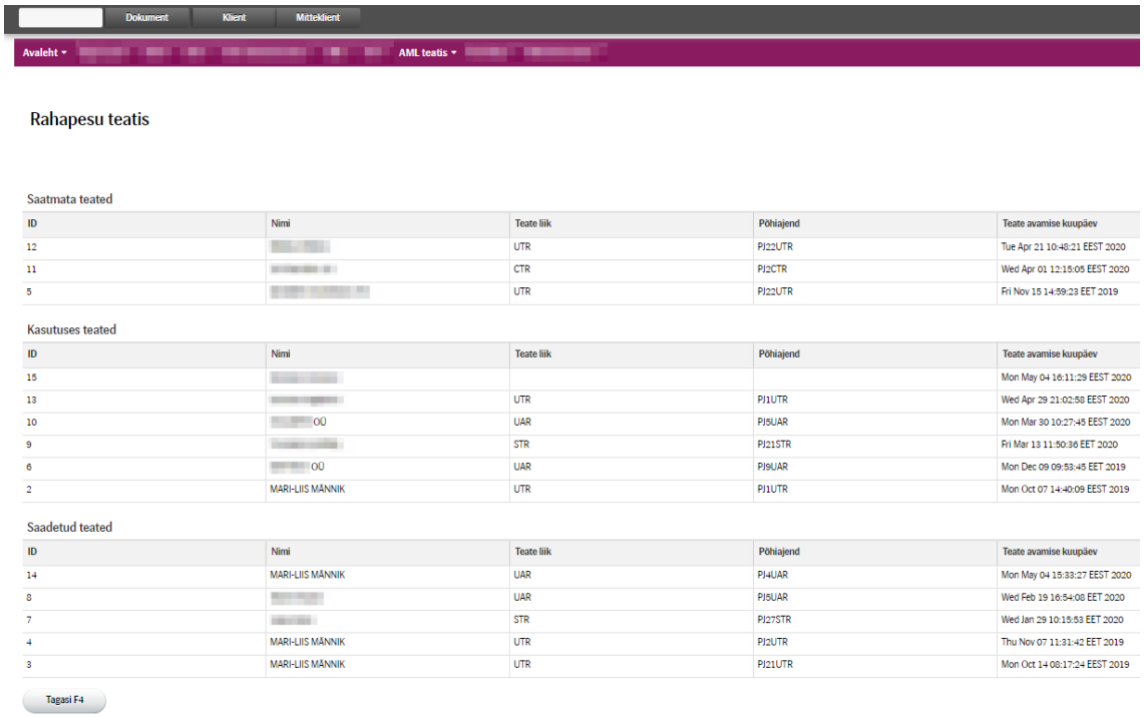
3.1 Rakendus

Rakendus arendati eraldiseisva plokina olemasoleva AS SEB Panga infosüsteemi, NBSFi, külge. NBSFi sisse logides avaneb kasutajale vaade vastavalt kasutajaõigustele, mis on talle isikuliselt määratud. Eelkõige puudutab see valikuid infosüsteemi menüüribal. Arendatud rakendusele saavad ligi ainult need vastavuskontrolli töötajad, kes on *Compliance* osakonna poolt volitatud kahtlaste toimingute kohta teatisi esitama ning nendele töötajatele on kasutajaõigusena määratud “AMLNOTIFIC” õigus. Õiguse saamiseks esitab *Compliance* osakond taotluse kasutajatoele, kes koostab tööülesande kasutajatoe spetsialistile, kes omakorda lisab kasutajaõiguse koodiga vastavale töötajal teatiste edastamise menüüpunkti õigused.

Menüüribal liikudes “AML Teatis” menüüpunkti valikule, avaneb rippmenüü, kus on kaks valikut - “Teatised” ja “Raportid”.

3.1.1 Teatiste avaleht

Valides “Teatised”, avaneb vaade, kus teatised on jagatud vastavalt staatustele kolme tabelisse - alustatud ehk salvestatud, kasutusel olevad ning saadetud teatised. Teatiste kohta on tabelites välja toodud järgnev info: teatise ID, esimesena lisatud osapoole nimi, teate liik, põhiajend, teate avamise kuupäev, selle algataja ning see, kas teatisega on kiire (Joonis 2).

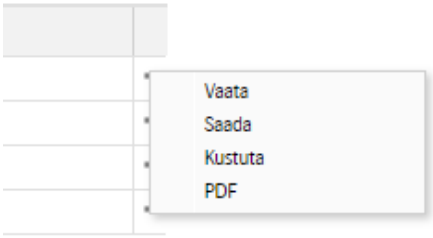


Joonis 2. Teatise osaline vaade.

Kui kasutaja on alustanud teatise täitmist ning soovib selle pooleli jätta, siis vajutades nupule “Salvesta” saab teatis endale staatuse “Alustatud”. Selle teatise uuesti avamisel käivitatakse teenus, mis edastab *back-end*’ile informatsiooni teatise avaja kohta ning teatis pannakse staatusesse “kasutusel”. Kui mõni muu kasutaja püüab seda samal ajal avada, siis tuleb talle teade, et teatis on juba kasutusel teise kasutaja poolt ning kuvatakse selle kasutaja tunnus.

Alustatud teatiste tabelis olevate kirjete viimases tulbas on nupud, millega avaneb rippmenüü (Joonis 3), kus on valikuteks teatise vaatamine (k.a muutmine), saatmine, kustutamine ning PDF-faili alla laadimine. Kui kasutaja on valinud “Saada”, siis avaneb uus lehekülg, kus kuvatakse, kas saatmine oli edukas või mitte. Kui teatise saatmisel

esines vigu, siis kõik ebakorrektsed väljad kuvatakse kasutajale listina, et neid saaks parandada (Joonis 4).



Joonis 3. Rippmenüü valikud teatiste tabelis.

Staatus ERRORID

Vigane väli	Veateade
notification:mainCause	mainCauseTerm value missing.
notification:notificationType	notificationtypeTerm value missing.
notification:transaction	TransactionDescription is mandatory

Joonis 4. Vigane teatis.

Saadetud teatiste tabelis olevate kirjete viimases tulbas on kaks nuppu - “PDF” ja “Kommentaar” Kommentaari lisamisel avaneb rakendusesisene hüpikaken suure tekstikastiga, kuhu on võimalik sisestada kuni 2000 tähemärki. Kui saadetud teatisele on juba varasemalt lisatud kommentaar, siis tehakse andmebaasile päring ja täidetakse kast selle kommentaariga ning kasutajal on võimalus seda muuta (Joonis 5).

Joonis 5. Kommentaari lisamine.

Vajutades “Teatised” lehel nupule “Uus teatis”, avaneb vorm “Uus rahapesu teatis”, mis on jaotatud plokkideks “Teate koostaja”, “Teataja”, “Teade”, “Lisaajendid”,

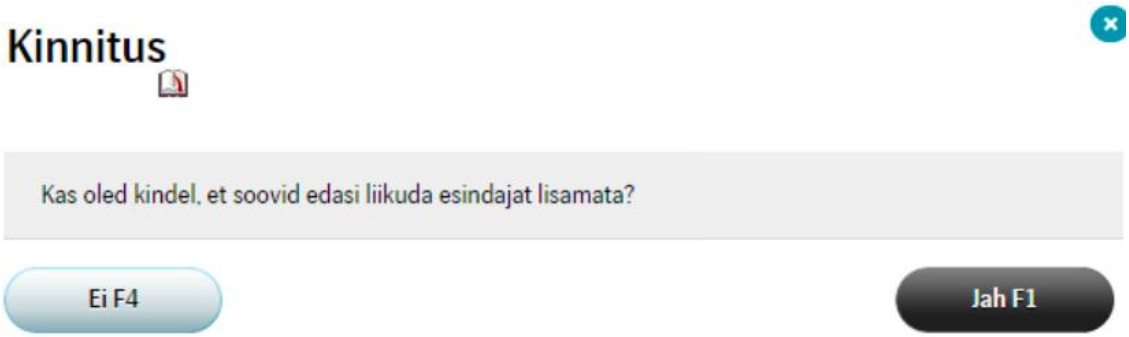
“Osapooled”, “Tehingud”, “Kogusummad” ja “Dokumendid”. Esimesed neli plokki said lõplikul kujul valmis projekti esimeses etapis.

3.1.2 Osapooled

Vajutades vormil nupule “Lisa osapool”, avaneb uus vorm hüpikaknas (Joonis 6). Vormi ülaosas on nupp “Otsi klienti”. Sellele vajutades avaneb aken, kuhu saab sisestada kliendi nime, isikukoodi või registrikoodi. Vajutades nupule “Otsi”, otsitakse andmebaasist, kas sellise nime või koodiga kliendile leitakse vaste. Kui vaste leitakse, siis kuvatakse kõik sobilikud valikud tabelis. Tabelikirjel vajutades saadetakse valitud kliendi kood *back-end*’i, kus teostatakse otsing andmebaasidest kliendi informatsiooni jaoks. Kasutaja vaade liigub tagasi algsele osapoole lisamise lehele ning kõik kliendi andmed täidetakse automaatselt.

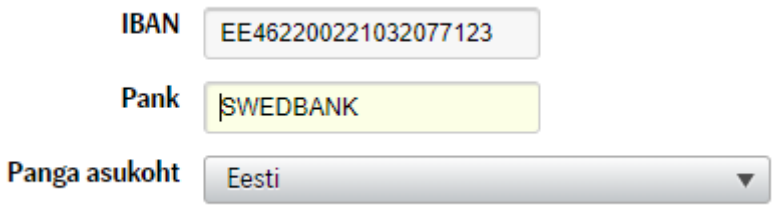
Joonis 6. Automaatselt täidetud osapoole andmed.

Osapoolele esindaja, volitaja ning seotud isiku lisamisel on kasutajal analoogne võimalus klienti andmebaasist otsida. Kui osapool on juriidiline isik ning kasutaja ei ole lisanud osapoolele esindajat, siis kuvatakse salvestamisel kinnitusvorm, mis küsib kasutajalt üle, kas ta on kindel, et ei soovi osapoolele esindajat lisada (Joonis 7). Vajutades nupule “Jah” lisatakse osapool tabelisse ning valides “Ei” suunatakse kasutaja tagasi eelnevale lehele, kus ta saab vajadusel esindaja lisada.



Joonis 7. Kinnitusvorm.

Kliendiinformatsiooni ploki all kuvatakse eraldi tabel kliendiga seotud kõikide pangakontode kohta. Automaatse väljade täitmisega ilmuvad tabelisse kõik kliendi pangakontod. Seejärel on võimalik kasutajal teostada valik, millised pangakontod on konkreetse kahtlase toiminguga seotud. Kui andmebaasist vajalikke pangakontosid ei leitud, siis on võimalik kasutajal need käsitsi osapoolele juurde lisada. Vajutades nupule "Lisa konto", avaneb uus vaade, kus tuleb sisestada konto IBAN, pank ning riik. Kui kasutaja sisestab IBAN-i lahtrisse kontonumbri, siis teostatakse andmebaasidest vaikimisi otsing, et võimalusel täita automaatselt väljad kontonumbri, panga ja riigi kohta (Joonis 8). Lahtrid täidetakse automaatselt juhul, kui kontonumber algab "EE"-ga ning on 20 tähemärki pikk.



Joonis 8. Automaatselt täidetud konto andmed.

Vajutades vormi allosas paremal asuvale nupule "Salvesta osapool", salvestatakse osapoolele järjekorranumber vastavalt sellele, mitmendana ta lisatud on. Numbri alusel kuvatakse osapooled põhivormil tabelis kasvavas järjekorras, kus neid saab kas muuta või kustutada vajutades tabelis vastavast kirjest paremal asuvale noolekesele.

Kui osapoolele ei ole identifikaatorit lisatud või AS SEB Pangal puudub selle kohta informatsioon, siis genereeritakse osapoolele automaatselt unikaalne identifikaator, mis koosneb tähtedest "KD" ning üheksast numbrist.

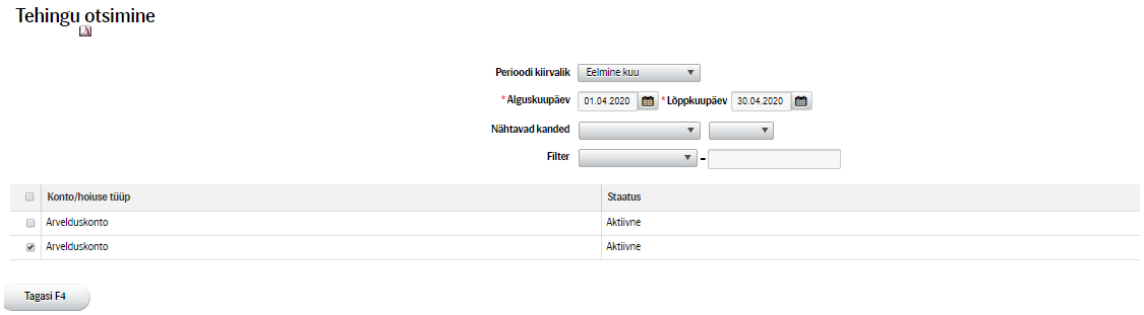
3.1.3 Tehingud

Vajutades vormil nupule “Lisa tehing” avaneb hüpikaknas uus vorm (Joonis 9), mille ülaosas on nupp “Otsi tehingut”. Nupule vajutades saab otsida AS SEB Panga klientide tehinguid. Kui kasutaja vajutab nupule, siis avaneb uus aken. Teenus otsib üles kõik osapooled, kes on teatisele lisatud ning kuvab neid rippmenüüs (Joonis 10). Sealt tuleb valida konkreetne klient, kelle tehingut kasutaja otsida soovib.

Joonis 9. Tehingu lisamine.

Joonis 10. Tehingu osapoole valimine.

Valides rippmenüüst kliendi, saab kasutaja vajutada nupule “Kuva kontod”, mis viib ta uude vaatesse (Joonis 11). Andmebaasile tehakse päring kliendi kontode kohta. Leitud kontod kuvatakse kasutajale tabelis. Kontodele, mis on seotud otsitava tehinguga, tuleb lisada linnuke. Lisaks on kasutajal selles vaates mitmeid valikuid otsingu kitsendamiseks. Esiteks, on võimalik valida perioodi. Näiteks saab valida, kas soovitakse näha eilseid, tänaseid, jooksva nädala või kuu tehinguid. Valik tehtud, muudetakse automaatselt algus- ning lõppkuupäeva väljad vastavalt valitud vahemikule. Lisaks on võimalik valida tehingute tüüp - intressi väljamakse, laenu tagasimaksed, maksekorraldus, sularaha või teenustasu. Samuti seda, kas tegu on kreedit või deebet tehinguga. Viimasena saab lisada spetsiifilise filtri, et otsida tehinguid näiteks kindla viitenumbri või konto alusel.

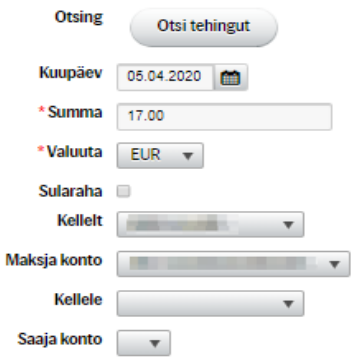


Joonis 11. Osaline tehingu otsimise vaade.

Pärast soovitud kontode linnutamist ning kitsenduste lisamist, saab kasutaja vajutada vormi allosas paremal asuvale nupule “Kuva väljavõte”, mis avab uue vaate. Käivitatakse protseduur, mis hakkab tootmissüsteemist tehinguid otsima. Uuel vaatel on toodud tabelina kõik tehingud, mis vastavad valitud kriteeriumitele. Seal on kuvatud tehingu kohta aeg, konto omanik, tema konto number, saaja konto omanik, saaja konto number, tehingu info, sissetulek, väljaminekud, valuuta ning tehingu liik (Joonis 12).

Joonis 12. Osaline tehingute valimise vaade.

Vajutades tabeli kirjele viiakse kasutaja tagasi algsele tehingu lisamise lehele ning andmed tehingu kohta täidetakse automaatselt (Joonis 13).



Joonis 13. Automaatselt täidetud tehingu lisamise plokk.

Tehingu lisamise lehe allosas paremal on nupp “Lisa SWIFT dokument”, mille abil saab tehingule lisada dokumenti, mis sisaldab konto väljavõtet. Nupuvajutusel avaneb uus vaade, kust saab vastava dokumendi kohta valida kuupäeva, lisada numbri, konto omaniku, kontonumbri ja sisu. Samuti on kasutajal võimalus lisada SWIFT dokumente, lohistades vastavasse kasti soovitud dokumendid. Lisatud fail või failid kuvatakse vaate allosas tabelis “SWIFT dokumendid”, kus saab dokumente vajadusel ka kustutada. Vajutades nupule “Salvesta dokument”, kuvatakse lisatud SWIFT dokument tehingu lisamise lehel tabelis “SWIFT kirjete koopiad”, kus seda saab muuta või kustutada.

Vajutades vormi allosas paremal asuvale nupule “Salvesta tehing”, salvestatakse tehing ning kuvatakse see “Tehing” ploki tabelis. Lisatud tehingut saab kas muuta või kustutada vajutades kirjest paremal asuvale noolekesele. “Tehing” ploki all on tabel “Kogusummad”, kuhu kuvatakse tehingute kogusummad valuutade järgi.

Tabeli “Kogusummad” järel on lahter tehingu kirjelduse jaoks ning kaks linnutatavat kasti küsimustega “Kas küsiti vara päritolu?” ja “Tuli tagastusnõue?”, mida valides ilmuvad vastavad plokid ka vara päritolu ja tagastusnõuete dokumentide lisamiseks. Teise etapi jooksul arendati nendele juurde võimalus laadida faile otse kasutaja arvutist.

3.1.4 Dokumendid

Vajutades dokumendi plokis nupule “Lisa dokument”, avaneb hüpikaknas uus vorm (Joonis 14). Vormi ülaosas on rippmenüü, kuhu valikud päritud eelgenereeritud setupist. Rippmenüüst saab valida dokumendi liigi, millest sõltuvalt tekivad vormile väljad, mida tuleb täita. Näiteks, valides dokumendi liigiks “Arve”, kuvatakse kasutajale kohustuslik arvenumbri lahter, valiku “Arvelduskonto väljavõte” korral aga konto omaniku ning pangakonto rippmenüüd. Erinevaid valikuid on kokku 13. Täidetavate väljade järel on kast, kuhu saab lohistada vajalikke faile.

Joonis 14. Dokumendi lisamise vaade.

Vormi allosas on kaks tabelit - üks dokumendiga seotud osapoolte kohta ning teine lisatud failide kohta. Tabelis “Lisatud dokumendid” (Joonis 15) kuvatakse kasutaja poolt üles laaditud failid, “Osapooled” tabelis aga näiteks kelle kohta on vastav dokument (nt kui liik oli “Helisalvestus” või “Foto”), kes on konto omanik juhul kui dokumendi liigiks on “Arvelduskonto väljavõte” või kes on kaardi omanik, kui liigiks on “Maksekaart”. Tabelitest on võimalik kirjeid kustutada.

Joonis 15. Lisatud failide tabel.

3.1.5 Teatise eelvaade

Valides “Teatised” vaates rea lõpust avanevast rippmenüüst “PDF”, kuvatakse kasutajale link nimega “PDF”, millele vajutades avaneb vaade “Printimise info”, kus on kuvatud faili suurus kilobaitides ning nupp “Prindi”. Nupuvajutusega on kasutajal võimalik rahapesukahtlase tegevuse kohta koostatud teatise formaalne ning kompaktne visuaalne eelvaade PDF-failina alla laadida. Allalaadimise järel kuvatakse kasutajale PDF-fail.

PDF dokumendi struktuur on jaotatud infoplokkideks sarnaselt teatise vormile. PDF-faili päises kuvatakse dokumendi nimi “Teatis Rahapesu Andmebüroole” ning paremal ääres on kuvatud AS SEB Panga logo.

Esiteks, on dokumendis kuvatud “Teataja” ja “Teate koostaja” plokid. “Teataja” all on kuvatud AS SEB panga nimi, registrikood, seaduse kohustatud subjekt, põhitegevusala ning teatega seotud tegevusala. “Teate koostaja” plokis teate koostanud töötaja ees- ja perekonnanimi, isikukood ning sidevahendid.

