TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Taavi Pärt 183366IAAM

KAARDIMAKSETE VASTUVÕTMISE PETTUSTE MONITOORINGUSÜSTEEMI ANALÜÜS

Magistritöö

Juhendaja: Nadežda Furs-Nižnikova
MBA

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: Taavi Pärt

18.05.2020

Annotatsioon

Magistritöö autor töötab finantsettevõttes töövahendite arenduse juhina. Käesoleva töö eesmärgiks on läbi viia äri- ja süsteemianalüüs, mille baasil on võimalik luua tarkvarasüsteem kaardimaksete vastuvõtmisel toimuvate pettusekahtlusega tehingute tuvastamiseks. Töö tellija ja lõppkasutaja on finantsettevõtte kaardimaksete vastuvõtmise osakond.

Magistritöö käigus tehakse äri- ja süsteemianalüüs ning kirjeldatakse lahenduse arhitektuur ja disain. Töö autor valib ja põhjendab töös kasutatavaid analüüsimeetodeid. Lisaks analüüsib ja põhjendab autor süsteemi arendamiseks kasutatavate tehnoloogiate ja tehnoloogiliste raamistike valikut.

Töö tulemusena on valminud infosüsteemi äri- ja süsteemianalüüs, lahenduse arhitektuuri kirjeldus ning rakenduse minimaalne töötav toode. 2020. aasta suvel hindavad finantsettevõtte juhatus ja huvitatud osapooled, kas alustada rakenduse täisversiooni arendamist. Juhul, kui lahenduse arendamisega soovitakse edasi liikuda, jõuab rakendus toodangukeskkonda hinnanguliselt 2020. aasta viimases kvartalis.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 87 leheküljel, 11 peatükki, 28 joonist, 10 tabelit.

Abstract

Analysis of Card Acquiring Fraud Monitoring System

The aim of the master's thesis is to conduct a business and system analysis, on the basis of which it is possible to create a software system with which is used to identify suspicious card acquiring transactions.

Objectives of the master's thesis are:

- analysis and choice of analysis methodologies;
- business analysis;
- system analysis;
- architecture of the solution;
- design of the solution;
- analysis and choice of development and deployment frameworks;
- creating minimum viable product.

Author compares and analyses possible alternatives to proposed solution under analysis, maps the stakeholders, conducts interviews with them and defines business description of the solution based on the interviews. Author defines 4 business processes in BPMN, same business processes will be described in more detail as Use Cases. Author creates following UML (Unified Modelling Language) diagrams to describe the proposed solution:

- Entity-Relationship diagram;
- Use Case diagram;
- Activity diagram;
- Component diagram;
- Sequence diagram;
- Deployment diagram.

As a result of the master's thesis minimum viable product has been created. Also it is possible to start development of full version of Card Acquiring Fraud Monitoring System

The thesis is in Estonian and contains 87 pages of text, 11 chapters, 28 figures, 10 tables.

Lühendite ja mõistete sõnastik

KMV	Ingl. k. *Card Acquiring*; kaardimaksete vastuvõtmine
	Ingl. k. *Business Process Management*; äriprotsesside juhtimine
BPMN	Ingl. k. *Business Process Modeling and Notation*; äriprotsesside modelleerimiskeel
ERD	Ingl. k. *Entity Relationship Diagram*; olemi-suhte diagramm, metoodika andmemudelite koostamiseks ja kirjelduse esitamiseks
Kaupmees	Kaardimaksete vastuvõtmise teenuse kasutaja
FURPS	Tarkvaranõuete klassifitseerimise mudel
Lõppkasutaja	Roll või ametikoht, kes hakkab arendatavat rakendust kasutama. Käesoleva töö kontekstis on see KMV-müügiinsener.
HTTPS	Ingl. k. *Hyper Text Transportation Protocol Secure*; turvaline protokoll autenditud ja krüpteeritud informatsiooni edastamiseks arvutivõrkudes
Klient	Kaardimaksete vastuvõtmise teenuse kasutaja
MTT	Ingl. k *Minimum Viable product*; minimaalne töötav toode
MTTC	Ingl. k *Mean Time to Change*; keskmine muudatuse teostamiseks kuluv aeg
SEPA	Ingl. k *Single Euro Payment Area*; ühtne euromaksete piirkond
SSO	Ingl. k. *Single sign-on;* ühe kontoga sisse logimine
RPO	Ingl. k. *Recovery Point Objective*; maksimaalne lubatav andmekadu
RTO	Ingl. k. *Recovery Time Objective*; *s*üsteemi taastamiseks kuluv ajaline eesmärk
Tellija	Isik või organisatsioon, kes tellib rakenduse analüüsi või arendamise. Käesoleva töö kontekstis on see kaardimaksete vastuvõtmise osakond.
UC	Ingl. k. *Use Case*; kasutusmall kirjeldab süsteemis aktori poolt tehtavaid tegevusi
UML	Ingl. k. *Unified Modeling Language*; üldotstarbeline noteeringukeel
VBA	Ingl. k. *Visual Basic for Applications*; Microsofti toodete skriptimiskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Magistritöö hõlmab kaadimaksete vastuvõtmise (edaspidi KMV) pettuste monitooringusüsteemi analüüsi teostamist ja minimaalse töötava toote loomist. Töö kirjutamise hetkel on kasutusel olemasolev monitooringusüsteem, mis ei täida kõiki lõppkasutaja ootuseid. Magistritöö eesmärk on läbi viia äri- ja süsteemianalüüs, et oleks võimalik luua minimaalne töötav toode ning seejärel teha otsus soovitud lahenduse täisversiooni arendamise osas.

Esimeses osas antakse ülevaade kaardipettuste valdkonnast ja sõnastatakse probleemipüstitus.

Teises osas seatakse töö eesmärk, sõnastatakse autori roll ning defineeritakse töö skoopi kuuluvad ja töö skoobist välja jäävad teemad.

Kolmandas osas valitakse arendusmetoodika ja kirjeldatakse ning analüüsitakse rakenduse äri- ja süsteemianalüüsil kasutatavate metoodikate valikuid.

Neljandas osas analüüsitakse erinevaid kaardipettuste tuvastamise ja ennetamise meetmete liike ning kirjeldatakse käesoleva töö jaoks tehtud valikuid. Samuti hinnatakse alternatiivsete lahenduste vastavust lõppkasutaja esitatud nõuetega.

Viiendas osas viiakse läbi ärianalüüs ja loetletakse huvitatud osapooled. Intervjuude põhjal luuakse rakenduse ärikirjeldus ja -ärireeglid ning koostatakse äriinfo mudel olemi-suhte diagrammina. Samuti kirjeldatakse ja visualiseeritakse peamised äriprotsessid.

Kuuendas osas teostatakse süsteemianalüüs – kirjeldatakse süsteemi kasutusmalle, mis visualiseeritakse kasutusmallide mudelina. Luuakse tegevusdiagramm süsteemi kõikide tegevuste osas. Lisaks põhjendatakse rakenduse arendamiseks ja juurutamiseks kasutatavate tehnoloogiate ja raamistike valikut.

Seitsmendas osas defineeritakse rakenduse arhitektuur. Süsteemid kuvatakse komponentdiagrammil ning andmete täpsem liikumine on välja toodud

monitooringulahenduse ja andmete laadimise järgnevusdiagrammidel. Rakenduse juurutamise mudel on visualiseeritud evitusdiagrammina.

Kaheksandas osas kirjeldatakse magistritöö tulemusel loodud minimaalse töötava toote funktsionaalsust. Samuti antakse ülevaade tagasisidet, mida minimaalse töötava toote kohta on lõppkasutajalt saadud. Täpsemalt saab minimaalse töötava toote kohta lugeda Lisast 1.

Üheksandas osas hinnatakse töö käigus saavutatud eesmärke. Autor toob välja edasise tegevuskava, annab soovitusi edasiarendusteks ning hindab rakenduse korduvkasutuse potentsiaali.

2 Probleemipüstitus

Käesolevas peatükis annab autor ülevaate kaardipettuste valdkonnast. Samuti sõnastatakse probleem, mille lahendamisega töö käigus tegeletakse.

2.1 Kaardipettuste valdkonna ülevaade

Kaardimaksete vastuvõtmine sisaldab endas kahte eraldi teenust. Esimene on kaardimaksete vastuvõtmine makseterminalides, teine on kaardimaksete vastuvõtmine läbi interneti. Viimast kasutavad näiteks e-kaupmehed ja hotellid üle Euroopa.

KMV-pettust ehk kaardimaksete vastuvõtmise pettust on mitut tüüpi. Näiteks võib kaupmees osta kurjategijatelt varastatud krediitkaardiandmeid ning proovida nendelt väikseid summasid võtta, lootes, et need ei ärata kahtlust. On ka petiseid, kes üritavad kätte saadud kaardiandmetega kaardid limiidini tühjaks kulutada. Lisaks leidub kaupmehi, kes loovad variettevõtte, aga selle varjus lasevad teistel kurjategijatel varastatud kaartidega tehinguid teha [1].

Suurima rahalise kahjuga KMV-pettus toimus aastatel 2003–2013 ja selle käigus peteti ettevõtetelt ja finantsasutuselt välja ligikaudu 200 miljonit USA dollarit [2]. Enamiku pettuse rahalisest kahjust kannab kaardi väljastaja (*Issuer*). Et mitte riskida leppetrahviga ja regulaatorite ettekirjutustega, peavad maksete vastuvõtmise teenust pakkuvad finantsettevõtted täitma kaardiorganisatsioonide seatud tingimusi ning regulatiivseid nõudeid.

Kaardipettuste liigid füüsilise kaardiga: [3]

- Pettused võltsitud kaardiga (*Skimming*) tehakse kaardi magnetriba kopeerimisega selleks, et kulutada raha väljaspool SEPA piirkonda – riigid, kus EMV standardeid (PIN koodiga autentimine, kiipi lugev ATM) pole veel rakendatud. Näiteks USA ja teatud Aasia piirkonnad.

- Pettused kaotatud ja varastatud kaartidega – kaotatud kaarte kasutatakse peamiselt volitamata kaarditehingute tegemiseks, vähem levinud on kaartide varastamine.

Kaardipettuste liigid Internetis: [3]

- Puhas pettus – kui kurjategijad omandavad kaarditehingute tegemiseks täiskomplekti andmeid, sealhulgas ligipääsu kaheastmelise autentimise infole. Kaupmeestel on sel juhul peaaegu võimatu tuvastada, et ostu sooritaja ei ole kaardi reaalne omanik.

- Identiteedivargus – pettur kasutab volitamata tehingu tegemiseks kaardiomaniku isikuandmeid. Seda pettust võib liigitada ka „kaotatud või varastatud kaardiga“ pettuseks, kuna kliendikaardi andmed varastatakse ja neid kasutatakse näiteks veebis kaupade ja teenuste ostmiseks edasimüügi eesmärgil. Seda tüüpi pettused kattuvad osaliselt puhta pettusega.

- „Sõbralik“ pettus või kliendipettus – maksja sooritab veebis kaupade või teenuste ostmiseks ehtsa tehingu. Peale seda taotleb ta kaardi väljastanud pangalt tagasimakset, väites, et teda on petetud. Seda tüüpi pettused on Euroopa Keskpanga väitel viimastel aastatel kasvanud.

2.2 Probleemipüstitus

Olulisemad finantsettevõtted on tänapäeval suures osas IT-ettevõtted. Klientidele pakutavad teenused ja tooted on digitaalsed ja inimeste ootus on, et tehingud toimuksid sekundite või minutite jooksul. Lisaks klienditoodetele on finantsettevõtetel ka sisemised ja toetavad protsessid nagu näiteks juriidiline tugi, raamatupidamine, maksete haldus ja muud tugiteenused.

Töötajatepoolne surve protsesside digiteerimiseks ja automatiseerimiseks on suur, kuid IT-arenduse ressurss on alati piiratud. Samas ei saa ka sisemisi protsesse tähelepanuta jätta. Sisemistest tööprotsessidest enamik sisaldavad suurel määral andmete hankimist ning nende baasil järelduste tegemist. McKinsey 2016. aastal läbi viidud analüüsi kohaselt on 60% andmete sisestamise ja haldamisega seotud protsessidest hõlpsalt automatiseeritavad [1]. Ettevõtjatele on see valik: kas suurem alginvesteering või suuremad tööjõukulud tulevikus.

Peamise küsimusena käsitletakse töös KMV-monitooringu lahenduse süsteemianalüüsi ja tehnilist lahendust. Pettuse monitooringu täpsemad reeglid ja nende efektiivsus on ärisaladus ja kuulub riskijuhtimise valdkonda, töös puudutatakse viimati mainitud teemasid vaid üldisel tasemel.

Põhiline töös käsitletav probleem seisneb selles, et hetkel finantsettevõtetes kasutusel olev pettuste monitooringusüsteem on valdavas osas manuaalne ning on loodud suurteks arendusprojektideks mittesobivas programmeerimiskeeles VBA (*Visual Basic for Applications*). Puudub lahendus, mis liidestuks otse andmeallikatega, sellest tulenevalt kasvab andmete sisestamisele kuluv aeg ning tõuseb käsitööst tulenevate vigade esinemise tõenäosus.

Olemasolev reeglite arvutamise mootor on aeglane – kiirem mootor võimaldaks pettusekahtlusega maksete tuvastamise lühema aja vältel.

Kasutusel olevas pettuste monitooringusüsteemis ei ole lõppkasutajal võimalik monitooringu reegleid muuta ega kombineerida, vaid selleks peab pöörduma arendaja poole. See aga muudab süsteemi kasutajal reeglitega katsetamise võimatuks, ilma et ta kulutaks arendaja aega. Seeläbi on raskendatud efektiivsete pettusetuvastamise reeglite väljatöötamine.

Töö käigus loodava äri- ja süsteemianalüüsi alusel valmiva rakenduse täisversioon asendab olemasolevat VBAl põhinevat poolmanuaalset lahendust. Uus lahendus peab liidestuma väliste süsteemidega, et automaatselt andmeid laadida. Lõppkasutaja peab saama monitooringu reegleid sisestada, muuta ja kombineerida, et tagada hõlpsam pettuse reeglite väljatöötamine. Pettuste tuvastamise mootor peab olema piisavalt kiire, et kuvada lõppkasutajale tulemusi mõne sekundi jooksul. Süsteem peab liidestuma maksevahenduse teenusepakkuja süsteemiga, et edastada sinna asjakohast infot. Lisaks peab lahenduses olema võimalus kliente teavitada (pettusekahtlusest) e-mailiga, mis sisaldab eeldefineeritud andmevälju.

Analüüsitava infosüsteemi täisversiooni pakutavaks väärtuseks võrreldes olemasolevaga on:

- Lõppkasutaja aja kokkuhoid andmete automaatse laadimise näol;

- Lõppkasutaja aja kokkuhoid tulenevalt teenusepakkuja süsteemi automaatsest uuendamisest;
- Lõppkasutaja aja kokkuhoid klientide poolautomaatse teavitamise näol;
- Kiirem pettuste tuvastamine tulenevalt kiiremast arvutusmootorist;
- Madalam käsitöö eksimustest tulenev risk – parem andmete kvaliteet;
- Kodeeritud reeglite asendamine kasutaja poolt muudetavatega – see võimaldab lõppkasutajal tulemuslikumalt efektiivseid monitooringu reegleid välja töötada.

3 Töö eesmärk

Käesolevas peatükis sõnastab autor magistritöö eesmärgi. Autor kirjeldab enda rolli töökohal ja magistritöös. Tuuakse välja magistritöö skoop ja tööst välja jäävad punktid.

3.1 Eesmärgi püstitus

Töö eesmärgiks on läbi viia äri- ja süsteemianalüüs loodavale KMV-pettuste monitooringu süsteemile, töötada välja loodava rakenduse arhitektuur ning analüüsida süsteemi arendamise ja juurutamise põhimõtete ning tehnoloogiate valikuid. Analüüsi tulemusena valmib monitooringulahenduse minimaalne töötav toode, mis täidab osa analüüsis kirjeldatud kasutusmallide eesmärkidest. Täpsemalt on võimalik minimaalse töötava toote kohta lugeda peatükist 9. Täiendavalt tuuakse välja ettepanekud süsteemi edasisteks täiendusteks.

3.2 Autori roll

Magistritöö autor töötab finantsettevõttes töövahendite arenduse juhina. Töövahendite arendus automatiseerib sisemisi tööprotsesse ning teeb väiksemaid arendusi, mis alati ei jõua suuremate arendustiimide tööplaani, kuid on sellegipoolest vajalikud. Meeskond prioritiseerib töid aja kuluefektiivsuse alusel, kuid võtab arvesse ka käsitööprotsessist tulenevat riski.

Töövahendite arenduse meeskonna eripäraks on see, et meeskonnaliikmed teevad üldjuhul kõik arenduse etapid iseseisvalt. See tähendab, et magistritöö kõik etapid ärianalüüsist kuni minimaalse töötava toote loomiseni on teinud töö autor.

3.3 Magistritöö skoop

Magistritöö skoopi kuuluvad äri- ja süsteemianalüüs ning rakenduse minimaalse töötava toote arendus. Lisaks kirjeldatakse töö käigus lahenduse arhitektuuri.

Magistritöö skoopi kuulub:

- äri- ja süsteemianalüüsi metoodikate analüüs ja valik;
- ärianalüüsi teostamine;

- peamiste kasutajalugude visualiseerimine ja kirjeldamine;
- süsteemianalüüs;
- süsteemiarhitektuuri kirjeldus;
- tehnoloogiate ja raamistike analüüs ja valik;
- geneerilise minimaalse töötava toote loomine.

Magistritöö skoopi ei kuulu:

- pettuse tuvastamise reeglite analüüs ja kirjeldus;
- arenduse mahuhinnangute andmine;
- rakenduse täisversiooni arendamine.

3.4 Olemasolev lahendus

Hetkel kasutatav lahendus baseerub VBAl ja MS Excelil. Spetsialist logib käsitsi sisse kaardiorganisatsioonide süsteemidesse ning kopeerib sealt vajalikud andmeread Exceli lehele – järgnevalt tuleb MSSQL baasist hankida kaupmehi puudutavad andmed ning sisestada need Excelisse. Seejärel kuvab süsteem iga kaupmehe kohta pettuseriski vastavalt sisestatud reeglitele.

Murekohaks on andmete sisestamisele kuluv aeg ning andmete käsitsi töötlemisel tekkiv suur operatsiooniriski realiseerumise oht. Olemasolev mootor on aeglane ning ebastabiilne, mis tähendab, et Excel ei pruugi ka pärast mitme-tunnist tööaega soovitud tulemust anda.

4 Metoodikate ülevaade

Selles peatükis antakse ülevaade analüüsi- ja arendusmetoodikatest ning põhjendatakse töö jaoks tehtud valikuid.

4.1 Kasutatav arendusmetoodika

Tarkvara arendamiseks on laias plaanis valida kahe erineva lähenemise vahel – lineaarne kosemudel või mõni inkrementaalne metoodika (näiteks Scrum või Kanban). Kosemudel arendati välja 1970. aastatel, see läbib lineaarselt erinevaid etappe, seejuures peab eelnev etapp olema valmis, enne kui uuega alustatakse. Algselt oli see kasutusel suurte militaarprojektide juures [4]. Kuna kosemeetodiga on planeerimisfaasis parem ülevaade projekti ajakavast ja kuludest, siis eelistatakse seda siiani avaliku sektori ja riigihangetega tellitavatel arendustel. Samuti on kosemudel mõistlik, kui tellija ja teostaja ei ole samast ettevõttest ja kõiki küsimusi ei ole võimalik igal hetkel üle täpsustada. See annab arendajale kindluse, et ta saab juba dokumenteeritud nõuete alusel arendada.

Kosemudeli etapid: [4]

1. Nõuete kaardistamine ja analüüs. Süsteemi teenused, piirangud ja eesmärgid selgitatakse välja koostöös süsteemi lõppkasutajatega. Seejärel luuakse kaardistatud infost süsteemi nõuete kogum.

2. Süsteemi ja tarkvara kavandamine. See etapp võimaldab määrata konkreetsemaid nõudeid kavandatavale infosüsteemile – selle käigus defineeritakse süsteemi arhitektuur ning süsteemi osade omavahelised suhted ja liidestused.

3. Arendamine ja testimine. Selle etapi käigus realiseeritakse kavandatud süsteem toimivaks tarkvaraks. Iga süsteemi komponent testitakse, et ta vastaks süsteemi nõuetele.

4. Süsteemi liidestamine ja integratsioonitestid. Süsteemi alamosad liidetakse ühtseks tervikuks. Viiakse läbi integratsioonitestid, et näha, kas süsteem tervikuna töötab nii nagu ette nähtud.

5. Haldus ja hooldus. Tavaliselt on see ajaliselt kõige pikem etapp. Süsteem on lansseeritud ning tegeletakse vigade parandamisega. Vajadusel arendatakse süsteemile uusi võimekusi.