Osapoolte infoploki all on kuvatud osapoolte nimed, tüübid ning isiku- või registrikoodid. Pangakontode plokis samade osapooltega seotud kontonumbrid, pangad ning panga asukohad, peale mida on toodud välja tabel ka konkreetsete seostega osapoolte ja pangakontode vahel.

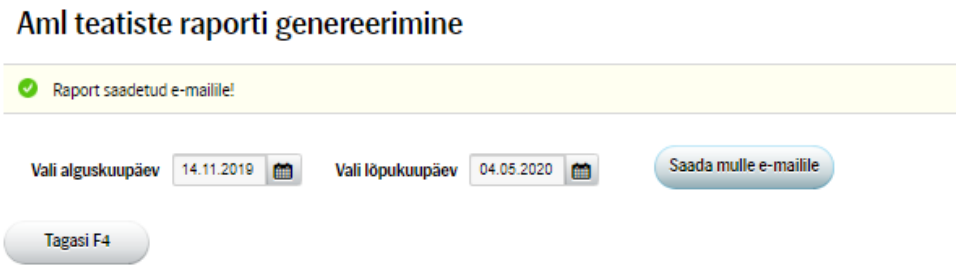
Tehingu osapoolte all on kuvatud kõik tehingutes osalenud osapooled koos esindajate ja volitajatega (kui need on antud osapoolele lisatud). Seotud isikute all kuvatakse osapooltele lisatud seotud isikud ning nende rollid.

Tehingute ploki all kuvatakse tehingute kirjeldus, tüüp, kuupäev, summa, valuuta, osapool koos kontoga, kes tehingu algatas, osapool koos kontoga, kellele tehing suunatud oli ning valuuta, milles teine pool summa kätte sai, tehingute kogusumma ning info vara päritolu küsimise ja tagastusnõude kohta.

PDF dokumendi lõpus on esitatud plokk lisatud dokumentide kohta, kus kuvatakse faili nimi ning faili tüüp. PDF dokumendi jaluses on välja toodud AS SEB Panga kontaktinformatsioon ning RAB-i kontaktinformatsioon.

3.1.6 Raport

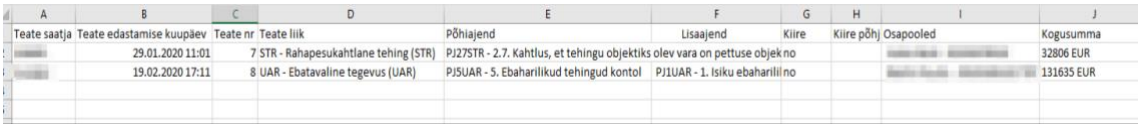
Valides “AML Teatis” menüüpunkti rippmenüüst valiku “Raportid”, avaneb vaade (Joonis 16), kus kasutajale kuvatakse kaks kuupäeva valiku lahtrit, kus mõlemas on vaikimisi väärtuseks käesolev kuupäev. Soovitud ajavahemik valitud, saab kasutaja vajutada kuupäevade kõrval olevale nupule “Saada mulle e-mailile”.



Joonis 16. Raporti genereerimine.

Nupuvajutusega saadetakse informatsioon valitud kuupäevade ning aktiivse kasutaja kohta *back-end*'i, kus otsitakse baasi salvestatud teatistest välja valitud ajavahemikku sobivad. Teatiseid otsitakse nende sulgemise kuupäeva alusel. Kuna baasi salvestatakse näiteks teate liigi kohta ainult sellele vastav setupi kood, kuid raportis on vaja kuvada ka selle kirjeldus, siis tuleb otsida *back-end*'is ka vastavad setupid ja salvestada ajutistesse tabelitesse nende kirjeldused.

Raportisse lisatakse teatise kohta erinevaid andmeid. Info lisatakse teate saatnud kasutaja, saatmise kuupäeva, teate järjekorranumbri ja liigi kohta. Lisatakse veel põhiajend ning lisaajendid koos nende setupi koodide ja kirjeldustega, kahtlaste tehingute kogusumma ja valuuta, tehingu kirjeldused ning see, kas tehing toimus sularahas. Samuti kõikide osapoolte nimed, nende isiku- või registrikoodid ja info selle kohta, kas küsiti vara päritolu, kas teatisega on kiire ning kui jah, siis mis on selle põhjus (Joonis 17).



A	B	C	D	E	F	G	H	I	J
Teate saatja	Teate edastamise kuupäev	Teate nr	Teate liik	Põhiajend	Lisaajend	Kiire	Kiire põhj	Osapooled	Kogusumma
	29.01.2020 11:01	7	STR - Rahapesukahtlane tehing (STR)	PJ27STR - 2.7. Kahtlus, et tehingu objektiks olev vara on pettuse objek		no			32806 EUR
	19.02.2020 17:11	8	UAR - Ebatavaline tegevus (UAR)	PJ5UAR - 5. Ebaharilikud tehingud kontol	PJ1UAR - 1. Isiku ebaharili	no			131635 EUR

Joonis 17. Osaline raporti sisu.

Voogedastuse abil kantakse andmed ühe teatise kaupa üle kasutaja kodukataloogi loodud CSV-faili. Kodukataloog on iga kasutaja personaalne serverikataloog. Kui kõikide sobivate teatiste kohta on informatsioon CSV-faili kantud, kontrollitakse, kas faili kantud teatiseid oli vähemalt üks ning kui jah, siis otsitakse *front-end*'ist saadud kasutaja kliendikoodi alusel üles vastav e-posti aadress ning saadetakse sellele koostatud CSV-fail. Kui sobivaid teatiseid ei leitud, siis kuvatakse kasutajale teavitus, et antud perioodil ühtegi teatist ei edastatud.

3.2 Arhitektuur

Teise etapi jooksul lisati panga ühte suurimasse andmebaasi juurde üheksa väljaga tabel üleslaaditud failide andmete hoiustamise jaoks. Lisaks sellele, lisati esimeses etapis loodud tabelitele kokku 20 uut välja. Suurem osa uutest väljadest on seotud uue funktsionaalsusega laadida üles faile. Need väljad lisati näiteks SWIFT, väljavõtte ja ID-dokumentide tabelitesse. Mitmeid välju tuli juurde ka teatisega seotud põhitabelisse ml-notification, kuhu lisati väljad selle kohta, kes hetkel teatist kasutab, mis on teatise staatus, kas teatis on RAB-i saadetud, mis on RAB-i vastus, millal ja kes viimati teatist muutis ning millal ja kelle poolt teatis suleti. Lisaks, väljad saadetud teatisele kommentaari lisamiseks ning viimase osapoole järjekorranumbri hoiustamiseks.

Tabelid on omavahel seotud identifikaatorite järgi. Eelnimetatud tabelite korral on ml-notification tabelis primaarvõtmeks ml-notification-id, sellega seotud tabelites on sama nimega väli aga võõrvõtmeks. Nii luuakse seos kahe tabeli vahel. Peamiselt on kasutatud üks-mitmele seoseid. Näiteks on tabelite ml-notification ja transaction-item vahel üks-mitmele seos, sest ühel teatisel võib olla mitu tehingut, kuid ühe tehingu kohta saab teha vaid ühe teatise. Seda seetõttu, et iga tehing, mis on konkreetse teatise küljes omab andmebaasis unikaalset tehingu identifikaatorit.

Andmebaasi tabelid ja nende kirjed on loodud .df failina ehk *data* failina ning tabelite loomiseks faili abil kasutasime Progress OpenEdge tarkvara. *Data* failis tuleb uue tabeli loomiseks kasutada käsklust ADD TABLE ning kindlasti tuleb defineerida ka *area* ehk piirkond, kuhu tabelid salvestatakse, et hoida kokku andmebaasi mälu ressurssi ning paigutada kõik võimalikult kompaktselt (Joonis 18). Olemasolevale tabelile uue välja lisamiseks tuleb kasutada käsklust ADD FIELD. Kindlasti tuleb välja tuua ka tabel,

millele välja lisatakse ning välja tüüp (Joonis 19). Antud projektiga on lisatud väljadele ka kindlad formaadid, algsed väärtused, positsioon tabelis ning maksimum pikkused.

```
ADD TABLE "additional-cause"
AREA "op128_data"
DUMP-NAME "additional-cause"
```

Joonis 18. Tabeli lisamine .df failis.

```
ADD FIELD "additional-cause-id" OF "additional-cause" AS integer
FORMAT "->,>>>,>>>,>>9"
INTIAL "0"
POSITION 2
MAX-WIDTH 4
ORDER 10
```

Joonis 19. Välja lisamine .df failis.

Back-end on struktureeritud kolmeks osaks - klassid, protseduurid ja *include*'id. Klassid on need, mida kutsuvad välja *front-end* teenused ning need peavad nii *front-end*'is kui ka *back-end*'is olema sama nimega. On olemas klassid kirjete andmebaasi salvestamiseks ning klassid nende andmebaasist pärimiseks või kustutamiseks. Protseduurid on need, mida klassid välja kutsuvad, et mingeid kirjeid töödelda. *Include* (.i laiendiga) klassid sisaldavad suuri andmekogusid ja ajutisi tabeleid, et ühilduda andmebaasi tabelitega. *Include* klassid aitavad vähendada koodi duplikeerimist, kuna nendes sisalduvaid tabeleid saab taaskasutada.

Back-end'is on loodud igale objektile ajutine tabel, mis salvestatakse peale töötlemist ümber andmebaasi. Tabeli kirjeteks on objekti muutujad ning tabelite seosed on kirjeldatud ProDataSetis, mis on Progress OpenEdge tarkvaras kogum ühest või mitmest ajutisest tabelist ning võib sisaldada nendevahelisi relatsioone. Antud rakenduse esimeses etapis loodud ProDataSetile lisati teise etapi jooksul 13 uut ajutist tabelit ning 25 uut seost. Kõik uued tabelid ning seosed on seotud dokumentidele failide lisamisega. Uued ajutised tabelid on näiteks ttAmlAttachedFile, ttAmlAttachedFileItems ja ttAmlAuthAttachedFileItems.

ProDataSetis saab tabelite vahel seoseid luua mitut moodi. Antud projektis kasutati selleks kahte võimalust. Esimene viis on luua seos kahe ajutise tabeli vahel kasutades DATA-RELATION funktsionaalsust ning seejärel defineerida mõlema tabeli jaoks väljad, mille järgi seos luuakse (Joonis 20). Teine viis on luua kahe ajutise tabeli vahel seos vanema ID järgi kasutades funktsionaalsust PARENT-ID-RELATION, kus seos vanema ja lapse puhvrite vahel on loodud vanema RECID välja järgi (Joonis 21). RECID on unikaalne neljabaidine sisemine identifikaator. Peale seose loomist defineeritaksegi vastav RECID.

```
DATA-RELATION rel10 FOR ttAmlDocument, ttAmlAttachedFileItems
RELATION-FIELDS(document-id, document-id) NESTED
```

Joonis 20. Koodinäide DATA-RELATION.

```
PARENT-ID-RELATION rel58 FOR ttAmlAuthAttachedFileItems,
ttAmlAttachedFile
PARENT-ID-FIELD ttAmlAuthAttachedFileItems_recid
```

Joonis 21. Koodinäide PARENT-ID-RELATION.

Andmete töötlemiseks kasutati ProDataSeti abil seotud ajutisi tabeleid, kuhu loeti andmed andmebaasist käsklusega BUFFER-COPY, mis teostab kõikide andmebaasi tabeli kirjete kopeerimise ajutise tabeli kirjetesse. Kopeerimine toimub vastavalt välja nimele. Samuti on BUFFER-COPY funktsiooniga võimalik määratleda väljad, mida soovitakse kopeerimisest välja jätta või väljad, mida soovitakse kaasata.

Front-end poolel kasutasime *Model-View-Controller* (MVC) arhitektuurimustrit. MVC *Model* komponent vastab andmetega seotud loogikale, millega kasutaja töötab. Antud rakenduse puhul esindavad seda erinevad *data* klassid, näiteks AmlNotifierData (Joonis 22), kus on defineeritud kõik teataja ploki alla käivad väljad ning vastavad *get* ja *set* meetodid. *View* komponenti kasutatakse kogu rakenduse kasutajaliidese loogika jaoks, mida antud rakenduses esindavad ZUL-failid. *Controller* toimib liidesena *Model* ja *View* komponentide vahel. See töötleb kogu äriloogikat ja sissetulevaid päringuid, manipuleerib andmetega kasutades *Model* komponenti ja suhtleb *View* komponendiga, et renderdada lõppväljundit. Antud rakenduse puhul esindab kontrollerit *UseCase* klass, mis suhtleb omakorda *ServiceDescriptor* klassidega, mis kutsuvad välja *back-end* teenuseid ning tagastavad siis väärtused uuesti *front-end*'i *UseCase* klassile.

```
public class AmlNotifierData extends AmlNotifierQuery {
	private String amlRegCode;

	public String getAmlRegCode() {
		return amlRegCode;
	}

	public void setAmlRegCode(String amlRegCode) {
		this.amlRegCode = amlRegCode;
	}
}
```

Joonis 22. AmlNotifierData osaline vaade.

3.3 Disain

Projekti dokumentatsioonis tellija algselt arendajatele disaini nõudeid ei esitanud. Dokumentatsioonis oli pikemalt lahti selgitatud, kuidas võiksid erinevad andmeväljad olla struktuuriliselt paigutatud ning milline võiks olla lahtrite järjestus ja suurus. Koos äriarenduse esindajatega, kes koostasid nõuete dokumentatsiooni, vaadati läbi RAB-i veebivormi struktuur ja loogika ning fikseeriti näited, kuidas pangasisese rakenduse puhul mõningaid kohti paremini lahendada. Osadel juhtudel jäeti teatud kohad ka sarnaseks. Mitmed nõuded selgusid arendustegevuse käigus kasutajapoolse tagasiside tulemusel.

Põhiliselt muudeti erinevate elementide ja lahtrite asukohti ning suurust. Kõik vormidel täidetavad andmeväljad on paigutatud vormi keskele ning plokkide pealkirjad asuvad vormidel vasakul suurema fondiga. Nupud asuvad rippmenüüde kõrval paremal või konkreetse ploki kohal. Lähtuvalt AS SEB Panga rakenduse arendamise kasutajakogemuse printsiipidest, paigutati “Tagasi” nupp igal vormil alla vasakusse nurka ning “Salvesta” nupp igal vormil alla paremasse nurka. Tabelid on kõikide vormide puhul horisontaalselt vormi ühest otsast teise, et mahuks kuvama võimalikult palju informatsiooni ning kasutajal oleks kogu tabel tervikuna visuaalselt nähtav.

Vormil on kastiplokkidele, kuhu kasutaja saab sisestada informatsiooni, arendatud juurde funktsionaalsus, millega kasutajal on võimalik kastiplokkide suurust käsitsi muuta. Lisaks on kohustuslike väljade juurde lisatud punased tärnikesed, mis juhivad kasutaja tähelepanu sellele, et see väli peab olema kindlasti täidetud. Mitmed väljad on

tingimuslikud ehk ilmuvad vormil füüsiliselt nähtavaks vaid siis, kui nende jaoks on teatud eeldused täidetud ning selleks, et liigsed väljad ei segaks kasutaja tööd.

3.4 Kood

Rakenduse valmimiseks kirjutatud kood jaotub kaheks - *back-end* ja *front-end*. *Front-end* on kirjutatud Java ning XHTML arenduskeeltes, *back-end* OpenEdge ABL ehk Progress arenduskeeles.

3.4.1 Front-end

Front-end'is on kõikide objektide jaoks olemas andmete ehk *data* klassid, milles on välja toodud objektide juurde kuuluvad muutujad. Klassides on lisaks muutujatele nende jaoks olemas *get* ja *set* meetodid, et objektide muutujaid välja kutsuda ja väärtustada. Muutujad on näiteks nimi, tüüp ja kirjeldus. Kõikidel objektidel on unikaalne identifikaator, mis on implementeeritud klassiga *Unique*. See lisab *data* klassile meetodi getUID, millele tuleb teha *override* ning meetod tagastab muutuja, mis on objektil unikaalne. Lisaks on osadele *data* klassidele implementeeritud klass *Selectable*, mis laseb kasutajal objekti valida neid näiteks tabelis linnutades.

Peamine klass, kus toimuvad kõik vajalikud protseduurid on AmlNotificationsUseCase, mis kutsutakse välja, kui kasutaja vajutab teatiste menüüpunkti rippmenüüst "Teatised". Esimesena kutsutakse välja init() meetod (Joonis 23), mis omakorda jooksutab kahte meetodit - loadAMLData ja showAMLData. Esimene kutsub välja *back-end* mooduli, mis tagastab *front-end*'ile kõik juba loodud teatised, et neid tabelis kuvada. Teine loob vaateakna, kus teatiseid kuvada ning lisab sellele lehele toimingud ehk nupud, mille vajutamisel teatud funktsionaalsus käivitatakse.

```
@UseCase
public class AmlNotificationsUseCase extends UseCaseBase {

	@Override
	public void init() throws TechnicalErrorBase, ValidationErrors {
		super.init();
		loadAMLData();
		showAMLData();
	}
}
```

Joonis 23. AmlNotificationsUseCase ja init meetod.

UseCase'is tuleb objekti töötlemiseks see vahemälusse luua. Seda tehakse meetodiga getOrCreateBeanFromCache, mis otsib hetkel jooksvast sessioonist objekti ning kui sellist ei leidu, siis loob selle. Seejärel saab kasutada objekti töötlemiseks selle meetodeid *get* ja *set*. Selleks, et täita objektide elemente automaatselt on neil vaja *query*'sid ehk päringuid, milles väärtustatakse objekti ID. Seejärel kutsutakse välja NBSX *back-end* moodul samanimelise klassiga *front-end'*is. *Enum*'ist "Back-end" valitakse *back-end* moodul, mida välja soovitakse kutsuda. Antud projektide puhul on selleks alati NBSX. Sõnumite saatmiseks kasutatakse Java Message Service API-t, mille käivitab juba eelnevalt olemas olnud meetod *SonicConnectorBase*.

Dokumentide lisamiseks on ZUL-failis defineeritud *dropUpload*, millele on lisatud objekt nimega uploadEventListener. Kui kasutaja valib enda arvutist failid, siis käivitatakse eelnimetatud *Listener* klass. *Listener* tüüpi klassis on meetod OnEvent(Event event), millele antakse sisse toimunud *event* ehk sündmus. Sündmuse objektilt saadakse kätte sellega kaasnev meedia, mis salvestatakse *Media* objekti. Selle objekti *ByteData* muutuja salvestatakse uuele *AttachedFile* objektile *body*'na ehk sisuna. Lisaks määratakse *AttachedFile* objektile nimi ja tüüp, mis saadakse ka eelnevalt salvestatud *Media* objektist. Viimasena lisatakse loodud *AttachedFile* objekt vahemälusse ning käivitatakse dokumentide *UseCase* - AmlDocumentsUseCase. See klass saadab faili uue loodud *connector*'iga repositooriumisse. *AttachedFile* objekt luuakse kõikide lohistatud failide jaoks.

Objektide kuvamiseks on tehtud ZUL klassid, mis on mõeldud vaadete jaoks. Neid kutsutakse *UseCase'*is välja *MVCPage*'ina. ZUL klassid asuvad <zk> *tag*'ide vahel. Iga ploki alguses on defineeritud *data* ehk millise objekti muutujaid kuvada tuleb. Peale seda

on välja toodud kõik muutujad sellises järjekorras, kuidas neid kuvada soovitakse. Nupud on defineeritud *oributton'*itena, mis on modifikatsioon ZK raamistiku *button*'ist. *Oributton*'id on defineeritud ka *UseCase* klassis *pageAction*'ina kindlatele vaadetele, kus on öeldud, millisesse funktsiooni see nupp kasutaja suunama peab.

3.4.2 Back-end

*Back-end'*i protseduur loeb XML-ina sisse objekti *query*. Samanimelises ajutises tabelis väärtustatakse ära väli ID. Seejärel hakatakse teostama meetodit processService, mis otsib andmebaasidest ID järgi vastavad andmed. Informatsiooni leidmisel luuakse uus ajutine tabel, mille väljad väärtustatakse vajalike andmetega. WRITE-XML kirjutab selle tabeli XML formaati ning seejärel tagastatakse see fail uuesti *front-end'*ile üle Sonicu adapteri. Ajutistes tabelites on lisatud väljadele XML-NODE-NAME, mis ütleb ära, millise nimega väljad *front-end'*i poole peal loodud on. *Front-end* saab aru, et on sisse loetud sama nimega tabel ning loob selle objekti *bean'*i vahemällu.

Salvestades *back-end'*is objekti andmebaasi loetakse kõigepealt see sisse XML failina. Seejärel kontrollitakse, kas selline objekt on juba olemas andmebaasis. Kui ei ole, siis luuakse andmebaasi uus kirje ning väärtustatakse vajalikud väljad. Objekti otsitakse andmebaasist indeksi järgi, enamasti on selleks ID või mõni unikaalne muutuja nagu näiteks kontonumber või registrikood.

3.4.3 Teatise eelvaade PDF-failis

Teatise eelvaade kuvatakse kasutajale PDF-failis ning selle loomiseks saadab *front-end back-end'*ile kõigepealt päringu koos vastava teatise ID-ga (Lisa 2). Progress OpenEdge'is arendati *back-end*'i protseduur, mis saab teenuse klassilt sisendparameetriks teatise ID ja selle põhjal kogutakse bank ja logbank andmebaasidest konkreetse teatise andmed. Kogutud andmed salvestatakse ajutistesse tabelitesse.

Tabelite ja loodud ühenduste põhjal koostatakse Progress OpenEdge'i integreeritud hSaxWriter funktsiooniga nõuetele vastav XML-fail, mille korrektsust kontrollitakse RAB-i poolt väljastatud XSD-faili vastu, kus on defineeritud kogu edastatavate teatiste struktuur. XML-failile on loodud omakorda vastav XSL-fail (Joonis 24) nimega fo-AMLNOTIFICATION, kus on kirjeldatud PDF-failil kuvatava teatise eelvaate vorming, disain ja paigutus. Fo-AMLNOTIFICATION faili nimi on salvestatud objektina bank

andmebaasi setup andmebaasitabelisse koodiga ning *back-end* protseduuris leitakse antud koodi põhjal koostatud XML-failile vastav XSL. Seejärel käivitatakse teenus, mis edastab XSL-faili ja XML-faili FOP-serverile, mis koostab saadetud failide põhjal PDF-faili (Lisa 3) ning kuvab selle kasutajale, kes PDF-faili genereerimise teenuse käivitas.

```
<xsl:for-each select="soapenv:Body/prod:sendNotification/
prod:notification/bankAccountRoles/bankAccountRole">

      <fo:table-row>

            <fo:table-cell xsl:use-attribute-sets="table-border"
            number-columns-spanned="1">
                  <fo:block xsl:use-attribute-sets="text"
                  font-size="7pt">

                        <xsl:value-of select="personRefId/text()"/>

                  </fo:block>

           </fo:table-cell>

      </fo:table-row>

</xsl:for-each>
```

Joonis 24. XSL struktuur.

3.4.4 Postman

AS SEB Panga poliitika äriinformatsiooni käitlemine näeb ette, et erinevad väärtused, mida kasutatakse väljade eeltäitmisteks, rippmenüüde valikute sätestamiseks või andmed, mis on ärikriitilise tähtsusega ning võivad lühikese aja jooksul muutuda, salvestatakse andmebaasi tabelisse nimega setup. Poliitika eesmärk on hoida äriinformatsioon interaktiivsena ning vähendada IT-töötajate ajaressursi kulumist selle uuendamise peale. Setup tabelisse salvestatakse informatsioon arendaja poolt Progress OpenEdge'is kirjutatud skriptide abil, mis loevad väärtused sisse CSV-failidest. Setupidele tuleb lisada nende *parent*-setup, nimi, tüüp ja kood. Tihti on sarnastel setup kirjetel ühine objektikood ning nad kõik kuuluvad ühte listi.

Teatise kõikide rippmenüüde valikud seadis projekti puhul ette RAB. Teatisel saab kasutaja rippmenüüde valikutest valida kokku 1169 erineva väärtuse vahel, mis on kõik salvestatud setup kirjetena. Rippmenüü valikute väärtuste kätte saamiseks kasutati Postman rakendust. Postman päringu sooritamiseks tuli kõigepealt koostada formaalne taotlus RAB-i, et RAB võimaldaks AS SEB Pangale üle nende poolt pakutavate teenuste päringuid saata. Taotluse heaks kiitmisel avati pöördujale ee/COM/10004252/sebpank juurdepääs RABi poolt pakutavatele teenustele, mis käsitlevad teatiste edastamist. Seejärel sooritati Postmani abil *Get* päring (Joonis 25) RAB-i poolse REST API vastu, mida infrastruktuuriliselt haldab Siseministeeriumi infotehnoloogia ja arenduskeskus. Postmani päringule saadi vastu .xml laiendiga fail, kus olid reastatud kõik erinevad väärtused, mida teatise rippmenüüdes kasutada (Joonis 26). XML-failist kopeeriti saadud väärtused ümber XLSX-faili ning need jagati omakorda vastavalt teatise plokkidele eraldi XLSX-failidesse, mis salvestati ümber CSV-failidesse, mida on kokku 20.

```
ee/GOV/70008747/rabis/getClassifier/v1
```

Joonis 25. Postman päring.

```
<prod:classifierInstance>
      <prod:vocabularyCode>RABIS2_Ajend</prod:vocabularyCode>
      <prod:vocabularyName>Ajend</prod:vocabularyName>
      <prod:termCode>STR_2</prod:termCode>
      <prod:termName>2. Tehingute teostamisel.</prod:termName>
</prod:classifierInstance>
```

Joonis 26. Postman päringu vastus.

Progressis koostasid arendajad setup skripti, mis kogus kõikidest CSV-failidest, kuhu salvestati teatise rippmenüüde valikud, väärtused kokku ning salvestas need loodud kuuetäheliste koodide alusel setup tabelisse. Setup tabeli kõikidele väärtustele saavad ligi äriarenduse divisjoni ning ka IT-divisjoni volitatud töötajad PuTTY tarkvara abil. PuTTY tarkvaras on loodud kasutajaliides, kus äriarenduse divisjoni töötajatel on võimalik kiiresti erinevaid väärtuseid muuta, kustutada ning ka lisada.

3.4.5 Teatise saatmine ja RAB-i vastus

Meeskonnaprojektiga valminud lahenduse puhul oli kasutajal võimalik valida teatiste vaates tabeli kirjete lõpus olevast nupust avanevast rippmenüüst “Saada”, seejärel kuvas infosüsteem kasutajale hüpikakna teatise saatmise kinnitamiseks. Kui kasutaja nõustus,

et teatise informatsioon on korrektne ning teatis on valmis edastamiseks RAB-ile, vajutas ta nuppu “Kinnita”, mis tähendas, et arvatavasti teatis saadeti välja ning eeldatavalt jõudis teatis ka RAB-i. Kontrollimiseks oli võimalik saata e-mail päringuga, kas teatis jõudis nendeni.

Lõputöö üks eesmärkidest oli parendada üle X-tee teatiste saatmise protsessi. Lisaks oli eesmärk arendada lahendus, mis annaks informatsiooni teatise väljastamise kohta ning kuvaks kasutajale koheselt teavet selle õnnestumise ja kohale jõudmise kohta. See tähendas, et tuli arendada NBSF-i ja NBSX-i teenuste ning protseduuride koostööl toimiv lahendus (Lisa 4).

Kui kasutaja hüpikaknas kinnitab, et teatis on valmis edastamiseks, siis kuvatakse talle uus vaade saatmise õnnestumise kohta. Selle jaoks kirjutati ringi *back-end* protseduur, et see salvestaks andmeid korrigeeritud *include* faili uutesse ajutistesse tabelitesse. RAB-i poolse vastuse töötlemiseks arendati juurde uus *back-end* moodul, kus on kaks ajutist tabelit, millest ühes hoitakse positiivse stsenaariumi XML elementide väärtuste muutujaid ning teises negatiivse stsenaariumi XML elementide väärtuste muutujaid. Viimase stsenaariumi korral kuvatakse kasutajale vigaseid andmeväljasid või tehnilise vea põhjused. Kui teatise väljadele sisestatud andmetüübid on korrektsed, kõik lisatud osapooled on seotud kahtlase toimingu pangakontode või mõne muu aspekti kaudu ning teatis jõuab füüsiliselt X-tee vahendusel RAB-i institutsioonile, kuvatakse kasutajale “SUCCESS” teade. Seejärel teade krüpteeritakse ning saadetakse RAB-ile ja sisestatakse RAB-i infosüsteemi seal edasiseks töötlemiseks. Kui teatise informatsioon on sisestatud ebakorrektselt, teatise osapoolteks on lisatud teatisega mitteseotud isikuid või X-tee ühenduses esineb tehniline tõrge, siis kuvatakse konkreetse põhjuse kohta kasutajale vastavasisuline teade.

RAB-i puhul oli antud projektis AS SEB Panga jaoks tegemist pangavälise institutsiooniga ning seetõttu oli füüsilise suhtluse tehniliseks teostamiseks vaja kasutada AS SEB Panga turvaserverit, mis saatis AS SEB Panga poolt välja saadetava teatise XML-i formaadis RAB-i ning ka vastupidi. Turvaserveri formaalseks kasutamiseks kirjeldati vastav turvapoliitika. Lepiti kokku, millise sisuga ning milliste laienditega faile tohib pank välja saata ning vastu võtta.

RAB-i vastus jõuab panga süsteemidesse X-tee vahendusel XML-faili struktuuri kujul. XML-faili võtab vastu turvaserver, kust vastav *back-end* teenus loeb selle vastust töötlevasse moodulisse sisse, salvestab informatsiooni *temp-table*'itesse ning edastab teise teenuse abil *front-end'*ile vajalikud andmed vastusest. *Front-end* poolel on kirjeldatud teenus, mis paigutab disainitud vaatesse *back-end'*i poolt edastatud andmed.

X-tee vahendusel failide saatmiseks otsib skripti kujul protsess andmebaasist faili, mis tuleb saata pangast välja ning seejärel laeb faili üles AS SEB Panga turvaserverile. Turvaserveri käsurealt käivitatakse käsk (Joonis 27) päringu edastamiseks ning sooritatakse *Post* päring RAB-i teenusele (Joonis 28). Samamoodi toimib ka failide vastuvõtmine. Fail jõuab esmalt turvaserverile ning sealt üle Progressi moodulteenuse loetakse fail *back-end* protseduuride abil sisse.

```
curl -v http://lslee-xroad.dmz.baltic.seb.net -X POST -H
'Content-Type: text/xml; charset=UTF-8' --data-binary
@postFileName -o outFileName
```

Joonis 27. Curl käsk.

```
ee/GOV/70008747/rabis/sendNotification/v1
```

Joonis 28. Saatmise teenus.

RAB-i vastust töötlev moodul on üle teenuse ühendatud *front-end* klassiga ning pärast andmete ajutistesse tabelitesse paigutamist, edastab teenus andmed *front-end'*i, kus vastav informatsioon kasutajale kuvatakse. *Front-end'*is arendati juurde andmeklassid, kus hoitakse *back-end'*ist saadud andmeid RAB-i vastuse kohta ja uusi vaateid ning *UseCase*'i loogika muudatus, mis võtab andmeklassidest andmed ning edastab need vaadetesse.

3.4.6 Logimine

Back-end protseduuride ja klasside jaoks arendati loogikale juurde AS SEB Panga süsteemi logidesse kirjete tekitamise funktsioonid. AS SEB Panga infosüsteemide ja rakenduste tegevus logitakse erinevatesse logifailidesse lähtuvalt konkreetse mooduli sisust. RAB-i edastatavate teatiste rakendusele loodi eraldi uus logifail nimega curllog.log.

Logifaili kirjete tekitamiseks arendati RAB-i teatise saatmise *back-end* moodulile vastav loogika. Moodulile lisati globaalne *include'*i fail, mille sees kutsutakse välja teenus, mis saab sisendid antud moodulist ning võimaliku vea korral teeb kindlaks, millist tüüpi veaga on tegu. Seejärel käivitati enne mooduli kompileerimist RUN SYSLOG-OPEN käsuga logimise protseduur, mis jälgib mooduli tegevust. RUN SYSLOG-LOG käsuga on võimalik fikseerida kohad moodulis, mida soovitakse curlog faili logida. Curllog faili logikirje koostatakse hetkel, mil moodul turvaserverile faili edastab ja faili pangast väljastab. Logikirjes märgitakse mooduli nimi ning vastav kood, mis tuleneb *include*'i failist, kus määratakse staatus - 200 ehk kõik korras, 301 ehk päring ei ole aktsepteeritav, 500 ehk tehniline tõrge või esineb tundmatu viga. RUN SYSLOG-CLOSE käsuga lõpetati mooduli logimise protseduur pärast mooduli kompileerimise lõpetamist. Seejärel kirjutati logikirje curl.log faili, mis asub *live* keskkonna serveril.

3.5 Testid

*Back-end'*is valmisid ühiktestid kasutades ABL ühiktestimise raamistikku, mis on väga sarnane JUnit ja XUnit põhinevatele testimise raamistikele. Testide jaoks loodi uus projekt, mis pärib kõiki AML teatise projekti protseduure. Projekti alla on lisatud testprotseduurid, mille sees defineeritakse testid @Test annotatsiooniga. Ühiktestid on loodud ainult protseduuride testimiseks, et näha, kas andmebaasi tabelitega toimuvad kõik protsessid korrektselt. Mõnede testprotseduuride lõpus kasutatakse annotatsiooni @After, mis teeb kasutatud ajutised tabelid tühjaks.

Näiteks GetAmlBankData.p testprotseduuri klass testib seda, kas IBANi järgi otsitakse üles õige pank. Näiteks getBankDataSEB testiga antakse ühe SEB panga kliendi IBANi järgi protseduurile GetAmlBankData.p sisendiks vastav IBAN ning väljundina defineeritakse konto andmete jaoks ajutine tabel. Järgnevalt kontrollitakse *Assert* meetodite abil, et kontonumbri, panga nime ning asukoha väljad ei oleks tühjad ja et panga nimi oleks "AS SEB PANK" ning asukoht "EST" ehk Eesti (Joonis 29). Teine test teeb sama, mis esimene, kuid olemasoleva panga IBANi asemel annab sisendiks suvalise numbrite jada ning kontrollib *Assert* meetodite abil, et kontonumbri väljal oleks endiselt etteantud väärtus, kuid seekord peaksid panga nime ja asukoha väljad olema tühjad (Joonis 30).

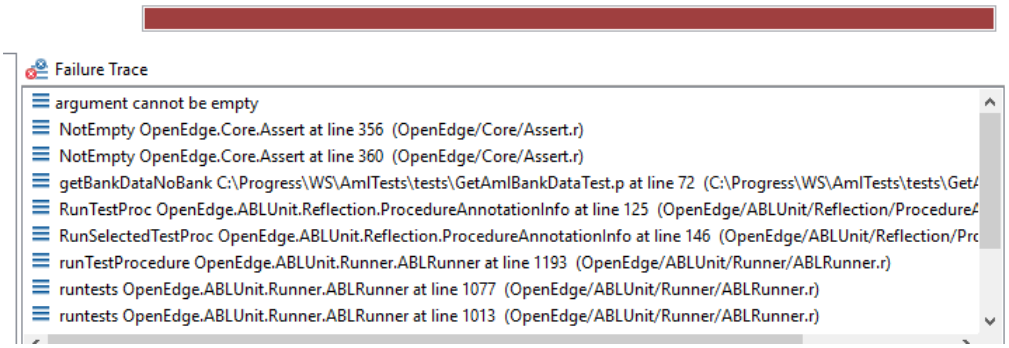
```
@Test.
PROCEDURE getBankDataSEB:
    DEFINE VARIABLE testIBAN_SEB    AS CHARACTER.
    testIBAN_SEB = "EE461010220074514017".

RUN services/aml/GetAmlBankData.p (INPUT testIBAN_SEB, OUTPUT
TABLE ttAmlAccountData).

    FIND FIRST ttAmlAccountData NO-LOCK NO-ERROR.
    Assert:NotEmpty(ttAmlAccountData.amlNewPartyAccount).
    Assert:NotEmpty(ttAmlAccountData.amlPartyBank).
    Assert:NotEmpty(ttAmlAccountData.amlPartyBankLocation).
    Assert:Equals(ttAmlAccountData.amlNewPartyAccount,
testIBAN_SEB).
    Assert:Equals(ttAmlAccountData.amlPartyBank, "AS SEB PANK").
    Assert:Equals(ttAmlAccountData.amlPartyBankLocation, "EST").
END PROCEDURE.
```

Joonis 29. getBankDataSEB meetod.

```
@Test.
PROCEDURE getBankDataNoBank:
DEFINE VARIABLE testIBAN_NoBank   AS CHARACTER.
      testIBAN_NoBank = "1122330".

    RUN services/aml/GetAmlBankData.p (INPUT testIBAN_NoBank,
OUTPUT TABLE   ttAmlAccountData).

    FIND FIRST ttAmlAccountData NO-LOCK NO-ERROR.
    Assert:NotEmpty(ttAmlAccountData.amlNewPartyAccount).
    Assert:IsEmpty(ttAmlAccountData.amlPartyBank).
    Assert:IsEmpty(ttAmlAccountData.amlPartyBankLocation).
    Assert:Equals(ttAmlAccountData.amlNewPartyAccount,
testIBAN_NoBank).

END PROCEDURE.
```

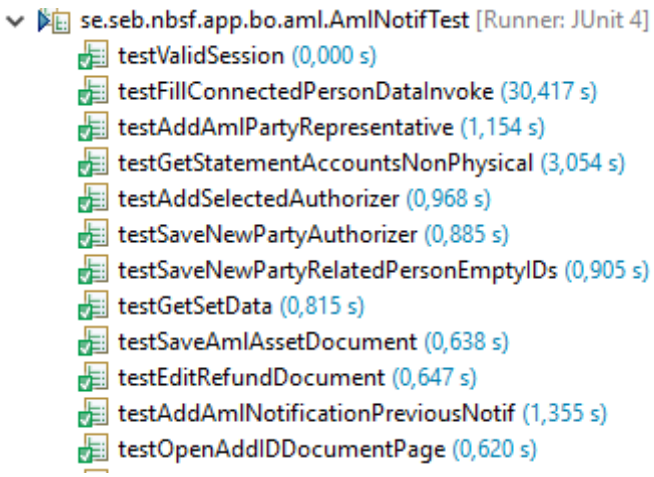
Joonis 30. getBankDataNoBank meetod.

Testprotseduuri käivitades jooksutatakse ükshaaval läbi kõik selles sisalduvad testid. Kui mõne testiga *Assert* meetod ebaõnnestub, siis kuvatakse kasutajale *Failure Trace* (Joonis 31) aken, milles on näha, kus ja miks viga tekkis. *Failure Trace* akna kõrval kuvatakse ka kõik ülejäänud testid ning nende staatus õnnestumise kohta.



Joonis 31. *Back-end*'i Failure Trace.

*Front-end'*i poole peal on koostatud testid *AmlNotificationsUseCase*'i meetodite jaoks (Joonis 32). Testid on loodud JUnit 4 raamistikku kasutades. Testide kirjutamiseks tuli seadistada init() meetod (Joonis 33), mis omab annotatsiooni *@BeforeClass*, mis tähendab, et seda käivitatakse ühe korra enne kõiki testmeetodeid. Seal luuakse uus NBSF sessioon ning käivitatakse Spring raamistik. Lisaks tuli JUnit klassile lisada *Run* ja *Debug Configuration* alla mitmeid .jar faile, millest testidel sõltuvus on. Need failid on oluline komponent Spring raamistiku konfigureerimisel, et käivitada moodulid õiges järjekorras. Testidega on kontrollitud, kas meetodite jooksutamisel tekib õige objekt vahemälusse, kas *back-end*'ist saadud XML on korrektses formaadis ning kas objektidega toimuvad vajalikud protseduurid. Näiteks testValidSession kontrollib, kas jooksutatav sessioon on õigesti konfigureeritud (Joonis 34). Testide ülesehituses on kasutatud *Try...catch* plokke, et töödelda tekkivaid erindeid (Joonis 35).