Inkrementaalsete arendusmetoodikate puhul läbitakse tarkvaraarenduse elutsüklis need samad etapid, kuid seda ei tehta alati ranges järjestuses ning järgneva punkti juurest võidakse tagasi tulla eelneva juurde. See tähendab, et kogu süsteemi ei defineerita enne arenduse algust, vaid pannakse paika põhifunktsionaalsused ning lõppkasutaja saab võimalikult kiirelt kätte rakenduse esimese töötava versiooni – seejärel annab kasutaja tagasisidet, mis võiks olla paremini ning arendajad võtavad selle järgmiste versioonide aluseks [4].

Inkrementaalsete metoodikate eeliseks on hiliste muudatuste sujuvam läbiviimine. See sobib hästi selliste süsteemide arendamiseks, kus ei suudeta või taheta planeerimisprotsessi alguses kõiki süsteemi omadusi kirjeldada.

Sommerville jagab tarkvarasüsteemid kaheksaks klassiks. Loodav KMV-monitooringu süsteem on veebirakendus, millele lõppkasutaja pääseb ligi oma arvutist, see tähendab, et magistritöös arendatav süsteem liigitub interaktiivseks tegevuspõhiseks rakenduseks *(Interactive Transaction-based Application).* Nendeks on veebirakendused, millele lõppkasutajad pääsevad ligi enda seadmest. Selliste süsteemide arendamiseks sobivad Sommerville'i järgi kõige paremini iteratiivsed arendusmetoodikad [4].

Magistritöös arendatava lahenduse puhul kasutatakse Kanban-metoodikat, mis on inkrementaalne, kuid jätab teostajale suhteliselt vabad käed töö korraldamiseks. Kanban jagab tööd kolme gruppi: ootel, pooleli, valmis – pooleli olevate tööde hulk on piiratud, et arendaja ei tegeleks korraga liiga paljude teemadega [5]. Kuna tellija ja arendaja asuvad samas ettevõttes, siis ei ole vajadust kõiki nõudeid eelnevalt kaardistada, vaid süsteemi spetsifikatsioon saab areneda järk-järgult.

Kanbani on töövahendite arenduste tiimi töö korraldamisel eelistatud Scrumile seetõttu, et väikese meeskonna puhul ei ole võimalik täita kõiki Scrumis ettenähtud rolle ning iteratiivne tööprotsess ei sobi ettevõttesiseste töövahendite arendamiseks kuigi hästi. Muudatused töö skoobis toimuvad tihti loetud päevade jooksul ning see muudaks sprindi planeerimisel tehtud töö osaliselt kasutuks. Kanbaniga on pidevaid muutusi lihtsam

hallata. Kui Scrumi puhul toimub koodi reliisimine sprindi lõpus ehk kord nädala või kahe jooksul, siis Kanbani puhul võib reliise teha nii tihti, kui vajalik.

Sisemiste töövahendite arendamisel on tihti vajalik koodi muudatusi reliisida mitu korda päevas – see tähendab, et arukas on järgida pidevintegratsiooni (*Continuous Integration*) ja pidevvalmiduse (*Continuous Delivery*) põhimõtteid, mis sobituvad töö autori hinnangul Kanbaniga paremini kui Scrumiga. Eelmainitud põhimõtete kirjelduse ning ülevaate nende rakendamisest käesoleva projekti evitamisel leiab peatükist 7.3.3.

4.2 Ärianalüüsi metoodikate valik ja ülevaade

Selles peatükis tutvustatakse töös kasutatavaid ärianalüüsi metoodikaid ning põhjendatakse töö jaoks tehtud valikuid. Peamiselt keskendutakse siin nõuete kogumise ja kaardistamise ning äriprotsesside kirjeldamise kuvamise tehnikatele.

4.2.1 Ärinõuete kogumine ja kirjeldamine

Nõuded rakendusele tulenevad teenuste kirjeldusest, mida antud rakendus peaks pakkuma, ning piirangutest ja tingimustest, mida süsteem peab täitma. Nõuded peegeldavad lõppkasutaja vajadust süsteemi järele, mis aitaks kasutajal mingit konkreetset küsimust lahendada. Vajaduseks võib olla näiteks info otsimine, sisendi andmine süsteemile või süsteemi juhtimine [4].

Võimalikud nõuete kaardistamise tehnikad BABOK (*Business Analysis Body of Knowledge*) v3 kohaselt: [6]

- võrdlusuuringud ja turuanalüüs: infot saadakse läbi võrdluse mõne välise süsteemi, toote või teenuse vastu. Turuanalüüsi kasutatakse, et välja selgitada, mida kliendid tahavad ning mida konkurendid juba pakuvad;

- ajurünnak: kasutatakse huvitatud osapooltega ideede genereerimiseks ja prioriseerimiseks.

- ärireeglite analüüs: kasutatakse reeglite kaardistamiseks, mis reguleerivad otsuste langetamist organisatsioonis;

- ühismängud: kasutatakse probleemi paremaks mõistmiseks. Aitab genereerida loovaid lahendusi;

- kontseptsiooni modelleerimine: kasutatakse oluliste mõistete ja ideede tuvastamiseks ning nende omavaheliste suhete defineerimiseks;

- andmekaeve: kasutatakse asjakohase teabe ja mustrite tuvastamiseks;

- andmete modelleerimine: kasutatakse olemite omavaheliste suhete mõistmiseks;

- dokumentide analüüs: kasutatakse olemasolevate süsteemide, lepingute, standardite ja määruste ülevaatamiseks;

- fookusgrupid: kasutatakse konkreetse grupi ideede ja hoiakute väljaselgitamiseks;

- liideseanalüüs: kasutatakse kahe üksuse – näiteks kahe süsteemi, kahe organisatsiooni, või kahe rolli – omavahelise interaktsiooni mõistmiseks;

- intervjuud: asjaosalistele esitatakse küsimusi, et tuvastada nõudeid või probleeme;

- vaatlus: jälgitakse hetkel eksisteerivat tööprotsessi ehk kuidas praegu reaalselt tööd tehakse. See meetod on eriti tulemuslik, kui tellija ei ole tehniliselt pädev nõudeid ise täpsustama;

- protsessianalüüs: kasutatakse praeguste protsesside mõistmiseks ja potentsiaalsete parenduskohtade tuvastamiseks;

- protsesside modelleerimine: kasutatakse koostöös huvitatud osapooltega eksisteerivate protsesside kaardistamiseks;

- prototüüpimine: olemasolevate nõuete baasilt luuakse mudel ehk prototüüp. Lõppkasutajad annavad prototüübi alusel sisendit uute nõuete ja parenduste kaardistamiseks;

- uuring või küsimustik: kasutatakse struktureeritud viisil teabe hankimiseks klientide, toodete, harjumuste ja vaadete kohta;

- töötoad: kasutatakse teabe hankimiseks klientide, toodete, harjumuste ja vaadete kohta. Tavaliselt toimub grupitööna ja viiakse läbi juhendajaga.

Magistritöös kasutatakse peamiselt poolstruktureeritud intervjuu meetodit. See tähendab, et töö autoril on ette valmistatud teemad ja jutupunktid, mida tellijaga arutada, kuid paigas ei ole konkreetseid küsimusi. Seejärel kasutatakse vaatlust ehk varjutamist, et kinnitada või ümber lükata intervjuu vormis kogutud nõuete paikapidavus.

Kuna kasutatud on inkrementaalset arendusmetoodikat, siis on nõuete kogumisel abiks ka minimaalse töötava toote arendamine – iga versioon arendatavast tarkvarast toimib prototüübina, mille kasutamise järel saavad lõppkasutajad teha parendusettepanekuid.

4.2.2 Äriprotsesside modelleerimiskeel BPMN

Äriprotsesside kujutamiseks on palju erinevaid notatsioone. BPM CBOK v3 (*Business Process Management Common Body Of Knowledge*) toob neist ära seitse olulisemat:

Enimkasutatud äriprotsesside kirjeldamise notatsioonid	
Notatsioon	**Kirjeldus**
BPMN (*Business Process Model and Notation*) 2.0	Standard, mis on loodud Object Management Groupi poolt. Hõlmab endas 103 märki ning on kasulik, kui mudelit on tarvis esitada erinevale publikule (Nt äri ja IT).

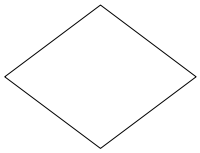
Ujumisrajad (*Swimlanes*)	Ei ole omaette notatsioon, vaid täiendus teistele notatsioonidele. Aitab eristada süsteemide aktoreid.
Vooskeem (*Flowchart*)	Algselt ANSI standardina heaks kiidetud, sisaldab väga lihtsat ja väikest sümbolikomplekti, mida pole standardiseeritud. Kasutatakse protsesside kiireks visandamiseks.
EPC (*Event Process Chain*)	Erinevad sündmused on protsesside käivitajad või lõpetajad. Kasulik keerukate protsesside modelleerimiseks.
UML (*Unified Modeling Language*)	Modelleerimise tehnikate ja sümbolite standardkomplekt. Kasutatakse peamiselt infosüsteemide nõuete kirjeldamiseks.
EPC (*Integrated Definition Language*)	Standard defineerib protsesside sisendid, väljundid, mehhanismid ja kontrollid ning liigitab protsesse üldisemast detailsemani. Kasulik üldise ettevõtte arhitektuuri kirjeldamiseks.
Väärtusahela kaardistamine (*Value Stream Mapping*)	Kasutusel Lean-tootmise juures. Lihtsate sümbolite kogum, mida kasutatakse ressursside ja ajakulu sidumisel protsessidega, et hinnata nende efektiivsust.

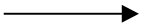
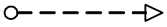
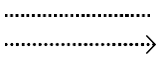
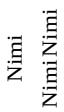
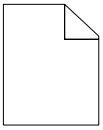
Tabel 1. Enimkasutatud äriprotsesside kirjeldamise notatsioonid [7].

Kuna arendatava projekti analüüs peab olema arusaadav ka laiemale publikule, siis on autori hinnangul mõistlik valida kahe enim levinud notatsiooni vahel. CBOKi järgi on vooskeemi lihtne kasutada, kuid selle piiratud sümbolite kogust jääb puudu, et kirjeldada protsesse detailselt. Vooskeemi sümbolid ei ole standardiseeritud ja seetõttu võib selle lugemisel kergelt tekkida mitmeti mõistetavusi [7].

BPMN võimaldab kirjeldada protsesse sarnaselt vooskeemile, kuid detailsemal ning standardiseeritumal moel. BPMNi eesmärk on pakkuda märgistikku, mis on ärikasutajatele piisavalt intuitiivne ja samas võimeline esitama tehnilistele kasutajatele vajalikul tasemel keerukust. BPMN on avatud standard, mis on koostatud varasemate tootjapõhiste standardite põhjal ning on tänaseks laialdaselt kasutusele võetud paljude tarkvaaratootjate poolt kui peamine protsesside kirjeldamise märgistik [8].

BPMNis on kolm peamist vooelementide rühma, mille elementidest BPMNi protsessijooniseid koostatakse. Igas elemendirühmas paiknevad elemendid on välimuselt sarnase kujuga ning erinevad teineteisest täpsustava ikooni poolest. Alljärgnevas tabelis on toodud BPMNi vooelementide rühmad koos sinna kuuluvate sümbolite näidistega. Lisaks põhielementidele on BPMNis kasutusel ka alamprotsessi tüüpi tegevus, millega piiritletakse protsessi osad, mida soovitakse näidata eraldi protsessina.

Enimkasutatud äriprotsesside kirjeldamise notatsioonid		
Element	Kirjeldus	Sümbol
1. Vooelemendid		
Sündmus (*Event*)	Sündmus on midagi, mis „juhtub“ protsessi täitmise käigus. Sündmused mõjutavad protsessi voogu ning tavaliselt on neil põhjus ja tagajärg. Sündmuseid on kolme tüüpi vastavalt sellele, millal need voogu mõjutavad: Algus (*Start*), Vahepealne (*Intermediate*), Lõpp (*End*).	
Tegevus (*Activity*)	Tegevus on üldine termin tööle või toimingule, mida organisatsioonis tehakse. Tegevus võib olla atomaarne või mitte-atomaarne (kombineeritud). Protsessiskeemil kasutatakse 3 tüüpi tegevusi: Protsess, Alamprotsess ja Toiming. Toiming ja alamprotsess kujutatakse protsessiskeemil ümarate nurkadega ristkülikuna. Protsess kujutatakse basseini sisuna.	
Lüüs (*Gateway*)	Lüüse kasutatakse protsessi hargnevuste ja koonduvuste kirjeldamiseks. Hargnevused võivad olla tingimuslikud või paralleelsed. Koonduvused võivad olla sulanduvad või liituvad. Lüüsi tüübi tähistamiseks kasutatakse sisemisi markereid.	
2. Ühenduselemendid		

Järgnevusvoog (*Sequence Flow*)	Järgnevusvoogu kasutatakse protsessi tegevuste täitmise järjekorra tähistamiseks.	
Sõnumivoog (*Message Flow*)	Sõnumivoogu kasutatakse protsessi osapoolte vahelise sõnumiedastuse tähistamiseks. Osapoolte valmisolek sõnumeid saata ja vastu võtta peab olema protsessis tähistatud. BPMNis paigutatakse erinevad osapooled erinevatesse basseinidesse.	
Seos (*Association*)	Seoseid kasutatakse vooelementidele info lisamiseks. Igale vooelemendile võib lisada tekstilist või graafilist lisainfot. Vajadusel võib info (andmete) liikumise suunda illustreerida noolega.	
3. Ujumisrajad		
Bassein (*Pool*)	Basseiniga tähistatakse protsessi osapoolt. Ühtlasi võib basseini kasutada teatud tüüpi tegevuste visuaalseks grupeerimiseks eesmärgiga lihtsustada protsessiskeemi loetavust.	
Rada (*Lane*)	Rada on basseini alamgrupp, mis ulatub basseini ühest otsast teise. Radasid kasutatakse tegevuste liigitamiseks ja loogiliseks grupeerimiseks.	
4. Artefaktid		
Andmeobjekt (*Data Object*)	Andmeobjekte käsitletakse artefaktidena, sest need ei oma otsest mõju portsessi järgnevus- ja sõnumivoole. Andmeobjektid annavad infot selle kohta, mida tegevuse tegemiseks on tarvis või milline on tegevuse tulem.	
Grupp (*Group*)	Gruppe kasutatakse samasse kategooriasse kuuluvate tegevuste rühmitamiseks. Kategooria nimi märgitakse grupi raami ülaäärde. Grupp ei mõjuta protsessi järgnevusvoogu. Gruppe kasutatakse dokumenteerimise või analüüsi eesmärkidel.	
Märkus (lisatud seosega) (*Annotation*)	Märkused on üks viisidest, millega on võimalik protsessiskeemile kanda lisainfot protsessi lugejale.	Kirjeldav tekst

Tabel 2. BPMNi põhielemendid [9].

4.2.3 Kasutusmallide kirjeldamine

Funktsionaalsete nõuete kirjeldamiseks on võimalik kasutada kasutajalugusid või kasutusmalle. Kasutajalood on keskendunud tegevuse lõpptulemusele ehk eesmärgile, samas kui kasutusmallid on detailsemad ning kirjeldavad, kuidas süsteem käitub [10].

Selle töö raames on eelistatud kasutusmalle kasutajalugudele, kuna ainus eesmärk ei ole võimalikult lühike arendustsükkel, vaid vajalik on anda ka kõrvaltvaatajale põhjalik ülevaade süsteemi toimimisest.

Jacobsoni kohaselt on kasutusmall seotud toimingute jada, mille algatab aktor mingi eesmärgi saavutamiseks – see on konkreetne tegevus süsteemis. Aktori ja kasutaja vahel on erinevus. Kasutaja on igaüks, kes süsteemi kasutab, aktor seevastu esindab rolli, milles kasutaja saab olla. Aktor on tüüp- või klasskasutaja – kasutaja on aktori klassi konkreetne näide. Sama kasutaja saab olla mitmes erinevas rollis.

Kasutusmallide tuvastamiseks on Jacobson soovitanud küsida järgmisi küsimusi: [11]

- Millised on iga aktori peamised ülesanded?
- Kas aktor vaid loeb või ka uuendab süsteemis olevaid andmeid?
- Kas aktor peab süsteemi teavitama väljaspool süsteemi toimuvatest muudatustest?
- Kas aktorit tuleb ootamatutest muudatustest teavitada?

4.3 Süsteemianalüüsi metoodikate ülevaade

Selles peatükis tutvustatakse töös kasutatavaid süsteemianalüüsi tulemuste visualiseerimise metoodikaid.

Andmete modelleerimiseks kasutatakse olemi-suhte diagrammi (*Entity Relationship Diagram*). Ülejäänud süsteemianalüüsi tulemuste esitamiseks kasutatakse UML (*Unified Modeling Language)* diagramme – kasutusmallide diagrammi (*Use Case Diagram*) ja tegevusdiagrammi (*Activity Diagram*).

4.3.1 Funktsionaalsete ja mittefunktsionaalsete nõuete kirjeldamine

Käesoleva töö funktsionaalsete ja mittefunktsionaalsete nõuete kaardistamiseks kasutatakse tarkavaraarenduses laialt levinud FURPS-mudelit. FURPS-mudel töötati välja Robert Grady ja Hewlett-Packardi poolt. IBM arendas mudelit edasi ja uueks nimeks sai FURPS+. FURPSi alusel liigitatakse kasutaja poolt kirjeldatud nõuded funktsionaalseteks ja mittefunktsionaalseteks nõueteks [12].

FURPS on lühend järgmistest sõnadest: [13]

- Funktsionaalsus (*Functionality*): See punkt vastab küsimusele: mida lõpptoode peab tegema? Ehk selle punkti käigus kaardistatakse rakenduse funktsionaalsus.

- Kasutatavus (*Usability*): kes toodet kasutab? Kuidas nad seda kasutavad? Millist kasutajaliidest on tarvis? Üks sageli tähelepanuta jäetud valdkond on kasutusjuhendid ja koolitused.

- Usaldusväärsus (*Reliability*): Millised on ootused süsteemi tööaja osas? Mida peetakse aktsepteeritavaks süsteemirikkeks? Kui kiiresti peaks süsteem suutma tõrkest taastuda? Kui pikk peaks olema keskmine ebaõnnestumiste vaheline aeg?

- Toimivus (*Performance*): milline on süsteemi oodatav jõudlus? Nõuete kaardistamise käigus tuleb mõelda rakenduse kiirusele, tõhususele, kättesaadavusele, täpsusele, viiteajale, taastumisajale ja ressursikasutusele.

- Toetatavus (*Supportability*): kui lihtne peaks olema süsteemi testimine ja kuidas seda läbi viia? Millised on nõuded süsteemi hallatavusele? Milliste parameetrite alusel peaks süsteem seadistatav olema?

FURPS+ hõlmab täiendavaid projekteerimise, juurutamise, kasutajaliidese ja taristu piiranguid: [14]

- Disainipiirangud: suunavad rakenduse kavandamise nõudeid. Näiteks võib nõuda, et kasutataks relatsioonilist, mitte No-SQL andmebaasi.

- Juurutamispiirangud: piiravad rakenduse arendamisel ja juurutamisel kasutatavaid tehnoloogiaid ja raamistikke. Näiteks programmeerimiskeele või veebiarenduse platvormi ja raamistike valik.

- Liidestamispiirangud: kehtestavad, milliste väliste süsteemidega ja kuidas peab süsteem suhtlema. Näiteks API või sõnumiedastusprotokolli ja andmetüübi piirang.

- Füüsilised piirangud: mõjutavad süsteemi käitamiseks kasutatavat riistvara – näiteks kuju, suurust ja kaalu.

4.3.2 Relatsiooniline andmemudel

Iga infosüsteemi planeerimise faasis tuleb läbi viia ka andmemudeli loomine. Seeläbi on võimalik kindlustada loodud andmemudelite ning nende alusel loodud andmebaaside struktuuride aegpüsivus. Andmemudeli ja selle alusel loodud andmebaasi struktuuri aegpüsivuse all mõistetakse andmemudeli arengu kriteeriumit, kus juba projekteeritud ja rakendatud andmemudelid säilitavad oma seoste struktuuri pikema aja jooksul muutumatuna ning muudatused mudelis on kirjeldatavad peamiselt kui uute komponentide (olemite ja olemi atribuutide) lisamine mudelisse, nende seostamine omavahel ning varasemalt mudelis eksisteerivate komponentidega [15].

Magistritöös kasutatakse andmemudeli modelleerimisel „varese jala“ notatsiooni, põhjusel, et see on erinevatest kasutusel olevatest notatsioonidest kõige ülevaatlikum ja metoodiliselt ühetaolisem [15].