Joonis 32. Osaline testide vaade.

```
static NBSFSession sess;
static AppClient appClient;

@BeforeClass
public static void init() {
	sess = new NBSFSession();
	appClient = (AppClient)
SpringHelper.getBean(AppClient.class);
	sess.login(new ORIUserSessionTestInitializer("client"));
	sess.getOriSession().getUserSession().getOfficerSession()
.setSessionUid("SXXXXA");
	appClient.setSession(sess.getOriSession());
}
```

Joonis 33. Initsialiseerimise meetod.

```
@Test
public void testValidSession() {
	try {
		Assert.assertTrue(sess.getUserSession().isValid());
	} catch (TechnicalErrorBase | ValidationErrors e) {
		e.printStackTrace();
	}
}
```

Joonis 34. Sessiooni korrektsuse test.

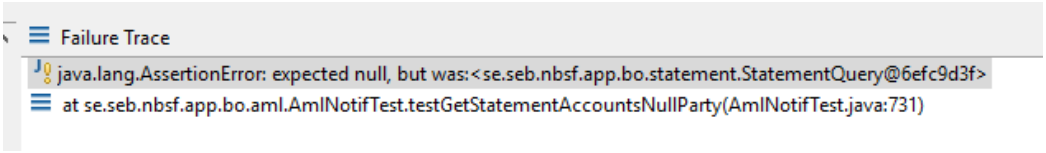
```
	@Test
	public void testloadAmlData() {
		try {

		} catch (TechnicalErrorBase | ValidationErrors e) {

		}
	}
```

Joonis 35. Testide ülesehituse näidis.

Kui test (Joonis 36) ebaõnnestus, kuvati kasutajale *Failure Trace* (Joonis 37) aken, milles viidati võimalikule ebakorrektsusele koodis. Selle põhjal oli arendajal võimalik sisse viia parandus, et kood töötaks korrektselt.

```
StatementQuery statementQuery = (StatementQuery)
appClient.getBeanFromCache(StatementQuery.class);
Assert.assertNull(statementQuery);
```

Joonis 36. Objekti testinäide.



Joonis 37. *Front-end*'i Failure Trace.

4 Analüüs ja järeldused

4.1 Tehnilise teostuse analüüs

Projekti sisuline loogika lähtus eelkõige RAB-i poolt meeskonnale antud juhendist ning äriarendusjuhtide koostatud dokumentatsioonist. Rakenduse tehnilise teostamise plaan ning rakenduse arhitektuur kooskõlastati IT-divisjoni arendajate ning arhitektidega. Ühiste koosolekute tulemusel leiti õiged raamistikud rakenduse arendamiseks ja testimiseks.

4.1.1 Nõuded

Põhiidee nõuete kirjeldamisel oli see, et vastavuskontrolli töötaja saaks oma tööd teha ühes keskses pangasüsteemis ning *Compliance* osakonna üldine tööpraktika teatiste koostamisel ja edastamisel ei muutuks. Samuti oli oluline aspekt andmete käsitlemise turvalisus, sest kõik see, mida üks töötaja teeb pangasiseses süsteemis logitakse ning panga seisukohalt on hea, kui niivõrd kriitilise tähtsusega tööülesannete täitmist on võimalik jälgida.

Edastatavate teatiste sisu puudutab klientide konfidentsiaalset informatsiooni, mis tähendab, et nende edastamine on reguleeritud mitmete seaduste ja määrustega. Eesti Vabariigis on Vabariigi Valitsuse poolt välja antud määrus “Infosüsteemide andmevahetuskiht”, mis kehtestati avaliku teabe seaduse (2020) § 439 lõike 1 punkti 5 alusel. Kõik, mida välja saadetakse üle X-tee, millisel kujul on informatsioon ning millises formaadis, määrab AS SEB Panga jaoks antud määrus ja RAB. See lihtsustas arendustegevuse planeerimist, sest suur hulk tehnilisi ning juriidilisi küsimusi olid leidnud lahenduse enne projekti algust.

4.1.2 Arhitektuur

Arhitektuuri loomisel lähtuti relatsioonilisest andmebaasi mudelist. Relatsioonimudelis on andmed loogiliselt struktureeritud relatsioonides ehk tabelites, mis on omavahel seotud. Tabelite omavaheline sidumine käib võtmete ehk identifikaatorite järgi. Kuna

tabelid on üksteisest sõltumatud, siis on andmed baasis muutuste suhtes paindlikud. (Harrington, 2016)

Andmebaasi mudeli disainimiseks kasutati Microsoft SQL Server Management Studiot. Antud tarkvara abil kirjeldati kogu mudel kõikide väljade, tabelite ning seostega. Andmebaasi mudeli skeem kooskõlastati AS SEB Panga NBSX süsteemiarhitekti ja IT *Team Lead*'iga, kes andsid tagasisidet, mille põhjal antud mudelit täiendati.

AS SEB Pangas kasutatakse OpenEdge ABL integreeritud andmebaasi, mis põhineb relatsioonilisel andmemudelil. Seetõttu lähtuti andmebaasi arhitektuuri ning tabelite loomisel just relatsioonilisest andmebaasi mudelist. Kuna objektidega seotud tabeleid oli vaja omavahel siduda, siis tuli kasutada tabelite jaoks võtmeid ehk identifikaatoreid.

Progress ABL-is loodi ajutised tabelid eelkõige selleks, et ennetada võimalikke tehnilisi tõrkeid andmebaasi töös otse andmeid protseduuridega töödeldes ja muutes. Samuti muudavad ajutised tabelid andmebaasi kasutamise sünkroonseks, kuna Progress ABL-is otse andmebaasi kirjet muutes võetakse konkreetne kirje lukku ehk muudetakse teistele kasutajatele kättesaamatuks. Ajutiste tabelite andmed on peale töötlemist võimalik hõlpsalt andmebaasi tabelisse ümber kirjutada. Lisaks oli vaja luua ProDataSet, et kirjeldada ajutiste tabelite omavahelised relatsioonid.

*Front-end'*is tuli järgida AS SEB Panga üldist struktuuri. Objektidele on loodud mudeli klassid, vaated on loodud *MVCPage*'ina ZUL-failis ning kontrolleriks on *UseCase* klass. *UseCase*'is antakse väärtuseid mudelitele ning kutsutakse välja vaateid.

Alternatiivid projekti arhitektuuri osas puudusid, sest AS SEB Panga kesksüsteemi rakenduste jaoks on määratud konkreetsed tehnoloogiad. Projekti rakendus integreeriti NBSF rakenduse külge ning *back-end* moodulid kirjutati NBSX rakendusse ja seetõttu ei olnud valikut võtta kasutusele teistsugust tehnoloogiat ei *front-end'*is ega *back-end'*is. Uute tehnoloogiate kasutusele võtmine oleks tähendanud olemasolevate arenduste ümber kirjutamist või täiendamist.

4.1.3 Disain

Rakenduse disain tuli arendada vastavalt NBSF süsteemi võimalustele. Vastavuskontrolli töötajate peamised soovid olid, et vorm oleks ühel lehel ning võimalikult lihtne ja et kõik nupud oleksid vormil sarnaselt paigutatud. Rakendus arendati eraldiseisva plokina juurde

juba eelnevalt valmis olnud tellerite töökeskkonnale NBSF. Tulenevalt NBSF süsteemi piirangutest, ei saadud teha vormile ega tabelitele ise disaini, vaid kõik disainielemendid ja mustrid tulid automaatselt vaadete aknas objektide loomisega. Muuta sai vaid objektide paigutust ning suurust. Kõik vaadete failid kasutavad AS SEB Pangas ühte CSS-faili, et terve süsteemi välimus oleks ühtne.

Kõik vastavuskontrolli töötajate soovid said projektiga täidetud. Nuppude ning väljade paigutusi arutati koos äripoolega ning lõppkasutajatega koosolekul, kus arendajad esitlesid ekraani või esitleti lõppkasutajatele erinevates staadiumites erinevaid disaini- ja paigutusvõimalusi.

4.1.4 Kood

Meeskond järgis *front-end'*i koodi kirjutamisel raamatu Clean Code (Martin, 2008) põhimõtteid. Kõik muutujate nimetused kirjeldavad enda sisu ning ei ole kasutatud ühe- või kahetähelisi nimetusi. Kõik klasside nimetused on UpperCamelCase (nt AmlNotifierData) ning kõik meetodid, funktsioonid ja muutujad on lowerCamelCase (nt generateAmlReport) formaadis. Kõik meetodite nimed on tegusõnad, mis kirjeldavad meetodi sisu ning kõik klasside nimed, mis ei kutsu välja *back-end* teenust, on nimisõnad. Kõik meetodid, mis kutsuvad välja *back-end* moodulit, peavad olema sama nimega, mis vastav *back-end* moodul ning kirjeldama enda eesmärki. Lisaks peab nende klasside nimetuse lõpus olema alakriips ning number üks (nt GetAmlNotifications_1).

Igat meetodit või sektsiooni eraldab üks rida ning kood on trepitud tühikutega. Loogelised avavad sulud on kirjutatud ridade lõppu ehk reavahetus toimub pärast avavat loogelist sulgu. Iga sulgev loogeline sulg on uuel real ehk reavahetus toimub enne sulgevat loogelist sulgu, v.a siis, kui sellele järgneb *else* või *else if*. Kõik meetodid ja funktsioonid täidavad ainult ühte ülesannet. Mudeli klassides on objektide muutujad deklareeritud privaatselt. Koodi kirjutamisel on kasutatud võimalikult palju pärimist, et ei esineks koodi duplikatsiooni ning baasklassid ei teaks enda järglastest. Kõikide objekti muutujate välja kutsumisel on tehtud *null* kontroll, et ei tekiks *NullPointerExceptionit*. Kõikidel *UseCase* klassis defineeritud meetoditel on implementeeritud TechnicalErrorBase ja ValidationErrorBase meetodeid, mis valideerivad andmeid ning vajadusel tekitavad erindi. Lisaks on *back-end* moodulite välja kutsumisel kasutatud *try-catch* kontrolli, et kasutajale ei tekiks viga, kui *back-end'*ist ei jõua *front-end'*i korrektne informatsioon.

Koodi muudatuse *push'*imisel ja *commit'*imisel oli kohustus kirjutada kirjeldusse täpne Jira töö number.

Back-end'i arendamisel järgiti reeglit, et iga protseduur, klass ja moodul teeb ainult ühte asja. Protseduuri nimetamisel lähtuti protsessi sisust, et nimetuse järgi oleks võimalik teistel arendajatel aimata, mida antud moodul täpsemalt teeb. XSL-failide nimetamisel oli reegel, et kui tegu on trükise failiga, peab faili nimetus algama fo-algusega, näiteks fo-AMLNOTIFICATION. Kõikide muutujate nimetustes, mis on analoogsed andmebaasi väljadega, kasutati sidekriipsu. Ajutiste tabelite nimetused algavad tt-algusega, mis viitab *temp-table*'ile, näiteks ttAmlNotification. Protseduuri üleslaadimisel Giti pidi *commit'*i selgitusse lisama JIRA töö numbri, et oleks võimalik tuvastada, millise tööga antud muudatus seotud on.

Iga juurde arendatud meetodi ja ploki puhul kasutati *Check Syntax* käsku, mis kontrollib koodi Progress ABL süntaksi vastu. *Check Syntaxi* abil kontrolliti, kas kirjutatud kood on korrektne ning kompileerub. Kõikide andmebaasi otsingute ja erinevate tsüklite puhul kirjutati rea lõppu NO-LOCK ja NO-ERROR, mis tähendab, et antud andmebaasikirjet ei võeta lukku ning võimaliku vea korral ei jäeta mooduli kompileerimist pooleli. Kui arendaja soovis andmebaasi kirjet muuta, pidi kasutama EXCLUSIVE-LOCK käsku, mis võtab baasikirje lukku ja pärast töötlemist ja salvestamist vabastab selle. Samuti arendati kõikide andmebaasi otsingute puhul juurde veatöötlus olukorra jaoks, kui konkreetne andmebaas ei ole kättesaadav. Otsides objekti andmebaasist FIND käsuga, tuleb alati teha kontroll IF AVAILABLE, et ei tekiks viga objekti kasutamisel, kui seda ei peaks eksisteerima. FOR-EACH tsüklites kasutati NO-UNDO käsku, mis tähendab, et kui tsükli sees tekib mõnes protseduuris viga, siis ei lõpetata kogu tsükli tegevust.

4.1.5 Testid

AS SEB Panga arenduspoliitika ei näe ette, et arendajad peaksid arendatud mooduleid, protseduure, klasse, meetodeid või funktsioone ühiktestidega katma. Seetõttu ei ole AS SEB Pangas varasemalt projektide skoopidesse sisse arvestatud koodi ühiktestimist ja sellepärast neid igapäevaselt ka ei kirjutata. Antud rakenduse jaoks lisati töö ajakulu hinnangusse nii arenduse implementeerimisele kui ka ühiktestimisele kuluv aeg. Projekti esimeses etapis ühiktestimist ei teostatud, seega tuli ühiktestidega katta lisaks teise skoobi arendusele ka esimeses skoobis programmeeritud kood.

Kuna AS SEB Pangas kirjutati ühikteste esmakordselt nii NBSF kui ka NBSX keskkondades, siis tuli need esmalt seadistada ühiktestimist võimaldavaks. Lisaks kohustus meeskond uurima sügavuti JUnit ning ABL ühiktestimise dokumentatsioone, et mõista tehnoloogiaid ja omandada vajalikud teadmised ühiktestide koostamiseks.

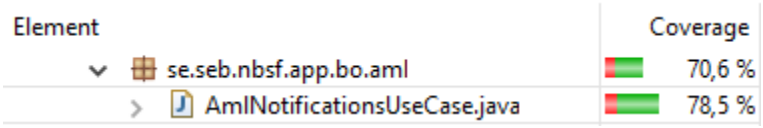
Antud projekti meeskond kirjutas ühikteste teatud määral paralleelselt arendustegevusega, kuid enamasti siiski pärast loogika ploki valmis arendamist. Põhjus seisnes selles, et äriarendusdivisjoni arendusjuhtide nõuded projektile muutusid pidevalt ning projekti meeskonna jaoks ei olnud mõistlik kirjutada esmalt ühikteste, sest sel juhul oleks rakenduse otsene arendamine ja tähtaeg edasi lükkunud. AS SEB Panga poolne nõue oli projekti tähtaegadest kinnipidamine ning *stand-up*'idel sooviti näha igapäevaselt rakenduse arenduste osas edasi liikumist.

*Back-end'*i arendusprotsessi jooksul kirjutati esimesena valmis konkreetne protseduur, mille funktsionaalsust testiti kõigepealt ABL Scratchpad'is. Protseduuri loogika plokk tõsteti käsitsi Scratchpadi, lisati talle sisendid ja väljundid ning testiti, kas antud sisendiga saadakse korrektne väljund. Seejärel koostati protseduurile samanimeline testklass *test* märkega, milles esmalt initsialiseeritakse protseduuri loogika ning seejärel jooksutatakse protseduuri andes sellele testsisendid. Ühiktesti jooksutatakse ABL ühiktesti raamistikuga ning arendajale kuvatakse teade testi õnnestumise kohta. Testi ebaõnnestumise korral kuvatakse põhjus, miks test läbi kukkus.

Back-end'is on kokku 19 testprotseduuri, milles kokku 76 testi. *Back-end*'is ei olnud võimalik lisada OpenEdge ABL platvormile laiendeid, seega ei saa esitada koodi katvuse meetrikat.

*Front-end'*i arendusprotsessi jooksul kirjutati esimesena valmis UseCase meetodid, mille funktsionaalsust testiti koheselt arenduskeskkonnas ning viimasena kirjutati meetoditele ühiktest. Algul kirjutati ühiktestid minimaalsel kujul, testimaks ainult objektide tekkimist vahemälusse, kuid hiljem lisati juurde ka objektide elementide korrektsuse testfunktsionaalsus.

Kokku on *front-end*'is kirjutatud 117 testi ning koodist ära kaetud 78,5 protsenti. (Joonis 38).



Joonis 38. Reakattuvus *front-end*'is.

4.2 Projekti teostamise põhjendus

Projekti teostamisel lähtus meeskond mitmete teadusartiklite ning raamatute õpetustest. Vajalikud raamatud saadi Tallinna Tehnikaülikooli raamatukogust ning *O'Reilly* andmebaasist. Temaatika ning metoodikaga seotud teadusartiklid leiti *Google Scholar* otsingumootori abil. Raamatute ja teadusartiklite läbi töötamise mõte oli saada kinnitust, et valitud metoodika, tööriistad ning arenduspoliitika on sobilikud.

Arhitektuuri loomisel kasutati relatsioonilist andmebaasi mudelit ning sellest aru saamiseks loeti ning analüüsiti raamatut "*Relational Database Design and Implementation, 4th Edition"*. Raamat annab ülevaate relatsioonilise andmemudeli koostamise, võimalike tööriistade ning kasutusvõimaluste kohta. (Harrington, 2016)

Raamatu õpetuse kohaselt on kõik andmebaasi tabelid unikaalsete pealkirjadega. AS SEB Panga andmebaasi tabeleid defineeritakse Progress OpenEdge *.df* failides ja tabelile, mis käsitleb teatise põhiandmeid pandi nimeks "MlNotification". Kõikide tabelite nimetused kirjeldavad nende sisu, näiteks andmebaasitabel, mis käsitleb AML teatise osapooli, nimetati "AmlParty". Ühegi andmebaasi tabeli pealkirja topelt ei kasutatud. Lisaks olid unikaalsed kõikide tabelite väljade nimetused.

Andmebaasi tabelid ühendati unikaalse primaarvõtme abil. Tabeli primaarvõti nimetati tabeli nimega sarnaselt ning teisele andmebaasi tabelile, mis oli esimese tabeliga ühendatud, määrati samanimeline võõrvõti. Kõik tabelid defineeriti konkreetse objekti jaoks, mis tähendas, et näiteks oli tabel tehingute jaoks ning oli tabel pangakontode jaoks.

Tulenevalt relatsioonilise andmemudeli disainimise eeskirjadest, koostati tabelid ning tabelite ühendused selliselt, et neid sai pidevalt muuta, luua juurde uusi ühendusi ning lisada juurde või kustutada andmetabeli väärtuseid selliselt, et rakenduse enda loogikat ei peaks muutma. Kui muudeti andmebaasi tabeli välja andmetüüpi, pidi muutma *back-end* ajutises tabelis samanimelist andmevälja andmetüüpi ning *front-end'*i objekti klassi.