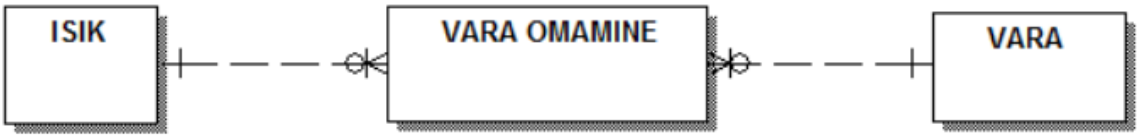
Vastavalt Riigi Infosüsteemi Ameti poolt välja antud relatsiooniliste andmemudelite koostamise juhendile peavad modelleerimisel olema täidetud järgmised nõuded: [15]

1. graafiliselt on eristatavad olemid (*entity/table*), mis tähistatakse ristkülikuna ja vaated (*view*), mis tähistatakse ümar-nurkadega ristülikutena;

2. igal olemil on primaarvõti (*primary key*), mis on surrogaatvõti ehk kunstlikult genereeritud väärtusega võti, mis ei oma seost reaalse eluga;

3. olemite vaheliste suhete kirjeldamisel eristuvad graafiliselt järgmised seose tugevused: üks; üks või mitte ühtegi; üks või mitu; mitte ühtegi, üks või mitu.

Alloleval joonisel on näide olemi-suhte diagrammist, kus isik on kindla ajaperioodi jooksul omanud mingit konkreetset vara.

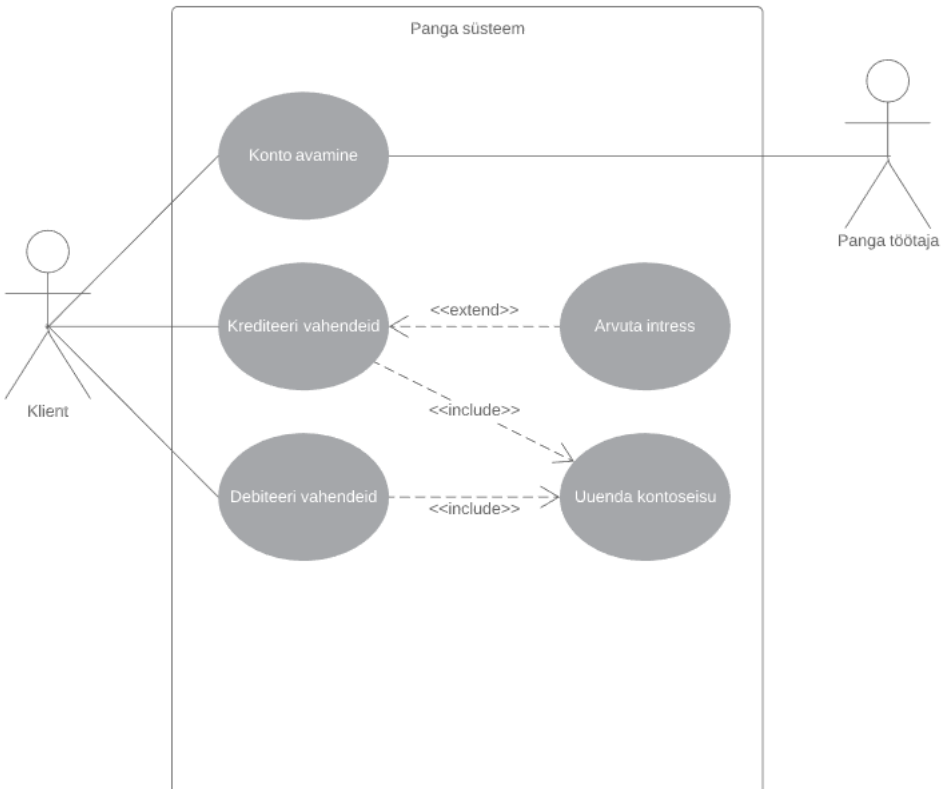


Joonis 1. ERD näide [15].

4.3.3 Kasutusmallide diagramm

Kasutusmallid aitavad saada parema ülevaate süsteemi funktsionaalsetest nõuetest. Lisaks kirjeldamisele kuvatakse kasutusmalle diagrammina, kus paremal ja vasakul servas asuvad aktorid, ning keskel olevas kastis on tegevused, mida need aktorid algatavad [16].

Alloleval joonisel on näha lihtne näide panga tuumsüsteemi kasutusmallide diagrammist. Aktoriteks on klient ja panga töötaja, võimalikud kasutusmallid on konto avamine, vahendite krediteerimine, vahendite debiteerimine, intressi arvutamine ja kontoseisu uuendamine. *Include* märksõna tähendab, et üks kasutusmall eelneb alati teisele, *extend* märksõna puhul võib, kuid ei pruugi, üks kasutusmall olla teise eelduseks.



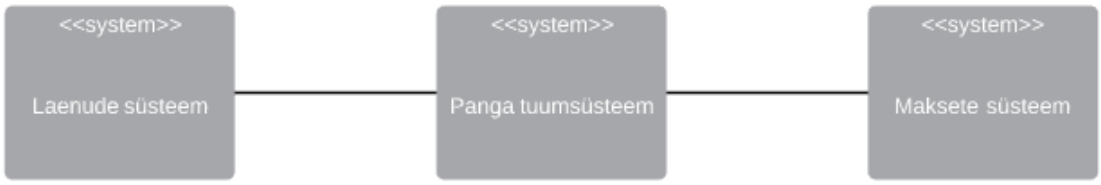
Joonis 2. Kasutusmallide diagramm pangasüsteemi näitel (Allikas: autori koostatud).

4.3.4 Kontekstimudel

Süsteemianalüüsi varases etapis on vajalik defineerida süsteemi piirid ehk mis kuulub arendatava süsteemi alla ja mis mitte. See tähendab tööd asjaosalistega, et kaardistada, millised funktsionaalsused ja võimekused peavad toimuma loodava lahenduse sees ning millised ülesanded jäävad liidestuvate süsteemide kanda või manuaalseks. Selle käigus on mõistlik kaardistada ka olemasolevate süsteemide funktsionaalsused, et ei tekiks funktsionaalsuste dubleerimist. Need otsused tuleks vastu võtta võimalikult varajases süsteemianalüüsi faasis, et piirata arenduse kulusid ning vähendada süsteemi nõuete ja arhitektuuri defineerimiseks kuluvat aega [4].

Kontekstimudelil kujutatakse loodavat infosüsteemi ning temaga mingil moel suhestuvaid süsteeme. Mudel ei näita, millised suhted on süsteemide vahel – näiteks kas liidestussüsteem loeb või kirjutab põhisüsteemist andmeid. Süsteemid võivad olla seotud sama andmebaasi jagades või suhelda API kaudu kuid kontekstimudel nii detailseid seoseid ei kirjelda, seega peab tervikpildi saamiseks kontekstimudelit alati vaatlema koos detailsemate äriprotsesse ja süsteemiarhitektuuri kujutavate diagrammidega [4].

Alloleval joonisel on kujutatud lihtne kontekstmudel pangasüsteemide näitel. Tuumsüsteem on peamine vaatluse all olev süsteem, mis suhestub laenude ja maksete süsteemidega.



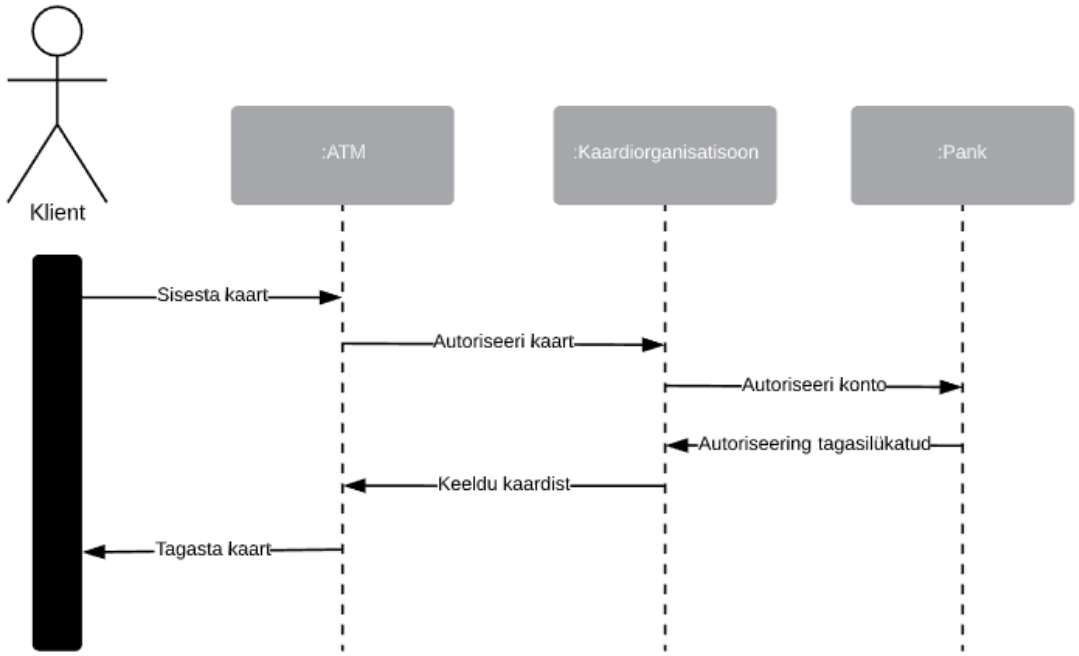
Joonis 3. Kontekstimudel panga süsteemide näitel (Allikas: autori koostatud).

4.3.5 Järgnevusdiagramm

Järgnevusdiagramm kuvab ühes kasutajaloos olevate objektide vahelist suhtlust ajalises järjestuses. Järgnevusdiagramm tuleks luua iga olulisema kasutajaloo kohta. Osalisteks on objektid, mis kujutatakse ristkülikutena joonise ülaosas. Objekti alumisest küljest väljub vertikaalne joon, mida nimetatakse objekti elujooneks [17].

Esimene sõnum tuleb kindlakstegemata allikast, ülejäänud sõnumite algatajateks on objektid. Sõnumi või väljakutse edastamist teisele osalisele kujutatakse noolega, mille peale on kirjutatud meetodi nimi ning vajadusel parameetrid. Tagastatavad väärtused, kujutatakse katkendliku joonena, mille ots on suunatud sõnumi vastuvõtja poole. Noole peale kirjutatakse tagastatava objekti, väärtuse või sõnumi nimi. Järgnevusdiagrammi eeliseks on see, et ta annab detailse ülevaate sellest, mis kasutusmallide piires toimub [17].

Alloleval joonisel on kujutatud lihtsat järgnevusdiagrammi, kus klient sisestab kaardi sularahaautomaati, kaardiorganisatsiooni süsteem autoriseerib kaardi, ning panga süsteem autoriseerib kliendi konto.



Joonis 4. Järgnevusdiagramm ATM-süsteemi näitel (Allikas: autori koostatud).

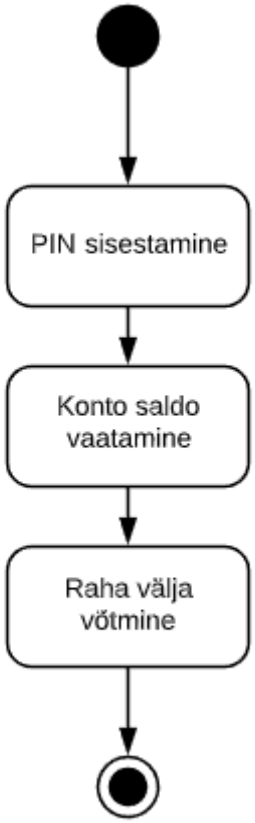
4.3.6 Tegevusdiagramm

Süsteemi olekute kujutamiseks UML-metoodikaid järgides on kaks peamist valikut: seisundimuutuse diagramm või tegevusdiagramm.

Seisundimuutuste diagrammi sündmuspõhine modelleerimine näitab, kuidas süsteem reageerib välistele ja sisemistele sündmustele. See põhineb eeldusel, et süsteemil on

piiratud arv olekuid ning et sündmused (stiimulid) võivad põhjustada ülemineku ühest olekust teise. Sommerville'i sõnul on sellist tüüpi diagramm eriti sobiv reaalajas toimivate süsteemide jaoks. Tegevusdiagramm on seisundidiagrammi erivariant, mis keskendub sündmustele ja tegevustele, mitte süsteemi olekutele [4].

Sellest võime järeldada, et seisundimuutuste diagramm on pigem kasulik automaatselt töötavate rakenduste, näiteks andmelaadimisskriptide või muu automaatika kirjeldamiseks. Tegevusdiagramm on see-eest efektiivsem, kui protsessis on palju interaktsioone lõppkasutajaga.



Joonis 5. Tegevusdiagramm ATM-süsteemi näitel (Allikas: autori koostatud).

4.4 Arhitektuuri kirjeldamise metoodikate ülevaade ja valik

Selles peatükis tutvustatakse töös kasutatavaid rakenduse arhitektuuri esitamise metoodikaid. Arhitektuuri visualiseerimiseks kasutatakse komponentdiagrammi

(*Component Diagram*), järgnevusdiagrammi (*Sequence Diagram)* ja evitusdiagrammi (*Deployment Diagram*).

4.4.1 Komponentdiagramm

Komponentdiagramm jaotab väljatöötatava süsteemi funktsionaalsuse alusel erinevateks tükkideks. Iga komponent vastutab kogu süsteemi jaoks ühe selge eesmärgi eest ja suhtleb teiste oluliste elementidega vaid teadmisvajaduse põhimõttel [18].

Komponentdiagrammi tähtsamad osad: [18]

- Komponent tähistab süsteemi osa, mis täidab mingit konkreetset funktsiooni ning mis on asendatav.
- Liidesesümbolid, mille otsas on ring, tähistavad liidest, mida komponent pakub. Seda sümbolit kutsutakse tihti „pulgakommiks“.
- Liidesesümbolid, mille otsas on vaid poolring, n-ö „pistikupesad“, tähistavad liidest, mida komponent vajab. Mõlemal juhul paigutatakse liidese nimi liidesesümboli lähedale.
- Pordid kuvatakse süsteemi või komponendi ääres oleva ruudu abil. Sageli kasutatakse komponendi nõutavate ja pakutavate liideste ühendamiseks porti.

Joonis 6. Komponentdiagramm pangasüsteemi näitel (Allikas: autori koostatud).

4.4.2 Evitusdiagramm

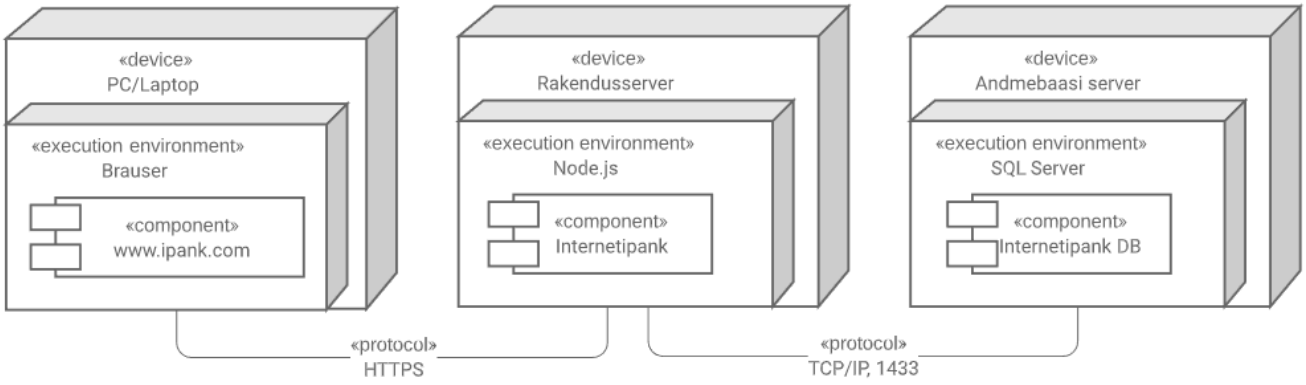
Evitusdiagrammi kasutatakse klient-serveri ja hajutatud süsteemide visualiseerimiseks, see on klassidiagrammi erivariant, mis keskendub klasside asemel süsteemi sõlmedele. Evitusidagrammil kujutatakse süsteemi nii nagu ta päriselus toimima peaks – tavaliselt on fookus riistvarakomponentidel ja neid ühendavatel lülidel, samas saab kasutada ka

tarkvaraliste sõlmede (*node*) kujutamiseks. Evitusdiagrammi võiks ka kutsuda süsteemi füüsilise arhitektuuri diagrammiks ja see on vajalik etapis, kus arendaja annab süsteemi üle IT-haldusele, kes hakkab igapäevaselt süsteemi toimimist (serverite haldus, reliisid, uuendused) tagama.

Peamised küsimused, millele evitusdiagrammi välja töötades peaks mõtlema: [18]

- Milliseid olemasolevaid süsteeme peab loodava rakendusega liidestama?
- Millised on nõuded süsteemi vastupidavusele (nt riistvara rikke korral)?
- Kes ja kuidas süsteemiga suhtleb ja kuidas seda tehakse?
- Millist tarkvara, sealhulgas operatsioonisüsteemi ja ning andmevahetusprotokolle, süsteem kasutab?
- Millise riist- ja tarkvaraga kasutajad otseselt suhelda saavad (arvutid, serverid, brauserid jne)?
- Kuidas IT-haldus monitoorib süsteemi peale selle kasutuselevõttu?
- Kui turvaline peab süsteem olema (tulemüür, füüsiliselt turvalisus, riistvara)?

Alloleval joonisel on lihtsustatud näide internetipanga evitusdiagrammist. Keskel asub internetipanga rakendus rakendusserveris, mis on ühendatud üle 1433 pordi MS SQL Serveri andmebaasiserveriga ning üle interneti ligipääsetav klientidele brauseris.



Joonis 7. Evitusdiagramm internetipanga näitel (Allikas: autori koostatud).

5 Alternatiivsete lahenduste võrdlus

Siin peatükis analüüsitakse erinevaid võimalusi KMV-pettuse monitooringusüsteemi loomiseks ning võrreldakse ettevõttesisest arendust karbitoote ja sisseostetud lahendustega.

5.1 Kaardipettuste tuvastamise ja ennetamise meetodite analüüs

Kaardipettuse tuvastamise ja vältimise süsteemid võib tinglikult jagada viide rühma: [19]

1. Maksekoha turvanõuded – PIN, CVV, kaheastmeline autentimine.
2. Tehingute blokeerimise reeglid – tehingud blokeeritakse reaalajas vastavalt kodeeritud, mudelipõhiste või skooringupõhistele reeglitele.
3. Skooringureeglid – eeldefineeritud reeglite tasemete täitumisel hangib lõppkasutaja kaupmehe ja tehingute kohta täiendavat infot.
4. Mudelipõhine süsteem – tuvastab statistilise mudeli põhjal potentsiaalsed pettusekahtlusega tehingud. Vajab rakendamiseks suurt andmehulka.
5. Uurijad – lõppkasutajateks on uurijad, kes hangivad kahtluse korral lisainformatsiooni ning otsustavad edasiste tegevuste üle.

Käesoleva projekti tellimisel kaalusid huvitatud osapooled neist tõsisemalt punkte 2, 3 ja 4.

Reaalajas tehingute blokeerimise kahjuks räägib see, et veamäär peab olema väga madal ning valepositiivsete tulemuste suurem hulk hakkab takistama normaalset äritegevust. See tähendab, et eelnevalt on vaja kasutatavad reeglid saada piisavalt täpseks ning seejärel on võimalik liikuda reaalajas töötava rakenduse juurde.

Mudelipõhise lahenduse loomiseks ja täiustamiseks on vaja suurt andmemassiivi ning andmeanalüütikute suurt ajalist panust. Selle analüüsimise ja treenimisega on andmeanalüütikud magistritöö kirjutamise ajal juba alustanud, kuid selle töö skoopi ei ole see teema ajaraami sobimatuse ning ärisaladuse kaitse tõttu mahtunud.

Olemasolevaid asjaolusid silmas pidades on kõige kuluefektiivsem luua skooringureeglitel põhinev süsteem, mida täiendavad lõppkasutajatest uurijad, kes annavad olukorrale lõpliku hinnangu. Tingimuste muutumisel tulevikus on võimalik teostada ka punktide 2 ja 4 rakendamine, kuid see ei mahu käesoleva töö skoopi.

5.2 Alternatiivsete lahenduste võrdlus

Siin peatükis on vaadeldud potentsiaalseid alternatiive magistritöös analüüsitavale lahendusele. Võrdlusesse on valitud tellija jaoks olulisemad funktsionaalsed ja mittefunktsionaalsed nõuded, mida on pikemalt kirjeldatud peatükkides 7.1 ja 7.2.

Käesolev töö on suunatud finantsettevõtte kaardimaksete vastuvõtmise pettuste monitooringu osalisele automatiseerimisele ja täiustamisele. Seeläbi on võimalik kokku hoida andmete sisestamiseks ja sorteerimiseks kuluvat aega, potentsiaalset vigade arvu ning muuta pettuste tuvastamist efektiivsemaks. Võimalik on toode sisse osta karbitootena, kuid need on enamasti suunatud maailma mõistes keskmistele ja suurtele finantsasutustele, seega ei ole kulu vaatest finantsettevõtte jaoks mõistlikud. Jääb üle tellida infosüsteem koostööpartnerilt või arendada see iseseisvalt – viimane valik on väiksemate mahtude juures parima tulude-kulude suhte ning paindlikkusega.