Koodi kirjutamisel lähtuti *front-end*'is objektorienteeritud paradigmast. Klassid defineerivad objektide struktuuri ja elemendid ning rakenduse jooksutamisel luuakse vajalikud objektid. Objektid ehk *Java Bean*'id sisaldavad andmeid enda kohta. Objektorienteeritud programmeerimisel saab kasutada palju pärimist, mis aitab vähendada duplikatsiooni. Objekte salvestatakse *BeanList*'idesse, kus neid on lihtsam sorteerida ja pärida. (Dale, N., Joyce, D. T., Weems, C., 2016)

Progress OpenEdge ABL on protseduuriline programmeerimiskeel, mis sisaldab ka objektorienteeritud elemente. Progress OpenEdge platvormi kasutamisel lähtuti Progress Software Corporation kodulehekülje õpetustest ning AS SEB Panga poolt meeskonnale antud füüsilisest manuaalist. Protseduuriline programmeerimine tähendab, et kood on jaotatud erinevateks alamprotseduurideks, mis kutsutakse välja defineerimise järjekorras. Objektorienteeritud elementidest kasutasime *back-end*'i kirjutamisel pärimist, mis koos protseduuridega võimaldavad koodi lihtsasti taaskasutada. (Progress Software Corporation, 2017)

Koodi kirjutamisel lähtuti AS SEB Panga praktikatest ja Robert C. Martini raamatust "Clean Code". Esimeseks arendusreegliks võeti "*The Boy Scout Rule"* ehk "*Leave the campground cleaner than you found it."*, mis tähendas meeskonna jaoks seda, et toimus koodi pidev puhastamine ja refaktoreerimine. Teisena järgiti raamatus toodud nimetamise reegleid, näiteks kasutati tähenduslikke ja lihtsasti häädatavaid nimetusi, klasside nimed on nimisõnad ning meetodite nimed tegusõnad. Kolmandana järgiti reeglit, et iga funktsioon teeks ühte asja ning et kood ei korduks. Neljandana järgiti raamatus välja toodud vormistamise reegleid, näiteks deklareeriti muutujad võimalikult lähedal nende esmasele kasutamisele ning kõikide meetodite vahele lisati üks tühi rida. Viiendana kasutati vigade jaoks erindite kuvamist/viskamist. (Martin, 2008)

Raamatus välja toodud reeglitest peeti üldjuhul kinni, kuid teatud reeglite vastu siiski eksiti. Mõned *for-each* tsüklite tingimused olid ühel koodireal pikemad kui 120 tähemärki. Teatud protseduurides kirjeldatud meetoditele lisati sõnalised kommentaarid, et arendaja, kes antud moodulile enne testkeskkonda jõustamist koodile ülevaatuse teeb, saaks kiiresti aru, mida antud funktsioon, meetod tegema peab. (Martin, 2008)

Agiilse metoodika Scrum raamistik valiti meekonna poolt kahel põhjusel. Esmalt lähtuti sellest, et AS SEB Panga arendusdivisjonidel oli üleminekuperiood traditsioonilisest

arendusmetoodikast agiilse arendusmetoodika Scrum raamistikule ning seetõttu eeldati panga poolt, et arendusprojekti vältel kasutatakse samuti Scrum raamistikku. Teine põhjus, miks valiti just Scrum, oli see, et meeskond töötas läbi erinevaid teadusartikleid, mis kajastavad läbiviidud uuringuid ja võrdluseid projekti arendamise metoodikate kohta.

International Journal of Emerging Technology and Advanced Engineering poolt väljastatud artiklis "*Traditional SDLC Vs Scrum Methodology – A Comparative Study*" viidi läbi võrdlus traditsioonilise ja Scrum metoodika vahel. Teadusartikli põhjal on Scrumil 14 tugevust. Nendest peamised on tööaja optimeerimine ja nõudmistele reageerimine, et seeläbi tagada klientide rahulolu, kõrgem arenduse kvaliteet võrreldes traditsiooniliste arendusmetoodikatega ning erinevate muudatuste aktsepteerimine ja eeldamine arendusprotsessi vältel, mistõttu on implementeerimine efektiivsem ja projekti skoop võib pidevalt muutuda. (Mahalakshmi ja Sundararajan, 2013)

Teadusartikkel andis selge sisendi erinevate arendusmetoodikate vahel valimiseks ja artiklis loetletud Scrumi tugevused traditsiooniliste arendusmetoodikate ees osutusid valiku tegemisel määravaks. Projekti algstaadiumis ei olnud selge projekti kestus ja maksumus, sest arendajatel puudus arenduskogemus AS SEB Pangas ning kasutatavad tööriistad olid uued. Meeskonna jaoks oli oluline paindlik tööaeg, mis võimaldaks vajadusel koolis käia. Kuna varasemalt ei olnud AS SEB Pangas ükski arendusprojekt tehtud *full-stack* arendusmeeskonna poolt, siis võis eeldada, et algselt kirja pandud nõuded ja ideed võisid pidevalt muutuda ja seeläbi pidi olema võimalus pidevalt uusi muudatusi implementeerida. AS SEB Panga poolne mentor ja äriarenduse divisjoni arendusjuhid, kes nõuded kirja panid, soovisid iga päev ülevaadet projekti kohta ning Scrum raamistiku igahommikused *stand-up'*id olid sobiv lahendus ülevaate andmiseks.

Raamat "*Essential Scrum: A Practical Guide to the Most Popular Agile Process"* andis meeskonnale süvendatud teadmised agiilse arendusmetoodika Scrum raamistiku kohta, kuidas jaotada rollid, millised on rollidest tulenevad ülesanded, kuidas on jaotatud vastutus, milline on Scrumi puhul täpne tööprotsess ja kuidas jõustatakse tehtud arendus arenduskeskkonnast tootmiskeskkonda. Tulenevalt raamatust on Scrum meeskonnas toote omanik, *Scrum master* ning arendajad. (Rubin, 2012)

Raamatu õpetuste järgi jaotati ka projekti meeskond. *Scrum master* rolli sai AS SEB Panga poolne mentor Mihkel Matson, kelle vastutuseks jäi *stand-up* koosolekute

planeerimine ning tekkinud probleemide likvideerimine, mis takistasid arendustegevust. Tooteomanikuks määrati äriarendusedivisjoni arendusjuht Enel Pitk, kes koostas projektile arendusnõuded ning tema vastutada jäi tellijatega suhtlemine, uute ideede dokumenteerimine ja kasutajatestide planeerimine. Arendajateks said Triin Tammaru, Lilian Väli ning Carl-Robert Reidolf, kelle ülesandeks sai kogu rakenduse implementeerimine.

Agiilse arendusmetoodika Scrum raamistikku järgides tegutseti tööprotsessi vältel lähtuvalt raamatu juhistest. Esmalt alustati sprintide planeerimisest. Sprintide pikkuseks määrati kaks nädalat ning neid oli kokku kaheksa. Samuti jaotati kogu arendustegevus sprintide vahel koheselt. Iga päev toimusid hommikused 15-minutilised *stand-up'*id, kus iga tiimi liige vastas kolmele küsimusele - "Mida tegin eile?", "Mida teen täna?" ja "Kas on takistusi?". Seejärel jätkati arendustegevusega. Sprindi lõpus analüüsiti tehtut retrospektiivil, kus fikseeriti toimunu ning vajadusel liigutati tegemata jäänud osad järgmisesse sprinti. Toote *back-log*'i hoiti *Jira* tarkvaras ühe töö all väiksemate tükkidena ning kui arendaja sai oma ploki valmis, võttis suure töö alt järgmise väiksema ploki, mis oli sprinti planeeritud.

Rahapesukahtlaste tehingute teatiste rakendus, mis projektiga valmis arendati, edastab krüpteeritud teatised RAB-ile läbi andmevahetuskihi X-tee. Andmevahetuskiht X-tee on tehniline ja organisatsiooniline keskkond. See võimaldab turvalist ja tõestusväärtust tagavat internetipõhist andmevahetust riigiasutuste vahel erasektoriga. Antud projektis on riigiasutus RAB ning erasektori ettevõte AS SEB Pank. Rakendust arendanud meeskond omandas vajalikud teadmised X-tee liidestuse, andmete krüpteerimise, teabe edastamise ning vastuvõtmise kohta peamiselt Riigi Infosüsteemide Ameti koduleheküljelt ning AS SEB Panga koostatud dokumentatsiooni põhjal. Analüüsiti ning töötati läbi Riigi Infosüsteemide Ameti poolt koostatud X-tee rakendusjuhis. (Riigi Infosüsteemide Ameti koduleheküg, 2020)

4.3 Logid

Kokku kirjutati kolme peale 9466 rida koodi. Neist 6248 rida Java ja XHTML arenduskeeltes ning 3218 rida Progressis. Koodiread esitati tiimina, sest rakendati paarisprogrammeerimist ning vahel kirjutati koodi vaid ühe kasutaja alt. Järgnevalt on esitatud tiimi ühine logi nädalate kaupa.

4.3.1 Sprint 1

Tabel 1. Nädal 1 (25.06.2019 - 28.06.2019).

kp	Carl-Robert	Lilian	Triin
25.06	- Äripoolega projekti nõuete üle vaatamine ja uute arenduste kooskõlastamine.	- Äripoolega projekti nõuete üle vaatamine ja uute arenduste kooskõlastamine.	- Äripoolega projekti nõuete üle vaatamine ja uute arenduste kooskõlastamine.
26.06	- Saadetava XML-faili struktuuri ümber disainimine, topelt tulevate emailide ja telefoninumbrite bugi parandamine. - X-tee dokumentatsiooni läbi töötamine.	- Lisaajendi nupu liigutamine paremalt rippmenüü kõrvale nii, et see liigub sellega väljaga koos. - Seotud isikute nuppude paigutus. - ZKoss dokumentatsiooni lugemine. -Ühiktestide raamistiku üles seadmine	- Osapoolte lisamisel AmlPartiesListi ja teate salvestamisel tekkivate *NullPointerException'*ite debugimine ja parandamine. - Kohustuslike väljade täitmise kontroll ja teated.
27.06	- XML-faili struktuuri loomise teenuse veatöötlus ning vajalike error staatuste lisamine. - XML-faili kontroll XSD-faili vastu.	- Kontrollid, kas lisatavad seotud isikud on juriidilised või füüsilised, et kusagil ei tekiks *NullPointerException'*it registrikoodi/isikukoodiga. -Ühikstestide raamistiku üles seadmine	- Eesnime ja perekonnanime tabelis kuvamine täisnimena. - Lisaajendi listi tühjaks tegemine loogika kui muudetakse põhiajendit. - Ühiktestide raamistiku üles seadmine
28.06	- Teatise XML-faili koostamise, krüpteerimise ja edastamise esimene testimine RAB-iga.	- Kontroll, et IBAN oleks õiges formaadis. - Pankade nimede kontroll. - Ühiktestid eelnimetatu jaoks.	- Sünni- ja surmaaja, isikukoodi ja nimede korrektsuse kontrollid. - Ühiktestid eelnimetatu jaoks.

Tabel 2. Nädal 2 (01.07.2019 - 05.07.2019).

kp	**Carl-Robert**	**Lilian**	**Triin**
01.07	- Tulemüüri, turvaserveri konfigureerimine RAB-i vastuse jaoks. - Serverilt faili lugemine.	- Bugi parandamine: tehingute kogusumma arvutamise loogika valuuta põhiselt. - Ühiktestide kirjutamine	- Bugi parandamine: tehingute kogusumma arvutamise loogika valuuta põhiselt. - Ühiktestide kirjutamine
02.07	- RAB-i vastuse töötlemiseks vajalike tabelite, moodulite programmeerimine. - X-tee dokumentatsiooni läbi töötamine.	- Vanemarendajaga kohtumine, et teha osapoolte leidmine automaatseks. - *Front-end*'is sarnaste arenduste otsimine ja analüüsimine. - X-tee dokumentatsiooni läbi töötamine.	- Vanemarendajaga kohtumine, et teha osapoolte leidmine automaatseks. - *Front-end*'is sarnaste arenduste otsimine ja analüüsimine. - X-tee dokumentatsiooni läbi töötamine.
03.07	- RAB-i vastuse töötlemiseks vajalike tabelite, moodulite programmeerimine.	- Hüpikakna vaate loomine klientide otsimiseks. - *Query* loomine, kuhu salvestatakse otsitav fraas, millega kutsutakse välja *back-end* moodul.	- *Back-end* mooduli loomine klientide otsimiseks andmebaasist. - Vajalike baasi tabelite otsimine.
04.07	- RAB-i vastuse moodulitesse sisse lugemine. - Logidesse RAB-i vastuse laekumise kohta kirjete loomine.	- *Front-end*'i ja *back-end*'i ühildamine osapoolte jaoks. - READ-XML errorid *front-end*'is. - Ühiktestide kirjutamine	- *Back-end* mooduli täiendamine klientide otsimiseks andmebaasist. - Ühiktestide kirjutamine
05.07	- RAB-i vastuse *front-end*'is kuvamise ja töötlemise klasside programmeerimine.	- Errorite üle vaatamine vanemarendajaga. - Juba olemasolevate osapoole otsingute ülevaatus.	- Errorite üle vaatamine vanemarendajaga. - Juba olemasolevate osapoole otsingute ülevaatus.

4.3.2 Sprint 2

Tabel 3. Nädal 3 (08.07.2019 - 12.07.2019).

kp	Carl-Robert	Lilian	Triin
01.07	- RAB-i vastuse *front-end*'is kuvamise ja töötlemise klasside programmeerimine. - Retrospektiivil osalemine.	- *Front-end*'i vajaliku vaate rakendamine, et tekiks vahemälusse mudeli bean andmebaasist klientide otsimiseks. - Retrospektiivil osalemine.	- Osapoole andmete õigesse mudelisse ringi salvestamine *back-end* teenuse abil. - Osapoolte tabeli struktuuri muutmine. - Retrospektiivil osalemine.
02.07	- RAB-i vastuse edastamise teenuse arendamine *back-end*'ist *front-end*'i.	- Klientide otsingu lisamine osapoole, seotud isiku ning esindaja ja volitaja lehele. - Setup korrastus (koodide muutmine ja kokku leppimine äripoolega).	- Osapoolte delete loogika. - *Edit* loogika näidete otsimine ja katsetamine. - Setup korrastus (koodide muutmine ja kokku leppimine äripoolega).
03.07	- RAB-i vastuse edastamise teenuse arendamine *back-end*'ist *front-end*'i. - Arendatud teenuse testimine äripoolega.	- Tehingu lisamise otsimine *front-end'*ist, et näha ja analüüsida olemasolevaid arendusi.	- Osapoolte *edit* loogika, uute *edit* klasside loomine. - Tehingud tabeli kirjetele *edit* ja *delete* lisamine. - Ühiktestid *edit* ja *delete* jaoks.
04.07	- RAB-i vastuse edastamise testimisel esinenud vigade parandamine. - RAB-i vastust töötleva *back-end* mooduli ühiktestimine	- Tehingu *live* süsteemist otsimiste arutlus NBSF kapteniga ning näidete analüüsimine. - Tehingu liigi peitmine, kuna see on alati summa.	- *Edit* loogika muutmine, juba olemasolevaid *data* ja zul klasse kasutades. - Dokumentide tabeli kirjetele *edit* ja *delete* lisamine. - Ühiktestide kirjutamine
05.07	- Postman päringu loomine ja tegemine üle X-tee RAB-i API vastu ning saadud väärtuste ümber salvestamine Excel failidesse .csv laiendiga.	- Tehingu vaatele nupu "Otsi tehingut" lisamine. - Vaate tegemine, kus saab valida rippmenüüst osapoolt. - Kõikide olemasolevate osapoolte vahemälust otsimine ning lisamine LOV listi, et kuvada rippmenüüs nimesid.	- Kõikidele erinevatele dokumendiliikide (SWIFT, väljavõte, ID) dokumentide lisamise tabeli edit ja delete. - Uute osapoolte, tehingute, dokumentide lisamisel teha tühjaks lahtrid, kui *edit* režiimis on lisamine pooleli jäetud.

Tabel 4. Nädal 4 (15.07.2019 - 19.07.2019).

kp	Carl-Robert	Lilian	Triin
01.07	- Rippmenüüde väärtuste kuvamiseks kasutajaliideses vajaliku skripti arendamine.	- Rippmenüüde väärtuste kuvamiseks kasutajaliideses vajaliku skripti arendamine (testimine kas kõik on failides õigelt olemas). - Kontode kuvamine tabelis valitud osapoolele tehingu otsingul.	- Rippmenüüde väärtuste kuvamiseks kasutajaliideses vajaliku skripti arendamine. - Kontodele linnutamise lahtrite lisamine, kõik kontod vaikimisi valitud.
02.07	- Rippmenüüde väärtuste kuvamiseks kasutajaliideses vajaliku skripti arendamine. - Äripoolele rippmenüüde demonstreerimine ja testimine.	- Rippmenüüde väärtuste kuvamiseks kasutajaliideses vajaliku skripti arendamine. - Äripoolele rippmenüüde demonstreerimine ja testimine. - Sisseehitatud kontode otsimise funktsionaalsuse kasutamine, kus on ka muud täpsustused. - Ühiktestide kirjutamine	- Rippmenüüde väärtuste kuvamiseks kasutajaliideses vajaliku skripti arendamine. - Äripoolele rippmenüüde demonstreerimine ja testimine. - Ühiktestide kirjutamine
03.07	- Rippmenüüde väärtustamise skripti täiendamine. - Teatiste tabelitesse jaotamise loogika kooskõlastamine äripoolega - *back-end'*i ühiktestimine	- Rippmenüüde väärtustamise skripti täiendamine. - Kontode otsimise lehelt vajaliku info vahemällu lugemine. - Tehingu lehel tabeli loomine.	- Rippmenüüde väärtustamise skripti täiendamine. - Kõikidele erinevatele dokumendiliikide (SWIFT, väljavõte, ID) dokumentide lisamisel teha tühjaks lahtrid, kui edit režiimis on lisamine pooleli jäetud.
04.07	- Rippmenüüde väärtustamise skripti jõustamine läbi testkeskkonna *Live*'i.	- Rippmenüüde väärtustamise skripti jõustamine läbi testkeskkonna *Live*'i. - Tehingute otsimine *live* süsteemist.	- Rippmenüüde väärtustamise skripti jõustamine läbi testkeskkonna *Live*'i. - Tõlgete lisamine tehingute tabelile.
05.07	- Loogika arendamine, et kasutajaliideses teatised jaotuksid tabelitesse. - *Back-end'*i ühiktestimine	- Tehingute otsimise *live* süsteemist erindite parandamine. - Ühiktestide kirjutamine	- Kustutatud tehingute, osapoolte, dokumentide uuesti ilmumise vigade parandused. - Ühiktestide kirjutamine

4.3.3 Sprint 3

Tabel 5. Nädal 5 (22.07.2019 - 26.07.2019).

kp	Carl-Robert	Lilian	Triin
01.07	- Loogika arendamine, et kasutajaliideses teatised jaotuksid tabelitesse. - Teatistele staatuste lisamise protseduuri arendamine.	- Äripoolega tehingute tabeli ülevaatamine. - Tabeli päiste muutmine. - Valitud tehingu info salvestamine õigesse mudelisse ning algsele lehele tagasi suunamine.	- Tõlgete lisamine *edit* ja *delete* nuppude jaoks. - Äripoolega arutamine, mida teha ID-koodidega kui osapoolel see puudub. - Erinevate dokumentide tabelite struktuuri muutused.
02.07	- Loogika arendamine, et kasutajaliideses teatised jaotuksid tabelitesse. - Teatiste staatuste järgi tabelitesse salvestamise testimine.	- Äripoolega tehingute ploki ülevaatus. - Paigutuse muutmine nuppude osas. - LOV listide ühendamine tehingu otsinguga. - Ühiktestide kirjutamine	- Äripoolega tehingute ploki ülevaatus. - Unikaalsete ID-de loomine, kui isikul pole identifitseerimise koodi. - Ühiktestide kirjutamine
03.07	- Teatise eelvaate XSL-faili arendamine.	- Unikaalsete ID-de loomine, kui isikul pole identifitseerimise koodi *front-end'*is. - Ühiktestide kirjutamine	- Unikaalsete ID-de loomine, kui isikul pole identifitseerimise koodi *front-end'*is. - Ühiktestide kirjutamine
04.07	- Teatise eelvaate XSL-faili arendamine. - Taotluse koostamine ja edastamine RAB-i, et AS SEB Pank liidestada RAB-i *Live* teenusega	- Unikaalsete ID-de loomine, kui isikul pole identifitseerimise koodi. - Ühiktestid unikaalse ID jaoks.	- Unikaalsete ID-de loomine, kui isikul pole identifitseerimise koodi. - Ühiktestid unikaalse ID jaoks.
05.07	- Teatise eelvaate XSL-faili arendamine.	- Unikaalsete ID-de loomise loogika ümber paigutamine *back-end'*i, ühiktestidest ilmnenud vigade alusel.	- Unikaalsete ID-de loomise loogika ümber paigutamine *back-end'*i, ühiktestidest ilmnenud vigade alusel.