Turul on olemas mitmeid karbilahendusi, näiteks Worldline Pay Online Watcher. See on reeglipõhine reaalajas pettusemonitooring, mis hõlmab endas nii pettuste tuvastamist, tagastusi kui ka juhtumite haldamise süsteemi [5]. Puuduseks on teenuse kõrge hind, mis jääb vahemikku 20 000 – 30 000 eurot aastas arvestamata integreerimise kulusid – arendades ettevõttesiseselt on võimalik säästa kuludelt ning tagada efektiivsem muutuste juhtimise tulevikus.

Nõue	Django arendus	Worldline Pay Online Watcher	Tableau	Exceli edasiarendus	Tellida arendus väjast
5 aasta keskmine kulu aastas peab jääma alla 15 000 € (arendus ja haldus) (PLMFN12)	X	-	X	X	-
Kasutaja peab saama monitooringu reegleid lisada ja muuta (UC2)	X	X	-	X	X
Klientide teavitamine peab toimuma (pool) automaatselt (UC4)	X	X	-	X	X
Koondinfo (UC1) laadimisel maksimaalne lubatud kestus on 2 sekundit (JÕMFN1)	X	?	-	-	X
Ettevõtte tehingute (UC3) laadimisel maksimaalne lubatud kestus on 1 sekund (JÕMFN1)	X	?	-	-	X
Lõppkasutajal ei tohi olla kasutuselevõtu testi (*User Acceptance Testing*) käigus kasutajakogemusele etteheiteid (KAMFN6)	X	?	-	-	?

Tabel 3. Alternatiivsete lahenduste võrdlus (Allikas: autori koostatud).

6 Ärianalüüsi tulemused

Selles peatükis määratakse loodava süsteemi huvitatud osapooled (*Stakeholders*). Osapooltega toimunud poolstruktureeritud intervjuude alusel luuakse lahenduse ärikirjeldus ning peamised ärinõuded. Lisaks visualiseeritakse eeltoodud sisendite põhjal äriinfo mudel olemi-suhte diagrammina

Kasutades BPMN notatsiooni kirjeldatakse monitooringulahenduse peamised tööprotsesse:

- UC1: Pettusekahtlusega klientide kuvamine;
- UC2: Pettuse tuvastamise reeglite lisamine;
- UC3: Kliendi kahtlaste tehingute ajaloo vaatamine;
- UC4: Kliendi pettusest teavitamine.

6.1 Huvitatud osapooled

Huvitatud osapoolte kaardistamine ja analüüs hõlmab nende sidusrühmade tuvastamist, keda kavandatav infosüsteem mõjutab. Selleks on võimalik kasutada erinevaid metoodikaid nagu näiteks osapoolte nimekirjade loomine või sidusrühmade kaardi loomine [6].

Käesoleva töö raames kasutatakse vastutusmaatriksi (RACI). See metoodika jagab huvitatud osapooled gruppidesse vastavalt oma rollile projekti õnnestumise ees ning annab seeläbi hea ülevaate vastutuse jagunemisest lahenduse loomisel: [6]

- Vastutav (*Responsible - R)*: Isikud, kes teostavad ülesande läbiviimiseks vajalikud tööd.

- Aruandev (*Accountable - A)*: Isik, kes on ülesande eduka lõpetamise eest vastutav ning kes võtab projektis vastu olulisemaid otsuseid. Siia gruppi pannakse vaid üks isik.

- Nõuandev (*Consulted - C)*: Asjaosalised, kellelt küsitakse nõu või infot projekti edukaks läbiviimiseks.

- Informeeritud (*Informed – I*): Asjaosalised, keda informeeritakse jooksvalt projekti staatusest ja tulemustest. See grupp erineb nõuandvast selle poolest, et info liikumine on siin ühesuunaline (analüütikult asjaosalisele).

Huvitatud osapool	**RACI**	**Roll arenduse juures**
KMV-osakonna juht	A	Lahenduse tellija.
KMV-müügiinsener	C	Töötaja, kes hakkab rakendust igapäevaselt kasutama. Koostöös müügiinseneriga kaardistati süsteemi funktsionaalsed nõuded.
Finantsjuht	I	Töövahendite arenduse juht raporteerib töökavast ja tehtud töödest finantsjuhile.
IT-juht	C	IT-juhiga kaardistati süsteemi mittefunktsionaalseid nõudeid, tulevast arhitektuuri ning taristut puudutavaid teemasid.
Süsteemiadministraatorite tiimijuht	C	Süsteemiadministraatoritega lepiti kokku õiguste ja ligipääsude haldus, infrastruktuuri tehniline lahendus ning tarnevoo (*Deployment Pipeline)* põhipunktid.
Infoturbe juht	C	Infoturbe juhiga lepiti kokku süsteemi logide formaat ja logimise tehniline lahendus.
Andmeanalüütik	C	Andmeanalüütiku huviks oli saada ligipääs süsteemi andmetele, et kasutada neid regulatiivseks raporteerimiseks.
Töövahendite arenduse juht	R	Töövahendite arenduse juht osaleb selle rakenduse arendamisel ärianalüütiku, süsteemianalüütiku ja arendaja rollis.

Tabel 4. Huvitatud osapoolte vastutusmaatriks (RACI) (Allikas: autori koostatud).

6.2 Intervjuud

Töö käigus viidi 2019. aasta neljandas kvartalis ja 2020. aasta esimeses kvartalis läbi poolstruktureeritud intervjuud lahenduse huvitatud osapooltega. Intervjuud puudutasid järgmisi alamteemasid:

- Pettuse tuvastamise reeglid;
- Süsteemi funktsionaalsed nõuded;
- Liidestused väliste süsteemidega;
- Turvanõuded;
- Ligipääsude ja õiguste haldus;
- Rakenduse infrastruktuur (rakendusserver, veebiserver, konteinerid);
- Süsteemi logide haldus;
- Muud teemad.

Intervjuud olid aluseks infosüsteemi ärikirjelduse, nõuete ja protsesside loomisel.

6.3 Loodava infosüsteemi ärikirjeldus

Finantsettevõtte KMV-pettuse monitooring on KMV-osakonna töötajatele ligipääsetav sisevõrgus. Töötajatel on võimalik jälgida kaupmeeste riskiindikaatorite täitumist, otsida kaupmehi pettuste baasist erinevate tunnuste järgi, sisestada pettuse tuvastamise reegleid ja riskiindikaatoreid, vaadata kaupmehe pettusekahtlusega tehingute ajalugu ning teavitada kaupmehi pettusekahtlusest. Süsteem saadab agregeeritud info põhjal automaatselt pettusekahtlusega tehingute info KMV *gateway* teenusepakkuja (edaspidi KGT) süsteemi.

Pettuse tuvastamise reeglite lisamine

Süsteemi on võimalik lisada erinevaid parameetreid, mille alusel arvutatakse pettuse skoorid. Kasutaja näeb kuva, kus on võimalik lisada või kustutada agregeeritud väärtuste veerge. Neile väärtustele saab seada piirmäärad, mille täitumise järel nad värvuvad punaseks. Kasutaja saab lisaks seada geograafilisi või ettevõtte nimega seotud piiranguid IF, AND ja/või OR tingimusi.

Reegleid selle töö raames detailselt ei kirjeldata, kuid nendeks on enamasti maksetega seotud suhtarvud või muud tunnused. Töö tulemusel arendatava lahendusega on väikeste täienduste järel võimalik monitoorida ka teisi reeglipõhiseid protsesse või süsteeme.

Pettusekahtlusega klientide kuvamine

Pettusekahtlusega laupmehed kuvatakse rakenduse esilehel. Kui sisestatud reegli skooril täitub määratud limiit, värvub lahter punaseks. Lisaks kuvatakse kliendi ettevõtte nimi, ärinimi, sisemine identifikaator, kaardiorganisatsiooni identifikaator, ettevõtte käive kuupõhiselt ning ettevõtte pettusekahtlusega käive kuupõhiselt.

Igal real kuvatakse ühte klienti kuupõhiselt. Kasutajal on võimalik andmeid kõikide veergude kaupa järjestada kasvavalt või kahanevalt. Kasutajal on võimalus konkreetset klienti otsida ettevõtte nime- või ärinimepõhiselt. Kasutajal on võimalik valida vaates kuvatavate ridade arvu.

Kliendi kahtlaste tehingute ajaloo vaatamine

Kui kasutaja vajutab avakuval ettevõtte nimele, suunatakse ta vaatesse, kus on detailsemalt kuvatud kogu kliendi pettusekahtlusega tehingute ajalugu. Selles vaates ei ole tehingute summad kuu kaupa agregeeritud, vaid iga tehing kuvatakse iseseisvalt.

Kliendi pettusest teavitamine

Kasutajal on võimalik kliendiga pettusekahtluse asjaolude väljaselgitamiseks kontakteeruda. Selle jaoks kuvatakse iga avakuva rea juures nupp, mida vajutades tuleb lahti e-kirja aken eeltäidetud põhjaga. Kirjas kuvatakse kasutaja poolt eeldefineeritud tekst ning pettusekahtlusega tehingute aeg, koht ja summa. Süsteem lisab automaatselt saaja väljale ettevõtte e-maili aadressi. Kasutajal on võimalik e-kirja sisu enne saatmist muuta.

6.4 Peamised ärireeglid

Järgnevalt on välja toodud süsteemi ärireeglid. Ärireeglite omavahelised suhted on esitatud peatükis 6.5 (Relatsiooniline andmemudel) – need on kujutatud olemi-suhte diagrammina.

Tähis	Ärireegel
ÄR1	Kaardimaksete vastuvõtmise tehingute jaoks peab ettevõte omama kliendilepingut. Üks kliendileping on vormistatud ühe ettevõttega. Aja jooksul võib ettevõttel olla mitu kliendilepingut.
ÄR2	Ühel ettevõttel võib olla üks esindaja. Ühel isikul võib olla samaaegselt mitu ettevõtet.
ÄR2	Ühel ettevõttel saab olla null kuni mitu tehingut. Ühel tehinguga saab olla seotud vaid üks ettevõte.
ÄR3	Ühel ettevõttel saab olla null kuni mitu pettusekahtlusega tehingut. Ühel kliendil saab olla null kuni mitu pettusekahtlusega tehingut.
ÄR4	Rakendusse sisse logimiseks peab isik omama vastavat (AD) õiguste rolli. Õiguste roll peab kuuluma ligipääsu lubavasse gruppi.
ÄR5	Üks tehing saab olla vaid ühte tüüpi. Tehingu tüübiga võib olla seotud üks kuni mitu tehingut.
ÄR6	Ühele kaupmehele kehtib null või mitu reeglit. Korraga saab kehtida üks või mitu reeglit.
ÄR7	Ühele firmale saab korraga kehtida null kuni mitu pettuse reeglit. Mõni reegel võib olla ka firma külge sidumata.
ÄR8	Üks maksekoht saab olla vaid ühte tüüpi. Maksekoha tüübiga võib olla seotud üks kuni mitu maksekohta.
ÄR9	Ühel kliendil peab olema salvestatud üks või mitu kontakti. Iga kontakt peab olema mingit tüüpi aga kontakti tüübiga võib olla seotud null kuni mitu kontakti.
ÄR10	Iga isik peab olema mingit tüüpi (nt töötaja, klient).