Tabel 6. Nädal 6 (29.07.2019 - 02.08.2019).

kp	**Carl-Robert**	**Lilian**	**Triin**
01.07	- Teatise eelvaate XSL-faili arendamine. - Retrospektiivil osalemine.	- .df failide loomine uute tabelikirjete jaoks ning nende ülevaatus arhitektiga. - Retrospektiivil osalemine. - Teatise eelvaate tagasisidestamine.	- .df failide loomine uute tabelikirjete jaoks ning nende ülevaatus arhitektiga. - Retrospektiivil osalemine. - Teatise eelvaate ülevaatus tagasisidestamine.
02.07	- Teatise eelvaate XSL-faili täiendamine. - Teatise eelvaate loomiseks vajaliku *back-end* protseduuri programmeerimine.	- *NullPointerException*'i parandamine: esindaja lisamisel juba lisatud osapoolte hulgast. - Vea parandamine, et tehingu valimisel väljavõtte kaudu info salvestuks.	- .df failiga *changed-by, changed-when, closed-by, closed-when* väljade lisamine - Lisatud väljadele loogika arendamine, mille alusel neid täidetakse. - Ühiktestide kirjutamine
03.07	- Teatise eelvaate loomiseks vajaliku *back-end* protseduuri programmeerimine.	- Tehingu lisamise *NullPointerException'*i parandamine. - Ühiktestide kirjutamine	- Tehingu lisamise *NullPointerException'*i parandamine. - Ühiktestide kirjutamine
04.07	- Teatise eelvaate loomiseks vajaliku *back-end* protseduuri programmeerimine. - Eelvaate jaoks vajalike ajutiste tabelite loomine *Include-i* faili.	- Vea parandamine, et EUR summa oleks kahe eraldi reana kogusummade tabelis. - Ühiktestide kirjutamine	- Teatistele kasutajaõiguste lisamise loogika arendamine, et mitu kasutajat ei saaks korraga ühte teatist modifitseerida. - Ühiktestide kirjutamine
05.07	- Teatise saatmisel tekkinud *Live* bugi parandamine - Teatise eelvaate *back-end'*i mooduli ühiktestimine	- Failide sisse lugemine *front-end'*i baitidena. - Seotud isiku listi parandamine - Ühiktestide kirjutamine	- Teatistele kasutajaõiguste lisamise loogika arendamine, et mitu kasutajat ei saaks korraga ühte teatist modifitseerida. - Arenduse testimine äripoolega - Ühiktestide kirjutamine

4.3.4 Sprint 4

Tabel 7. Nädal 7 (05.08.2019 - 09.08.2019).

kp	Carl-Robert	Lilian	Triin
01.07	- Teatise eelvaate loomiseks vajaliku *back-end* protseduuri programmeerimine - Teatise eelvaate loomise *back-end* protseduuri ühiktestimine.	- Failide sisse lugemine ning *back-end* mooduli käivitamine. - Mitme faili korraga lisamise võimaluse uurimine	- Unikaalsete ID-de korrastus. - *Attached* failidele delete funktsionaalsuse lisamine ning tabelis kuvamine.
02.07	- Teatise eelvaate kuvamiseks vajaliku *front-end* teenuse arendamine.	- *Live* süsteemis kastide suuremaks lohistamise võimalusele alternatiivi arendamine. Tehtu üle kirjutamine RDD *commit'*iga. - Muu disaini ülevaatus ja nuppude paigutus. - Mitme faili sisse lugemise arendus	- Unikaalsete ID-de korrastus. - Kasutuses teadete loogika täiendamine ning tekkinud vigade parandamine. - Ühiktestide kirjutamine
03.07	- Teatise eelvaate kuvamiseks vajaliku *Front-end* teenuse arendamine. - PDF-faili genereermise loogika arendamine.	- Mitme faili sisse lugemise arendus - Ühiktestide kirjutamine	- Osapoolte järjekorra loogika, uue välja lisamine .df failiga. - Ühiktestide kirjutamine
04.07	- Teatise eelvaate kuvamiseks vajaliku *Front-end* teenuse arendamine. - PDF-fail genereermise loogika arendamine.	- Osapoolte järjekorra üle vaatamine äripoolega. - Osapoolte tabelis esindaja, volitaja ja seotud isiku nime ja rolli koos kuvamine.	- Osapoolte järjekorra üle vaatamine äripoolega. - Osapoolte tabelis esindaja, volitaja ja seotud isiku nime ja rolli koos kuvamine.
05.07	- Teatise eelvaate XSL-faili ümber disainimine auditeerimise dokumendi nõuete järgi. - PDF-faili genereerimise veatöötluse arendamine.	- Osapoolte järjekorra loogika veaparandus - viga tekkis osapoolte kustutamisel.	- Osapoolte järjekorra loogika veaparandus - viga tekkis osapoolte kustutamisel.

Tabel 8. Nädal 8 (12.08.2019 - 16.08.2019).

kp	**Carl-Robert**	**Lilian**	**Triin**
01.07	- Teatise eelvaate genereerimise testimine äripoolega. - Teatise eelvaate XSL-faili täiustamine.	- Automaatne pangatuvastus, äripoolega vajalike pankade arutamine.	- Ebavajalike väljade kustutamine .df failiga. - Teatise eelvaate genereerimise testimine äripoolega.
02.07	- Teatise saatmisel võimalike ebakorrektsuste korral tekkinud vigade kuvamise *back-end* loogika arendamine.	- Automaatne pangatuvastus, arhitektiga õigete setupide arutamine.	- Setup kirjete uuendamine. - Äripoole nõuete analüüs, edasise plaani koostamine tegemata asjade põhjal.
03.07	- Teatise saatmisel võimalike ebakorrektsuste korral tekkinud vigade kuvamise *back-end* loogika arendamine. - Teatise vigaste väljade kuvamise vaadete arendamine *front-end'*is.	- Automaatne pangatuvastus *back-end* mooduli loomine, mis otsib IBANI järgi õige panga.	- Lisakontrollid, juriidilise isiku lisamisel esindaja/volitaja kontroll. Failide lisamisel kontroll, objektide kustutamisel kontroll. Osapoolte lisakontrollid.
04.07	- Teatise saatmisel võimalike ebakorrektsuste korral tekkinud vigade kuvamise *back-end* loogika arendamine. - Teatise vigaste väljade kuvamise loogika arendamine *front-end'*is.	- Automaatne pangatuvastus *front-end* mooduli loomine, mis salvestab olemasoleva *bean'*i üle baasidest leitud infoga. - Ühiktestide kirjutamine pangatuvastusele *front*	- Kontrollidele tõlked. - Uued kollased lehed, vajalike muudatuste *code review'*sse saatmine. - Unikaalse ID vea parandamine. - Ühiktestide kirjutamine pangatuvastusele *back*
05.07	- Teatise saatmise, võimalike vigade kuvamise, RAB-i vastuse kuvamise, teatise eelvaate genereerimise testimine RAB-i ja äripoolega.	- Saadetud teatise kommentaari vaate loogika *front-end'*is. Uue tabeli nupu loomine, uue vaate loomine kommentaari jaoks ja RDD *commit*, et see oleks kast teatud mahuga.	- Saadetud teatise kommentaari *back-end*, kommentaari andmebaasi salvestamine ja uue tabeli välja loomine df failiga.

4.3.5 Sprint 5

Tabel 9. Nädal 9 (19.08.2019 - 23.08.2019).

kp	Carl-Robert	Lilian	Triin
01.07	- Teatise saatmise, võimalike vigade kuvamise, RAB-i vastuse kuvamise, teatise eelvaate genereerimise testimine RAB-i ja äripoolega. - Retrospektiivil osalemine.	- Saadetud teatise kommentaar *front-end, back-end'*iga ühenduse saamine. - Retrospektiivil osalemine. - Ühiktestide kirjutamine	- Saadetud teatise kommentaar *back-end,* kommentaari baasist kätte saamine ja *front-end'*ile tagastamine. - Retrospektiivil osalemine. - Ühiktestide kirjutamine
02.07	- RAB-i vastuse töötlemisel tekkivate *nullpointer'*ite parandamine. - RAB-i vastuse *back-end* protseduur mooduli ühiktestide programmerimine.	- Isikukoodide asemel nimede kuvamine tehingute tabelis. - *NullPointerException* kontrollid kõigele *UseCase*'is mida kood püüab *get*'ida.	- Juriidilise isiku kellelt - kellele väärtustamine. - Saadetud teatise kommentaari ühiktestid. - Kasutajapoolsed kommentaari testid.
03.07	- Teatise saatmise *back-end* protseduuride ühiktestimine	- Muud valuutat ei kuva tehingute otsimisel - võimaliku vea otsimine. - Vea arutamine ning võimalike lahenduste presenteerimine äripoolele.	- Juriidilise isiku kellelt - kellele väärtustamine. - Rakenduse tegevuste logide loogika. - Ühiktestide kirjutamine
04.07	- Teatise saatmise *back-end* protseduuride ühiktestimine	- PDF-faili korrastamine, korrektselt summa kuvamine - *Front-end* moodulite ühiktestimine	- PDF-faili korrastamine, korrektselt summa kuvamine. - Rakenduse tegevuste logide loogika.
05.07	- *Front-end* moodulite ühiktestimine - PDF-faili korrastamine, äripoolega arutelu	- PDF-faili korrastamine, kastide paigutamine, äripoolega arutelu - *Front-end* moodulite ühiktestimine	- PDF-faili korrastamine, äripoolega arutelu. - *Front-end* moodulite ühiktestimine

Tabel 10. Nädal 10 (26.08.2019 - 30.08.2019).

kp	Carl-Robert	Lilian	Triin
01.07	- Teatise eelvaate kuvamise protseduuri veatöötluse loogika arendamine - PDF-faili korrastamine	- Raporti loomine. *Front-end'*is uue *UseCase* loomine ning uue menüüpunkti loomine.	- Raporti loomine, *back-end* moodul, mis loeb sisse XML-ina kaks kuupäeva ning aktiivse kasutaja cif-koodi.
02.07	- Rippmenüüde väärtuste muutumisel RAB-i poliitilise otsuse tagajärjel uue Postman päringu sooritamine ning rippmenüüde väärtuste muutmine - Teatise eelvaate kooskõlastamine äripoole ja juristiga	- Raporti loomine. *Front-end'*is uus mudel, mis täidetakse kasutaja poolt kuupäevade valimisel.	- Raporti loomine. *Back-end ProcessService* meetod, mis hakkab õigeid teateid striimima dokumenti.
03.07	- Teatiste staatuse lisamisel tekkinud vea parandamine ja staatuste lisamise *back-end* protseduuri ühiktestimine - df failide täiendamine	- Raporti loomine, äripoolega koosoleku läbiviimine, et välja selgitada, mida soovitakse raportis täpselt näha ning defineerida statistiliselt olulised parameetrid.	- Raporti loomine, äripoolega koosoleku läbiviimine, et välja selgitada, mida soovitakse raportis täpselt näha ning defineerida statistiliselt olulised parameetrid.
04.07	- Raporti loomine, *processService* loogika mõtlemine, kuidas andmeid CSV-failis kuvada.	- Raporti loomine, *processService* loogika mõtlemine, kuidas andmeid CSV-failis kuvada. - *Front-end* moodulite ühiktestimine	- Raporti loomine, *processService* loogika mõtlemine, kuidas andmeid csv-failis kuvada. - Ühiktestide kirjutamine
05.07	- Erinevate *back-end* ja *front-end* moodulite ühiktestide kirjutamine	- Vea parandamine - Osapoole kontode tabelile nupu lisamine, et otse dokumenti lisada. - *Front-end* moodulite ühiktestimine	- Raporti loomine - Ühiktestide kirjutamine

4.3.6 Sprint 6

Tabel 11. Nädal 11 (02.09.2019 - 06.09.2019).

kp	Carl-Robert	Lilian	Triin
04.09	- Refaktoorimine - Disain	- Refaktoorimine, meetodite väiksemaks muutmine - Disain	- Refaktoorimine, meetodite väiksemaks muutmine - Disain
05.09	- Refaktoorimine - Disain	- Refaktoorimine - Disain	- Refaktoorimine - Disain

Tabel 12. Nädal 12 (09.09.2019 - 13.09.2019).

kp	Carl-Robert	Lilian	Triin
11.09	- Refaktoorimine - Retrospektiivil osalemine	- Refaktoorimine - Retrospektiivil osalemine. - *Front-end* moodulite ühiktestimine	- Refaktoorimine - Retrospektiivil osalemine. - *Back-end* moodulite ühiktestimine
12.09	- Refaktoorimine - *Front-end* moodulite ühiktestimine	- Refaktoorimine - *Front-end* moodulite ühiktestimine	- Refaktoorimine - *Back-end* moodulite ühiktestimine

4.3.7 Sprint 7

Tabel 13. Nädal 13 (16.09.2019 - 20.09.2019).

kp	Carl-Robert	Lilian	Triin
18.09	- Funktsionaalsuse testimine - *Back-end* moodulite ühiktestimine - Muudatuslehed	- Funktsionaalsuse testimine - *Back-end* moodulite ühiktestimine - Muudatuslehed	- Funktsionaalsuse testimine - *Back-end* moodulite ühiktestimine - Muudatuslehed
19.09	- Funktsionaalsuse testimine - Muudatuslehtede koostamine	- Funktsionaalsuse testimine - Muudatuslehtede koostamine	- Funktsionaalsuse testimine - Muudatuslehtede koostamine

Tabel 14. Nädal 14 (23.09.2019 - 27.09.2019).

kp	Carl-Robert	Lilian	Triin
02.09	- Testimine RAB-iga	- Testimine RAB-iga	- Testimine RAB-iga
03.09	- Testimine RAB-i, äriarendusjuhtide ja kasutajatega	- Testimine RAB-i, äriarendusjuhtide ja kasutajatega	- Testimine RAB-i, äriarendusjuhtide ja kasutajatega

4.3.8 Sprint 8

Tabel 15. Nädal 15 (30.09.2019 - 04.10.2019).

kp	Carl-Robert	Lilian	Triin
02.09	- Testimine äripoolega - Retrospektiivil osalemine.	- Testimine äripoolega - Retrospektiivil osalemine.	- Testimine äripoolega - Retrospektiivil osalemine.
03.09	- Testimine äripoolega	- Testimine äripoolega	- Testimine äripoolega

Tabel 16. Nädal 16 (07.10.2019 - 11.10.2019).

kp	Carl-Robert	Lilian	Triin
09.10	Testimine	Testimine	Testimine
10.10	Testimine ja *Live*'i jõustamine	Testimine ja *Live*'i jõustamine	Testimine ja *Live*'i jõustamine

4.4 Hinnang projekti teostamise protsessi kohta

4.4.1 Projekti juhtimise ja teostamise protsess

Lõputöö arendamine toimus 16 nädala jooksul. Arendustööga alustati 25. juunil 2019 ning lõpetati 10. oktoobril 2019. Esimesed kümme nädalat tegeleti projektiga viiel päeval ning sellele järgneval kuuel nädalal kahel päeval nädalas. Kõik tööpäevad kestsid

vähemalt 8 tundi, seega töötas tiim esimesel kümnel nädalal nädalas kolme peale kokku ligikaudu 120 tundi ning viimasel kuuel nädalal 48 tundi. Kokku teeb see 16 nädala peale 1488 tundi tööd. Lisaks tegeleti projekti kõrvalt Progress OpenEdge ning Java oskuste arendamisega.

Projekti mentor oli AS SEB Panga poolt IT *Team Lead* Mihkel Matson. Lisaks, iga valdkonna vanem-tarkvaraarendajad: *front-end* poole pealt Jaanus Türnpuu ning *back-end* poole pealt Toomas Kask. Kõikide juhendajatega toimus iga tööpäeva hommikul *stand-up* kohtumine, mis kestis 15 minutit. Samuti jagati ära päeva ülesanded. Selgitati koos äriarenduse divisjoni esindajatega välja, mis on vaja järgmisena teha ning otsustati, kes seda teha võiks. Kuna rakendati osaliselt paarisprogrammeerimist, siis oli ülesandeid lihtsam jagada.

4.4.2 Hinnang projektile

Projekti protsessi puhul oli kitsaskohti pigem vähe, kuid üks peamiseid puudutas projekti viimaseid nädalaid, kui arendajad ei saanud kooli tunniplaani järgi täiskohaga projekti arendada.

Teatud juhtudel tehti siiski ületunde. Teine peamine kitsaskoht puudutas *live* keskkonna õiguseid. Arendajate ametinimetus AS SEB Pangas projekti vältel oli *Junior Software Developer* ning tulenevalt vähesest tööstaažist ei väljastatud arendajatele *live* keskkonna õiguseid, mis omakorda tähendas, et erinevate vigade ilmnemisel *live* keskkonnas oli keeruline neid tuvastada ja fikseerida. Seetõttu võttis vigade ilmnemisel nende parandamine rohkem aega, sest kaasata tuli vanemarendajaid, kes saaksid vaadata logidest, kus ning mis põhjusel viga esineb.

Üldjoontes sujus kogu protsess edukalt, mida kinnitab *Team Lead'*ilt ning äriarenduse divisjoni arendusjuhilt saadud positiivne tagasiside (Lisa 5). Ettenähtud aja jooksul sai planeeritud mahus projekti valmis ning õigeks ajaks kogu funktsionaalsus testitud. Meeskonna ja projektiga seotud isikute omavaheline suhtlus oli hea, saadi kõikidele tekkinud küsimustele kiirelt vastused, aidati üksteist ning tagasiside oli vahetu ja konstruktiivne.

4.4.3 Hinnang üldisele protsessile

Projekti ülikoolipoolne juhendaja oli Tallinna Tehnikaülikooli vanemteadur Tõnn Talpsepp ning projekti AS SEB Panga poolne mentor oli IT *Team Lead* Mihkel Matson. Koostöö nii ülikooli- kui ka pangapoolse juhendajaga oli konstruktiivne ja hea. Informatsiooni vahetus käis pidevalt ja tekkinud muredega oli võimalik projekti meeskonnal mõlema juhendaja poole pöörduda.

Töökorraldus oli paindlik ja selge, sest projekti alguses lepiti AS SEB Panga poolse mentoriga kokku tööajad ning kooskõlastati kõik projekti korraldust puudutavad aspektid tööruumide, tööriistade, keskkondade ja õiguste osas. Vormistati töölepingud, kuid seoses tiheda tunniplaaniga jäeti tudengitele võimalus igal hetkel, ka kokkulepitud tööajal, vajadusel õpinguteks ülikooli minna. Paindlik töökorraldus soosis efektiivset arendustegevust, sest nii ei kannatanud õppetegevus koolis ja tundes pangapoolset toetavat suhtumist oli meeskonnal isiklik huvi arendustegevust ka töövälisel ajal jätkata.

Meeskonnaliikmete omavaheline läbisaamine oli hea ja positiivne. Kuna antud projekti liikmed on varasemalt ühiselt arendusprojekte teinud nii koolis kui ka koolivälselt, teati üksteise tugevusi ja nõrkuseid nii oskuste kui ka iseloomu koha pealt, mis tõttu tagasilööke ega konflikte ei esinenud. Kõik meeskonnaliikmed panustasid võrdselt ja kõigile anti võimalus arendada *front-end'*i ja *back-end'*i.

4.5 Meeskondlik hinnang

Skaalal "-2" kuni "+2" hinnati meeskondlikult kõikide meeskonnaliikmete panust ja osalemist tööprotsessis hindega "0" ehk "panustas samaväärselt". Kõik täitsid hommikustel *stand-up*'idel jagatud ülesanded ning olid valmis panustama projekti arengusse ka töövälisel ajal. Etteheiteid üksteise suunal ei esinenud.

5 Kokkuvõte

Lõputöö projekti jooksul teostati AS SEB Pangale rahapesu kahtlusega teatiste edastamise rakendusele edasiarendused ning täiendused. Projekti meeskonda kuulusid kolm äriinfotehnoloogia kolmanda kursuse tudengit Lilian Väli, Triin Tammaru ning Carl-Robert Reidolf. Varasemalt olemas olnud arendus ei olnud piisavalt efektiivne ja kasutajasõbralik ning ühe teatise täitmine oli ajamahukas. Seetõttu olid lõputöö eesmärgid automatiseerida teatise koostamist, parendada olemasolevat arendust ning täiendada rakendust vajalike lisadega.

Agiilset arendusmetoodikat kasutades valmisid 16 nädala jooksul tudengite töö tulemusel rakendusele vajalikud edasiarendused ning täiendused. *Back-end'*i kirjutamiseks kasutati Progress OpenEdge platvormi ja OpenEdge ABL programmeerimiskeelt. *Front-end'*i kirjutati Eclipse platvormil Java arenduskeeles ning graafilise kasutajaliidese konstrueerimiseks kasutati ZK raamistikku.

Lõputööga valmis uuenenud rakendus, milles täidetakse vajalikud väljad automaatselt andmebaasidest kogutud infoga. Arendusega lisati kasutajatele võimalus laadida alla teatise eelvaate formaalne PDF-fail ning raport. Lisaks tehti rakendusele kasutajate nõuetele vastavaid täiendusi ning parandati UX ning UI disaini.

Projekti sisuline loogika ning piirangud lähtusid eelkõige RAB-i poolt meeskonnale antud juhendist ning AS SEB Panga poolt kasutusel olevatest raamistikest. Meeskonnaliikmete, IT *Team Lead*'i, äriarendusejuhi ning *Compliance* osakonna esindaja ühisel otsusel hinnati projekt edukaks. Lõputöö tulemusena said kõik püstitatud nõuded täidetud ning rakendus lisati vastavuskontrolli töötajate tööriistade hulka.

Kasutatud kirjandus

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

[2] N. B. Dale, D. T. Joyce ja C. Weems, Object-Oriented Data Structure Using Java, 4th Edition, Burlington: Jones & Bartlett Learning, 2016.

[3] D. Harned, Creating and Running an Agile Project in JIRA., Birmingham: Packt Publishing, 2018.

[4] J. L. Harrington, Relational Database Design and Implementation, 4th Edition, Burlington: Morgan Kaufmann, 2016.

[5] M. Mahalakshmi ja M. Sundararajan, „Traditional SDLC Vs Scrum Methodology - A Comparative Study," *International Journal of Emerging Technology and Advanced Engineering*, kd. 3, nr 6, pp. 192-196, 2013.

[6] J. Muli ja A. Okoth, Jenkins Fundamentals, Birmingham: Packt Publishing, 2018.

[7] R. C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design, First Edition., New Jersey: Prentice Hall, 2017.

[8] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, New Jersey: Prentice Hall, 2008.

[9] „Riigi Infosüsteemide Amet," 2019. [Võrgumaterjal]. Available: https://www.ria.ee/et/riigi-infosüsteem/x-tee. [Kasutatud 11 04 2019].

[10] Progress Software Corporation, „Progress OpenEdge Documentation," 2020. [Võrgumaterjal]. Available: https://www.documentation.progress.com. [Kasutatud 03 02 2020].

[11] K. S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Boston: Addison-Wesley Professional, 2012.

[12] Politsei- ja Piirivalveamet, „Politsei- ja Piirivalveameti koduleheküIg," 2020. [Võrgumaterjal]. Available: https://www.politsei.ee/et/rahapesu-andmebueroo. [Kasutatud 11 04 2020].

Lisa 1 – Rahapesu andmebüroo juhend rakenduse arendamiseks

I TEATE ESITAMISE ÜLDSÄTTED

1. Rahapesu ja terrorismi rahastamise tõkestamise seaduse (edaspidi RahaPTS) § 49 kohustab kohustatud isikuid rahapesu andmebüroole esitama teate tegevuse või asjaolude kohta, mille tunnused osutavad kuritegelikust tegevusest saadud tulu kasutamisele, terrorismi rahastamisele või sellega seotud kuritegude toimepanemisele või sellise tegevuse katsele või mille puhul on kahtlus või teadmine, et tegemist on rahapesu või terrorismi rahastamisega või sellega seotud kuritegude toimepanemisega.

2. Rahvusvahelise sanktsiooni seaduse (edaspidi RSanS) § 12 lõike 2 (erikohustustega isik § 14 lõige 2) kohaselt annavad füüsiline ja/või juriidiline isik, kellel on kahtlus või kes teab, et temaga ärisuhtes olev või tehingut või toimingut tegev isik, samuti ärisuhte loomist või tehingu või toimingu tegemist kavandav isik on rahvusvahelise finantssanktsiooni subjekt, viivitamatult teada rahvusvahelise finantssanktsiooni subjekti tuvastamisest, sellekohasest kahtlusest ja võetud meetmest rahapesu andmebüroole. Kui otsekohalduv EL määrus sätestab finantssanktsiooni, mis ei ole seotud isikutega vaid territooriumi, üksuse või kaubaga, tuleb ka selle rakendamisest või rakendamise vajaduse kahtlusest ning võetud meetmetest viivitamata teavitada rahapesu andmebürood.

3. Rahapesu või terrorismi rahastamise kahtluse korral, ebaharilike tehingute identifitseerimisel ja kui on tuvastatud rahvusvahelise finantssanktsiooni subjekt või on kahtlus rahvusvahelise finantssanktsiooni subjekti tuvastamise kohta, juhinduvad kohustatud subjektid rahapesu andmebüroo koostatud juhendist rahapesu ja terrorismi rahastamise kahtlusega tehingute tunnuste ja rahvusvahelise finantssanktsiooni subjekti kohta. Juhend avalikustatakse rahapesu andmebüroo kodulehel.

4. Teade edastatakse rahapesu andmebüroole digitaalselt, kasutades vormi rahapesu andmebüroo kodulehel (link „Saada teade“) või rahapesu andmebürooga kokkulepitud vorminguga (XML-vorming) vormis. Rahapesu andmebürooga kokkulepitud vormingu

edastamine toimub läbi X-tee süsteemi. Juhul, kui teadet ei ole võimalik eelnevalt kirjeldatud viisidel edastada on erandjuhtudel võimalik rahapesu andmebüroo loal edastada teade e-postiga.

5. E-posti teel esitatud teade peab sisaldama vähemalt järgmisi andmeid:

- teate koostaja ja teataja isiku- ja kontaktandmed;
- teate liik ja number;
- teate esitamise põhiajend, olemasolu korral lisaajend(id);
- kõigi tehingus osalejate identifitseerimisandmed, rollid tehingus ja asukohaandmed;
- andmed tehingu objekti kohta;
- tehingu kirjeldus;
- vajadusel lisatakse asjakohaste dokumentide koopiad.

E-posti teel edastatavad teated saadetakse rahapesu andmebüroo üldisele e-posti aadressile rahapesu@politsei.ee.

6. Kui teade esitatakse rahapesu andmebüroo veebilehe kaudu, identifitseerib teate koostaja oma isiku ID-kaardi või mobiil-ID-ga. Uue teate koostamiseks vajutada nuppu „Lisa uus teade“. Teataja ja teate koostaja andmed, mis on sisestatud arvutisse eelnenud seansil on avanenud vormil säilinud. Kui teate koostamine või saatmine on eelnevalt pooleli jäänud, kuvatakse poolikud teated tabelina, kus on teate number ja sisestamise kuupäev. Tabelis olevate teadete sisestamist saab jätkata (vajutades teate numbrile) või mittevajalik kustutada (vajutades ristiga prügikasti sümbolile).

7. Teate koostamiseks peab täitma kõik väljad, mis on täiendatud tärn sümboliga (lisatud on * sümbol). Ilma tärnita väljad tuleb täita informatsiooni olemasolul. Teate infoblokkide vahel ei ole võimalik liikuda, kui tärniga märgitud väljad ei ole täidetud. Nupp „Edasi“ muutub aktiivseks pärast kõikide tärniga väljade täitmist. Mitteaktiivset nuppu „Edasi“ vajutades kuvab süsteem kohustuslikud täitmata väljad punaselt.

8. Teate koostamise katkestamise korral on poolik teade võimalik salvestada vajutades nuppu „Salvesta“, hiljem on pooliku teate koostamist võimalik jätkata süsteemi sisenemisel ning soovitud teate valimisel.

9. Erinevate infoblokkide vahel on võimalik liikuda kasutades lehekülje all paremal nurgas asuvaid nuppe „Edasi“, „Tagasi“ või teate üleval osas toodud infoblokkide pealkirjadele vajutades

II „TEATE KOOSTAJA“ INFOBLOKK

10. Süsteem kuvab automaatselt teate koostaja ees-, perekonnanime ja isikukoodi. „Sidevahendid“ alajaotuses välja toodud väljadele „Telefon“, „E-post“ ja „Koduleht“ tuleb lisada teate koostaja ja teataja sidevahendite andmed. Kohustuslik on esitada vähemalt üks telefoninumber ja üks e-maili aadress. Vajutades nuppu „Lisa“ on võimalik lisada täiendavaid sidevahendeid, valides sidevahendi rippmenüüst ning sisestades kontaktandmed väljale „Kontakt“. Täiendavate andmete säilitamiseks tuleb vajutada „sidevahendid tüüp“ ja „Kontakt“ ridade järel asuvat nuppu „Lisa“. Täiendavaid sidevahendeid on võimalik lisada korduvalt, vajutades igakordselt nuppu „Lisa“.

11. „Teataja“ alajaotuses tuleb rippmenüüst valida isiku tüüp - juriidiline või füüsiline isik, mitteresident või resident (juriidilise isiku puhul ka mitteresidendist osanikuga isik) või e-resident (edaspidi isiku tüüp).

12.Valitud isiku tüübi kohta tuleb sisestada kontaktandmed vastavalt tekkinud andmeväljadele (isiku nimi/nimed, registrikood/isikukood, mitteresidendi puhul täiendavalt tegevuskoht, päritoluriik, maksuresidentsus (riik) või mitteresidendist osanikuga residendi puhul omanikfirma andmed).

13.Seaduse kohustatud subjekt tuleb valida rippmenüüst, mis järel aktiveerub veebipõhisel vormil põhitegevusalade valik ning viimasena aktiveerub teatega seotud tegevusala.

14. Aadressi lisamiseks tuleb „Aadressid“ alajaotuses rippmenüüst valida riik ning „Sisesta aadress“ reale sisestada aadress (aadress tuleb lisada võimalikult täpselt). Eesti aadressi lisamisel pakub Maa-ameti infosüsteem variante, millest tuleb valida õige. Täiendavaid aadresse on võimalik lisada vajutades „Aadressid“ taga olevat nuppu „Lisa“.

15. Kui teataja on tehingu üheks pooleks tehakse vormil selle kohta märge, linnutades „Teate esitaja on tehingu üks osapooltest“ väli.

16. „Teate koostaja“ infoblokist edasi liikumiseks vajutada nuppu „Salvesta“ ja salvestamise õnnestumisel „Edasi“.

III „TEADE“ INFOBLOKK

17. „Teade“ infoblokis valitakse esmalt teate liik („Ebaharilik tehing (UTR)“, „Ebatavaline tegevus (UAR)“, „Rahapesu kahtlane tehing (STR)“, „RSanS-i subjekt (ISR)“, „Summa üle piirsumma sularahas (CTR)“ või „Terrorismi rahastamise kahtlus (TFR)“). Võimalik on sisestada teatele teataja sisemises asjaajamise korras antud number, lisades selle „Teate number“ väljale.

18. Teate liigi valimisega muutub aktiivseks „Põhiajend“, kus rippmenüüst tuleb valida vastav kahtluse tunnus või teate saatmise ajend. Lisaajendi olemasolu korral lisatakse see „Lisaajendid“ alajaotuses nupule „Lisa“ vajutamisega. Lisaajendi valik tuleb teha rippmenüüst, mitme lisaajendi olemasolu korral vajutada nuppu „Lisa“ korduvalt.

19. Vajadusel lisada teatele märge „Kiire“, vastava välja linnutamisega. Märke tegemisel valida tekkinud rippmenüüst kiireloomulisuse põhjus. Vajadusel sisestada põhjuse lisaselgitus, see nimetatud väljale sisestades. Igale RSanS-i teatele lisatakse märge „Kiire“ ja teate kiireloomulisuse põhjenduseks valitakse rippmenüüst „ISR kahtlus“.

20. „Teade“ infobloki alumises osas kuvatakse teataja viimase saadetud teate number koos kuupäevaga, mis on informatiivse sisuga.

21. „Teade“ infoblokist edasi liikumiseks vajutada nuppu „Salvesta“ ja kui salvestamine õnnestus, siis „Edasi“.

IV „OSAPOOLED“ INFOBLOKK

22. „Osapooled“ infoblokis vajutada nuppu „Lisa osapool“. Juhul, kui teataja on teatega seotud on võimalik isik rippmenüüst „Vali seotud isikute hulgast“ valida. Juhul, kui teatajal on tehingus esindaja, tuleb vastav väli linnutada. Kui esindaja on teatega seotud on võimalik ta lisada rippmenüüst „Vali seotud isikute hulgast“, muul juhul tuleb esindaja isik lisada, vajutades nuppu „Lisa uus isik“.

23. Juhul, kui osapoolel on täiendavad seotud isikud vajutada nuppu „Lisa seotud isik“, rippmenüüst valida isiku roll ning lisada isik, kas teatega seotud isikute hulgast või vajutades nuppu „Lisa uus isik“.

24. Vajutades nuppu „Lisa uus isik“ tuleb esmalt valida isiku tüüp, seejärel avanevad seda isiku tüüpi iseloomustavad väljad. Isiku tüübi valimisel tekkinud read täidetakse võimalikult täpselt. Mitteresidentidest või e-residentidest isikute korral tuleb isiku kohta täiendavalt andmeid sisestada.

25. Isiku infoblokis olevas „Pangakontod“ alajaotuses tuleb konto olemasolul lisada konto number, pank, panga asukoht, kusjuures välispanga puhul tuleb alustada asukohariigist. Eesti IBAN numbriga pangakontode puhul kontrollib programm pangakoodi ja kontonumbri pikkust. Arvelduskonto väljavõtte lisamiseks (vajutatakse vastavas kohas nuppu „Lisa dokument“), täidetakse metaandmete väljad: kuupäev, väljavõtte ajavahemik ning sisu. Seejärel vajutatakse nuppu „Lisa fail“ ja lohistatakse vastav fail oma kohale. Kui kõik väljad on täidetud ja fail lisatud vajutada nuppu „Salvesta“.

26. Isiku blokis on võimalik lisada füüsilisele isikule ka isikut tõendava dokumendi andmed ja juriidilisele isikule temaga seotud dokumentide andmed, valides rippmenüüs dokumendi liigi ja täites metaandmete väljad. Dokumendi koopia faili saab lisada vajutades „Lisa fail“ ja lohistades vastava faili oma kohale. Dokumendi koopia fail tuleb lisada, kui vajalik teave ei ole kättesaadav Äri- ja Rahvastikuregistrist. Esitatud dokumentide kommentaarid saab lisada väljale „Sisu“.

27. Isiku andmete kohta täiendavate selgituste andmiseks tuleb need sisestada väljale „Märkused“. Edasi liikumiseks vajutada nuppu „Salvesta“. Selleks, et salvestatud isik fikseerida teates osapoolena, vajutada nuppu „Lisa“.

28. Pärast osapoole salvestamist on võimalik isiku kohta esitatud andmeid muuta, vajutades nuppu „Muuda“ või isik eemaldada, vajutades nuppu „Eemalda“.

29. Kui teatesse soovitakse lisada veel osapooli on iga uue isiku korral seda võimalik teha valikuga „Lisa osapool“.

30. Pärast kõikide tehingus osalejate ja nende esindajate lisamist vajutada nuppu „Edasi“.

V „TEHING“ INFOBLOKK

31. „Tehing“ infobloki sisu sõltub varasemalt märgitud teate liigist ning ajenditest. Ette kuvatav vorm tuleb täita võimalikult täpselt.

32. Tehingu kohta informatsiooni lisamiseks vajutada nuppu „Lisa tehing“ ning võimalikult täpselt täita tekkinud read ja rippmenüüst valida sobivad variandid. Kui tehinguid on rohkem kui üks, lisatakse iga tehing eraldi, vajutades nuppu „Lisa tehing“. Kui tehing on teostatud sularahas linnutatakse „Sularaha“ väli. Seejärel valida avanenud väljadel „Riigist“ ja „Riiki“ rippmenüüst sobivad variandid. Kui tehingus kasutatakse pangakontosid, peavad need olema eelnevalt sisestatud vastavate tehingu osapoolte juurde. Tehingute kogusumma märgitakse „Teatega seotud kogusumma“ alajaotusesse iga valuuta kohta eraldi.

33. Juhul, kui teate esitaja küsis või teab vara päritolu kohta linnutatakse „Kas küsiti vara päritolu?“ väli, lisatakse vara päritolu kirjeldus ning olemasolu korral dokumendid, vajutades nuppu „Lisa dokument“ ning täites metaandmed ja lohistades faili kohale.

34. Juhul, kui tehingu järgselt on tulnud tagastusnõue tuleb linnutada „Tuli tagastusnõue?“ väli. Seejärel lisada tagastusnõude SWIFT kirje koopia vajutades nuppu „Lisa dokument“.

VI „DOKUMENDID“ INFOBLOKK

35. Teatele dokumentide lisamine toimub „Dokumendid“ infoblokis, vajutades nuppu „Lisa dokument“. Dokumentide lisamiseks tuleb rippmenüüst valida „Dokumendi liik“. Dokumendi liigi valimise järgselt täidetakse nõutud metaandmete väljad. Faile on võimalik lisada vaid digitaalses vormis, vajutades nuppu „Lisa fail“ ja lohistades fail(id) selleks märgitud alale „Faile saab lohistada siia“. Samuti on võimalik dokument siduda osapoolega, vajutades rippmenüüle „Lisa osapool“. Juhul, kui teate koostaja soovib edastada dokumendi sisu kohta täiendavaid selgitusi tuleb see sisestada „Sisu“ reale kirjeldusena.

36. Krediidiasutused lisavad teatele kontode avamise dokumendid ja konto väljavõtted ajavahemiku kohta, millal kahtlased tehingud aset leidsid või rahvusvahelise finantssanktsiooni subjekt oli krediidiasutuse kliendiks. Väljavõte peab kajastama konto seisu vahetult enne teate koostamist.

37. Teatele peab lisatud olema vähemalt järgmiste dokumentide koopiad, kui vastavad andmed ei ole Äri- ja Rahvastikuregistrist kätte saadavad:

1) isiku juurde- identifitseeritud mitteresidendist füüsilise isiku isikut tõendava dokumendi isikuandmete ja fotoga lehekülg;

2) isiku juurde- identifitseeritud mitteresidendi juriidilise isiku registreerimistunnistus või sellega võrdväärne dokument;

3) tehingus osaleva volitatud isiku esindusõigust tõendav dokument, ettenähtud vormis;

4) tehingu aluseks olev kirjalik lepe või korraldus ja nende lisad;

5) muud tehingut iseloomustavad dokumendid.

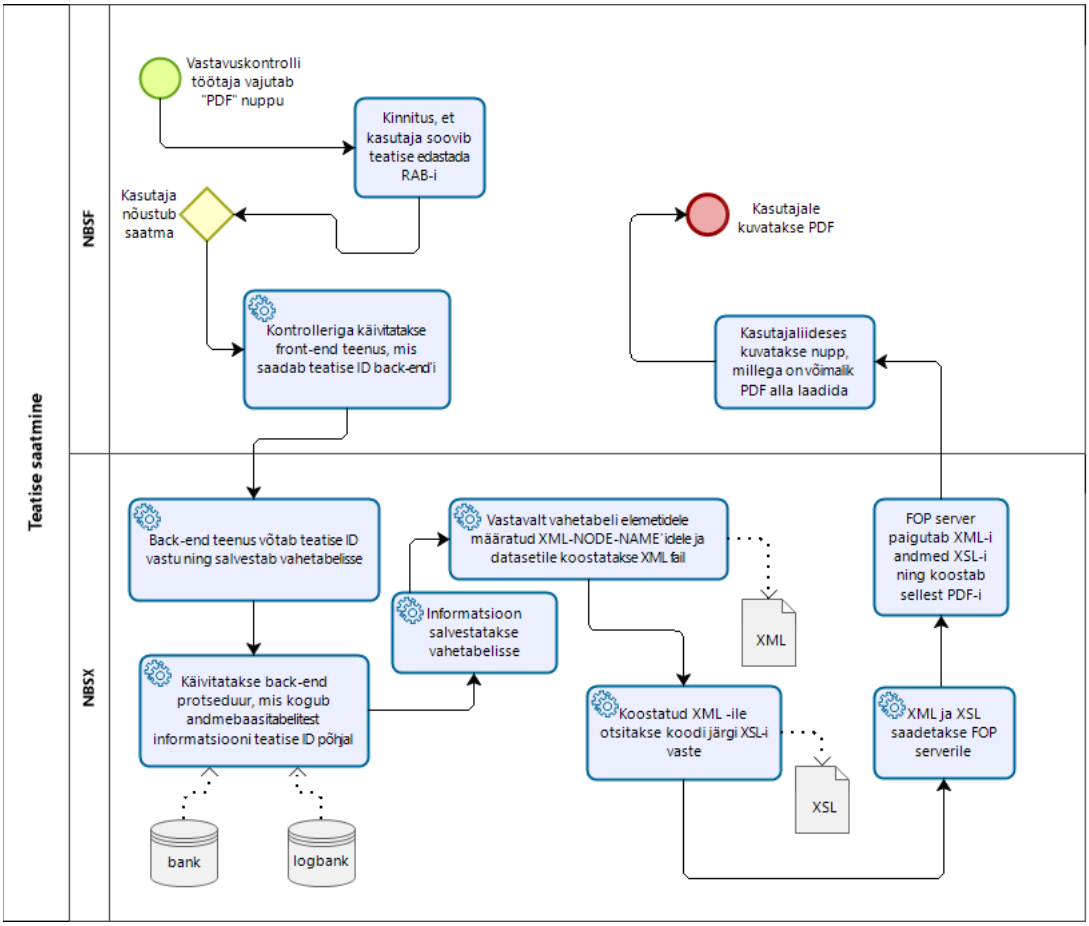
VI „ESITAMINE“ INFOBLOKK

38. „Esitamine“ infoblokis on enne dokumendi saatmist võimalik alla laadida PDF-vormingus eelvaade, vajutades nuppu „Lae alla eelvaade“. Teate muutmiseks on võimalik vajutada nuppu „Tagasi“. Teate esitamiseks vajutada nuppu „Esita teade“, seejärel teade krüpteeritakse ning saadetakse rahapesu andmebüroole ja sisestatakse rahapesu andmebüroo infosüsteemi RABIS edasiseks töötlemiseks. Teate esitamisega laeb süsteem automaatselt alla PDF-vormingus eelvaate teate koostaja arvutisse. Eelvaatest loobumiseks tühjendada väli „Esitamisel lae alla teate eelvaade“.

39. Kasutades veebipõhise vormiga sarnast rahapesu andmebürooga kokkulepitud XML-vormingus vormi, mis on ühendatud ettevõttes kasutatava infosüsteemiga, tagab kohustatud isik kohustuslike väljade täitmise kontrolli oma infosüsteemi vahenditega.

40. Tekkinud küsimused saadetakse rahapesu andmebüroo üldisele e-posti aadressile

Lisa 2 – Teatise eelvaate kuvamise protsess



Lisa 3 – Teatise eelvaade PDF

Teatis Rahapesu Andmebüroole

SEB

Teataja:	Teate koostaja:
Nimi: SEB Pank **Registrikood:** 10004252 **Seaduse kohustatud subjekt:** Krediidiasutus **Põhitegevusala:** Panga tegevus **Teatega seotud tegevusala:** Pangandus	**Eesnimi:** [redacted] **Perekonnanimi:** [redacted] **Isikukood:** [redacted] **Sidevahendid:** e-post: [redacted].test@seb.ee

Teade:

Teate saatmise kuupäev:
Teate number: 36
Teateliik: UTR
Põhiajend: 2.2. Ebaharilik tehing kontol
Lisaajend:
1.2. Isik käitub ebaharilikult
Kiire: Jah
Kiire selgitus: MUU
Kiire kommentaar:

Osapooled:

Nimi:	Tüüp:	Isikukood / Registrikood:
MARI-LIIS MÄNNIK	Füüsiline isik (resident)	47101010033
[redacted]	Füüsiline isik (resident)	[redacted]

Pangakontod:

Kontonumber:	Pank:	Panga asukoht:
[redacted]	SEB Pank	Eesti
[redacted]	SEB Pank	Eesti
EE301010011661681229	SEB Pank	Eesti
EE311010011644615225	SEB Pank	Eesti
[redacted]	SEB Pank	Eesti
[redacted]	SEB Pank	Eesti

Pangakontode seosed / rollid:

Nimi:	KontoNumber:	Roll:
[redacted]	[redacted]	OMANIK
[redacted]	[redacted]	OMANIK
MARI-LIIS MÄNNIK	EE301010011661681229	OMANIK
MARI-LIIS MÄNNIK	EE311010011644615225	OMANIK
[redacted]	[redacted]	OMANIK
[redacted]	[redacted]	OMANIK

Tehingu osapooled:

Nimi:	Esindaja:	Volitaja:
[redacted]		
Nimi:	**Esindaja:**	**Volitaja:**
MARI-LIIS MÄNNIK	[redacted]	

Seotud isikud:

Tehingud

Kirjeldus: Osapooled ei ole varasemalt omavahel tehinguid sooritanud. Tehingu selgitus oli ebapädev ning Mari-Liis Männiku kohta on käesoleva kuu jooksul sooritatud 3 rahapesukahtlase tehingu teatist.

Tehingu tüüp: SUMMA

Tehingu kuupäev:	Summa:	Valuuta:	Kellelt:	Konto:	Kellele:	Konto:
2020-05-08	1 225 000.00	EUR	[redacted]	[redacted]	MARI-LIIS MÄNNIK	EE311010011644615225

Valuuta	Kogusumma
EUR	1 225 000.00

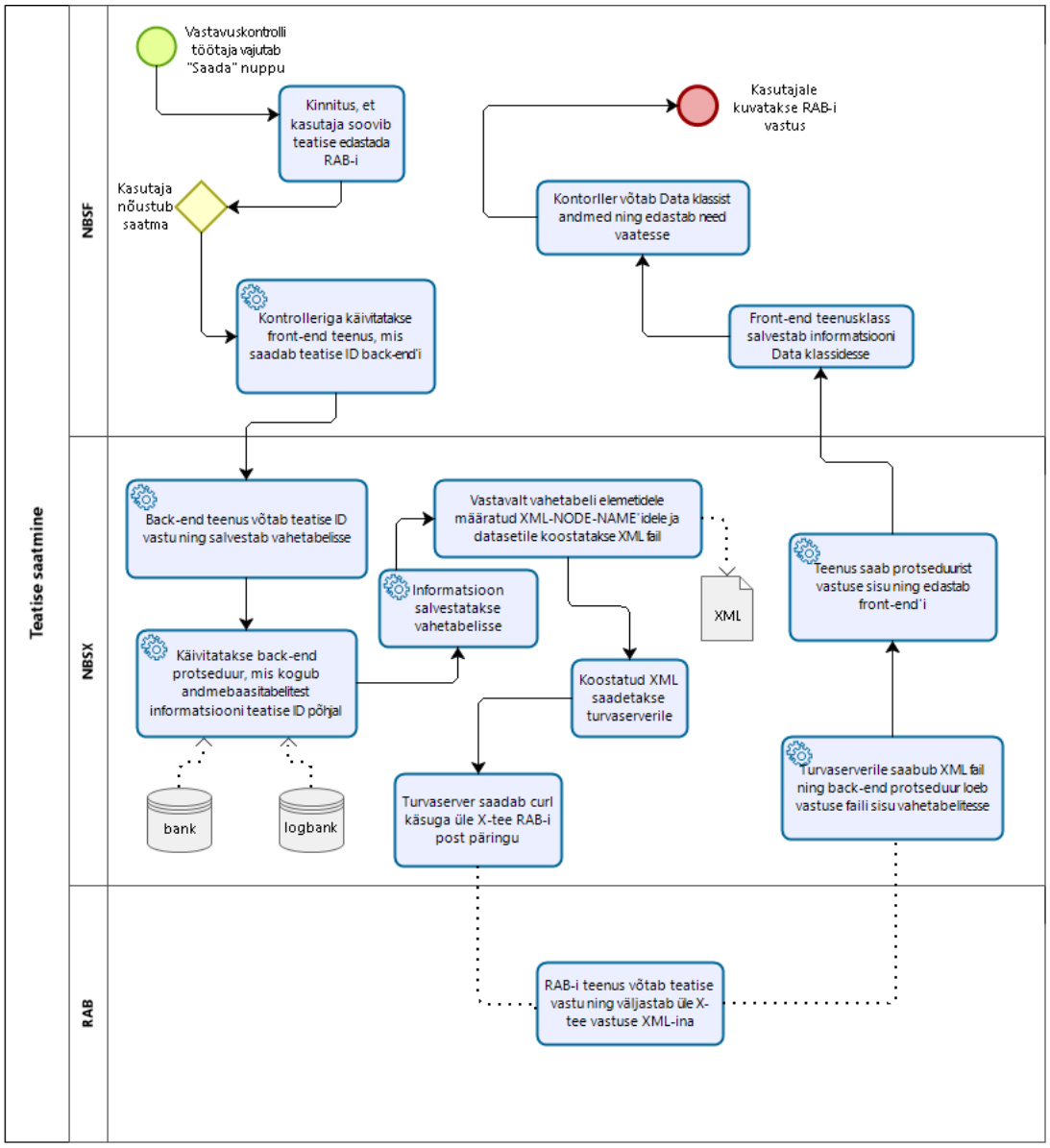
Kas küsiti vara päritolu?	Tuli tagastusnõue?
Ei	Ei

Dokumendid:

AS SEB Pank
Tornimäe 2, 15010 TALLINN
tel 372 66 55 100
e-post: info@seb.ee
212 707

Rahapesu andmebüroo
Tööstuse 52, 10416, TALLINN
tel. 372 61 23 840
e-post: rahapesu@politsei.ee

Lisa 4 – Teatise saatmise protsess



Lisa 5 – Arendusjuht Enel Pitki tagasiside projektile

Tulenevalt AML5 direktiivist (*Anti-Money Laundering Directive*) ning rahapesu ja terrorismi rahastamise tõkestamise seadusest on pankadele seatud erinevaid nõudeid klientide, tehingute, tegevuste jms kontrollide osas. Üheks kohustuseks pankadele on viivitamatult teavitada kahtlastest tehingutest rahapesu andmebürood. RAB analüüsib ja kontrollib kohustatud subjektidelt ning teistelt isikutelt laekunud teavet rahapesu või terrorismi rahastamise kahtluse kohta ja võtab vajaduse korral tarvitusele abinõud. Kuna raporteeritavate tehingute mahud kasvavad ja tähtis on kiire infovahetus, siis oli vajalik teha süsteemidele täiendavaid arendusi.

Süsteemide arendamisel ning uue rakenduse kasutuselevõtul oli eesmärgiks tegevuste maksimaalne automatiseerimine, et vähendada kasutajate käsitööd andmete sisestamisel. Vana rakenduse peamiseks puuduseks oli lühike sessiooniaeg ja kui kasutaja pidi kõik andmeväljad sisestama käsitsi, siis väga mahukate päringute koostamise korral võis sessioon lõppeda enne kui teade sisestatud sai. Kuna enamus infost (info andmesubjekti, tehingute kohta jms) on olemas pangasüsteemides, siis tuli leida lahendus maksimaalseks automatiseerimiseks, et kasutada andmebaasides olevat infot ja infovahetuse edastuseks XML-formaadis üle X-tee.

Töö täitmiseks avasime projekti 2019. aasta kevadel, milles osalesid neli IT-nooremarendajat (Marie Elise Soomre, Triin Tammaru, Lilian Väli, Carl-Robert Reidolf). Jagasime projekti kahte faasi. Esimese etapi skoop oli luua peamine funktsionaalsus – vastav menüüpunkt pangas kasutatavas programmis, raporteeritavate väljad, teate koostamine XML-formaadis ning ühesuunaline infovahetus üle X-tee. Projekti esimese faasi lõppaeg oli esialgu planeeritud mai lõpuks, kuid tulenevalt tööde keerukusest ning terviklahenduse testimisele kuluvale ajale, lükkus see mõned kuud edasi. Esimese faasi nihkumiste tõttu kandusid mõned nõuded ja funktsionaalsused teise faasi. Projekti teise faasi planeeriti seetõttu esmalt esimeses faasis tegemata jäänud nõuded ning seejärel automaatne täitmine (näiteks: kliendiandmed, kontode info, tehingutega seotud info), kasutajamugavuse parendused, statistikaraportid, kahesuunaline infovahetus üle X-tee,

infovahetuse toimimise testimine kolmanda osapoolega, koostatud teadete arhiveerimine ning muud väiksemad tööd.

Projekti elluviimiseks rakendasime agiilse arenduse meetodit, milles tööd olid jagatud väikesteks etappideks. Meeskonnaga toimusid meil igahommikused *stand-up* koosolekud, milles IT-arendajad andsid ülevaate tegevustest, mida nad eelmise päeva jooksul tegid, mis olid takistused, milles nad vajavad juhendajate abi ning mis on planeeritud tegevused eesolevaks päevaks. Arvestades asjaolu, et süsteemid olid IT-arendajate jaoks uued, mille iseärasustega nad ei olnud kursis ning samuti puudus neil veel teadmine, kelle poole pöörduda, tegid nad endale süsteemi väga kiiresti selgeks. IT-tehnilistes küsimustes abistas neid mentor. Olgugi et nad said toetuda mentori abile, võin kinnitada, et kogu nende tiim oli väga iseseisev, julge pealehakkamisega, lahendustele orienteeritud ning uute ideede loojad. IT-arendajad ei küsinud lahendusi vanem-arendajatelt, vaid püüdsid neid ise leida, kasutades kas koolis omandatut või ka näiteks infot internetist otsides, läbi mille nad lõid lisaväärtust pangale, tuues uusi teadmisi ka vanadele olijatele. Kogu meeskond töötas väga struktureeritult ning igal meeskonnaliikmel oli oma valdkond ning ülesanded, mille eest ta võttis vastutuse.

Kokkuvõtteks võib öelda, et projekt on olnud edukas ning saavutatud eesmärgid on täidetud. Tänaseks on süsteem edukalt kasutusele võetud, kuid aeg-ajalt esineb seal kitsaskohti, mida arendajad jooksvalt vajadusel parendavad. Takistused ei ole alati meie süsteemidest tulenevad, vaid sõltuvad ka kolmandast osapoolest. Samuti annavad kasutajad teada, kui nad soovivad täiendavaid muudatusi, mida arendajad võimalusel ellu viivad. IT-arendajad on pakkunud meie organisatsioonile oma uute teadmistega lisaväärtust ning andnud kindluse rakendada tudengeid ka edasiste projektide elluviimisel.

Lisa 6 – Eneseanalüüs

Projekti alustasin koos meeskonnaga äripoole nõuete ülevaatusest. Sellest lähtudes sai jaotada edasise töö kaheksa sprindi vahel ära. Esimesed 10 nädalat oli ühe sprindi pikkuseks kaks nädalat ehk 10 tööpäeva ning järgnevad neli nädalat oli pikkuseks 4 tööpäeva, kuna käisin paralleelselt ülikoolis.

Esimese sprindi tegelesin juba eelnevalt olemas olnud arenduse parandamisega. Tuli kindlaks teha, et ei teki rakenduse kasutamise käigus *NullPointerException*'eid ning valideerida, et kasutajate sisestatud informatsioon oleks korrektne. Peale vigade parandamist analüüsisin sügavuti NBSF rakendust, et leida osapoolte automaatika loomiseks parim lahendus. Sprindi lõpus alustasime Lilianiga ka meie rakendusse osapoolte automaatse otsingu arendamist. Mina tegelesin *back-end* mooduli loomisega, et saada andmebaasidest kätte vajalik informatsioon kliendi kohta, mida *front-end*'ile saata. Esimese sprindi lõpuks avastasime analüüsi tulemusena, et suures mahus saab kasutada juba *front-end*'is olemasolevat lahendust. Kolmas peamine ülesanne oli seada üles võimalus kirjutada *back-end*'is Progress OpenEdge keskkonnas ABL ühikteste. Selleks tuli uurida ABL Unit testraamistiku tööpõhimõtteid. Seega said esimese sprindiga kirjutatud esimesed ühiktestid *back-end*'is ning vastavalt info vajadusele tuli muuta *back-end* moodulit.

Teise sprindi alguses toimus esimene retrospektiiv, kus leppisime tiimisiseselt kokku, kuidas edaspidi tööd efektiivsemalt jaotada. Sprindi põhieesmärgiks oli kõikidele osapooltele, tehingutele ja erinevatele dokumentidele luua muutmise ning kustutamise loogika. Selleks tuli UseCase klassi luua uued meetodid, mis nimetatud funktsionaalsuseid sooritavad. Koos Carl-Roberti ja Lilianiga alustasime *back-end*'is rippmenüüde väärtustamiseks vajaliku mooduli täiendamist ning samuti teatistele staatuste määramise loogika arendamist. Andmebaasi tuli selleks lisada neli uut välja ning *front-end*'is jaotada tabel staatuste põhjal kolmeks.

Kolmanda sprindi peamiseks eesmärgiks oli parandada kasutusjuht, kus lisatud osapoolel ei ole identifikaatorit. Selle jaoks sai algul arendatud moodul *back-end*'i, kuid üsna kiiresti sai loogika ümber tõstetud *front-end*'i, et ei tekiks vigu objektide muutmisel. Identifikaatori loomiseks pidin analüüsima RAB-i poolseid nõudeid. Veel tuli jooksvalt parandada äripoole testimisega fikseeritud erindeid ning muuta tabelites kuvatavat informatsiooni. Lisaks kirjutasin meeskonnaprojektiga valminud moodulitele ühikteste ning alustasin teatisele kasutajaõiguste lisamise loogika arendamisega.

Neljanda sprindi peamine ülesanne oli korrastada olemasolevat arendust, luua osapooltele järjekorra loogika ning panna paika edasine plaan. Järjekorra loogika arendamiseks tuli osapoole objektidele lisada järjekorranumber. Lisaks tuli silmas pidada osapoole kustutamisel järjekorranumbriga tekkivaid võimalike erindeid. Andmebaasi sai lisatud ka uus väli saadetud teatise kommentaari jaoks ning alustasin selle salvestamise mooduli arendamist. Mina tegelesin kommentaari *back-end*'is salvestamise ning *front-end*'ile tagasi saatmise moodulite arendamisega. Lisaks arendasin lisatud failidele esialgse kustutamise loogika ning arendasin erinevate osapoolte, dokumentide jne lisamise funktsionaalsustele mitmeid lisakontrolle.

Viienda sprindi jooksul lõpetasin saadetud teatise kommentaari *back-end* moodulite ning teatise raporti loomise arendamisega. Tegelesin *back-end* moodulitega, mis käivitavad andmete voogedastuse .csv faili. Kuna äripool soovis raportis näha ka tehingu kirjeldust, mis on andmebaasi salvestatud *clob* ehk suuremahulise väljana, siis tuli analüüsida erinevaid võimalusi selle voogedastuseks. Lisaks parandasime Lilianiga tehingu osapoolte kuvamisega seotud loogikat ning PDF-il summa kuvamise formaate.

Kuuenda sprindi peamiseks ülesandeks oli olemasoleva koodi refaktoreerimine, testide parandamine ning *front-end*'i disain ja UX'i parendamine. Seitsmenda sprindi jooksul testisime arendustiimiga kõikvõimalikke kasutusjuhte ning samuti testisime erinevaid juhtusid RAB-iga. Kaheksanda sprindi eesmärgiks oli testida kogu rakenduse funktsionaalsust koos äripoolega.

Kõige keerulisemaks osutus minu jaoks kogu töö planeerimine, sest selle maht osutus oodatust palju suuremaks ning varasemate projektide tegemisel ei pidanud me arvestama tellija soove. Arendusprotsessi käigus osutus raskeks ka osapoolte automaatika loomine. Kuigi sisuliselt oli raporti loomine keerulisem, siis sellel hetkel kui me Lilianiga

osapoolte automaatikat arendasime, ei olnud meil veel nii palju kogemust. Tunnen, et kõige arendavam oli minu jaoks IT-ettevõtte üldprotsessidest osasaamine tarkvaraarendajana töötades. Olin lõputöö projektiks hästi ettevalmistunud, sest olime eelnevalt meeskonnaprojekti käigus erinevate süsteemide ja tehnoloogiatega tutvunud.