	Iga isiku tüübiga ei pea olema seotud isikut.

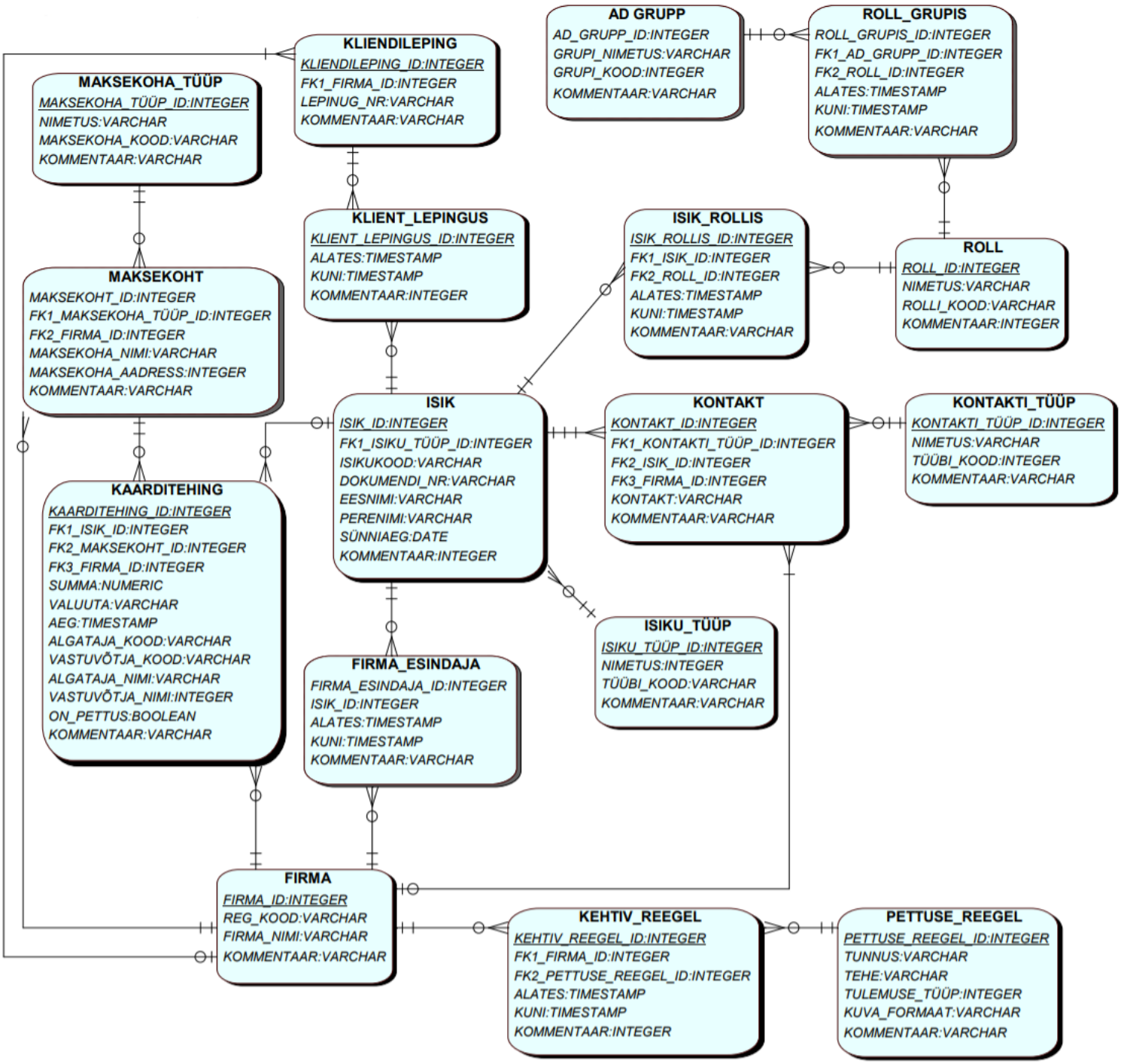
Tabel 5. KMV-monitooringu ärireeglid (Allikas: autori koostatud).

6.5 Relatsiooniline andmemudel

Üldjuhul kasutatakse ärireeglite modelleerimiseks UML-klassidiagrammi – kui kasutatakse objektorienteeritud (OOP) lähenemist, siis on kvaliteetse modelleerimise korral võimalik klassidiagrammid otse ümber kirjutada koodi klassideks. See tagab äri (mitte-IT) inimestele hea ülevaate süsteemist ning muudab koodi üldjuhul paremini loetavaks, muudetavaks ja taaskasutatavaks [20].

Kuna süsteemi arendatakse lisaks OOP-meetodile ka protseduurilisel ja imperatiivsel meetodil, siis autori hinnangul annab olemi-suhte diagramm loodava süsteemi struktuuri andmemudelina paremini edasi. Peamiseks põhjuseks, miks käesoleva projekti teatud osade juures on eelistatud protseduurilist meetodit, on Django ORMi (*Object Relational Mapper*) eripärad. Autori hinnangul on funktsioonidel põhinevad ORM-päringud paremini loetavad ja hallatavad võrrelduna klassidel põhinevate päringutega. Valitud programmeerimisparadigmade, raamistike ja tehnoloogiate valikuid on põhjendatud peatükis 7.3.

Äriinfo mudel on kujutatud olemi-suhte diagrammina. Diagrammil on välja toodud rakenduse jaoks vajalikud olemid ning nende atribuudid. Enamik tabelitest asub juba eksisteerivate süsteemide andmebaasides – näiteks olem ISIK asub ettevõtte tuumsüsteemi baasis, olem AD_GRUPP asub Active Directory süsteemi baasis ning olem KAARDITEHING asub kaardisüsteemi baasis, kuid olemid PETTUS_TEHING, PETTUSE_REEGEL ja KEHTIV_REEGEL on uued tabelid, mis asuvad KMV-pettuste monitooringu andmebaasis.



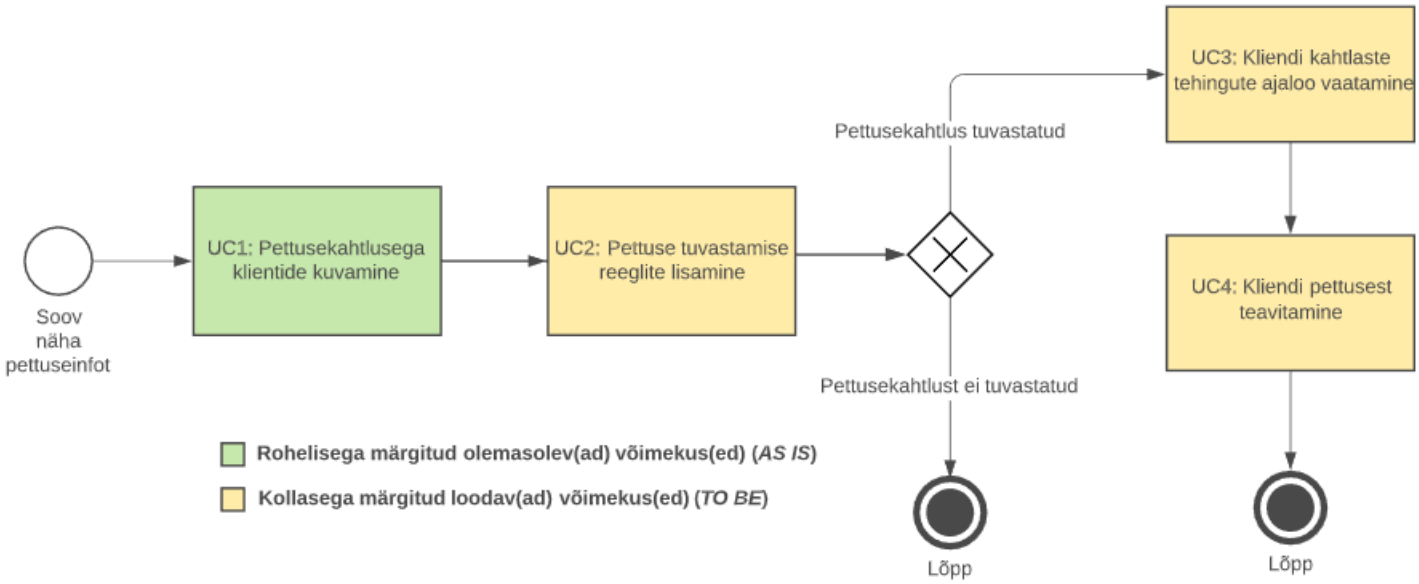
Joonis 8. KMV-pettusemonitooringu äriinfo mudel olemi-suhte diagrammina (Allikas: autori koostatud).

6.6 Lahenduse äriprotsessid

Siin peatükis on uued loodavad äriprotsessid lõppkasutaja vaatepunktist. Protsessid on kujutatud BPMN-diagrammina. Vajaduse korral on märgitud rohelisega olemasolevad (*AS IS*) võimekused ning kollasega loodavad (*TO BE*) võimekused.

6.6.1 Üldprotsess

Protsess algab kasutaja soovist vaadata pettusekahtlusega klientide loetelu ning erinevate indikaatorite seisu. Vaatamiseks logib ta rakendusse sisse ning kasutajale kuvatakse pettusekahtlusega kliendid ning erinevate indikaatorite tase. Kasutaja saab vaadata iga kaupmehe kahtlaste tehingute ajalugu ka detailvaatena. Kui kasutaja soovib, saab ta pettuse tuvastamise reegleid ja indikaatoreid lisada, muuta ja kustutada. Kui kasutaja peab pettusekahtlust piisavalt oluliseks, vajutab ta kliendi juures asuvat teavitamise nuppu ning saadab kliendile teate kahtlusest, mis asub eelsisestanud põhjal ning millele süsteem on lisanud asjakohased andmed.

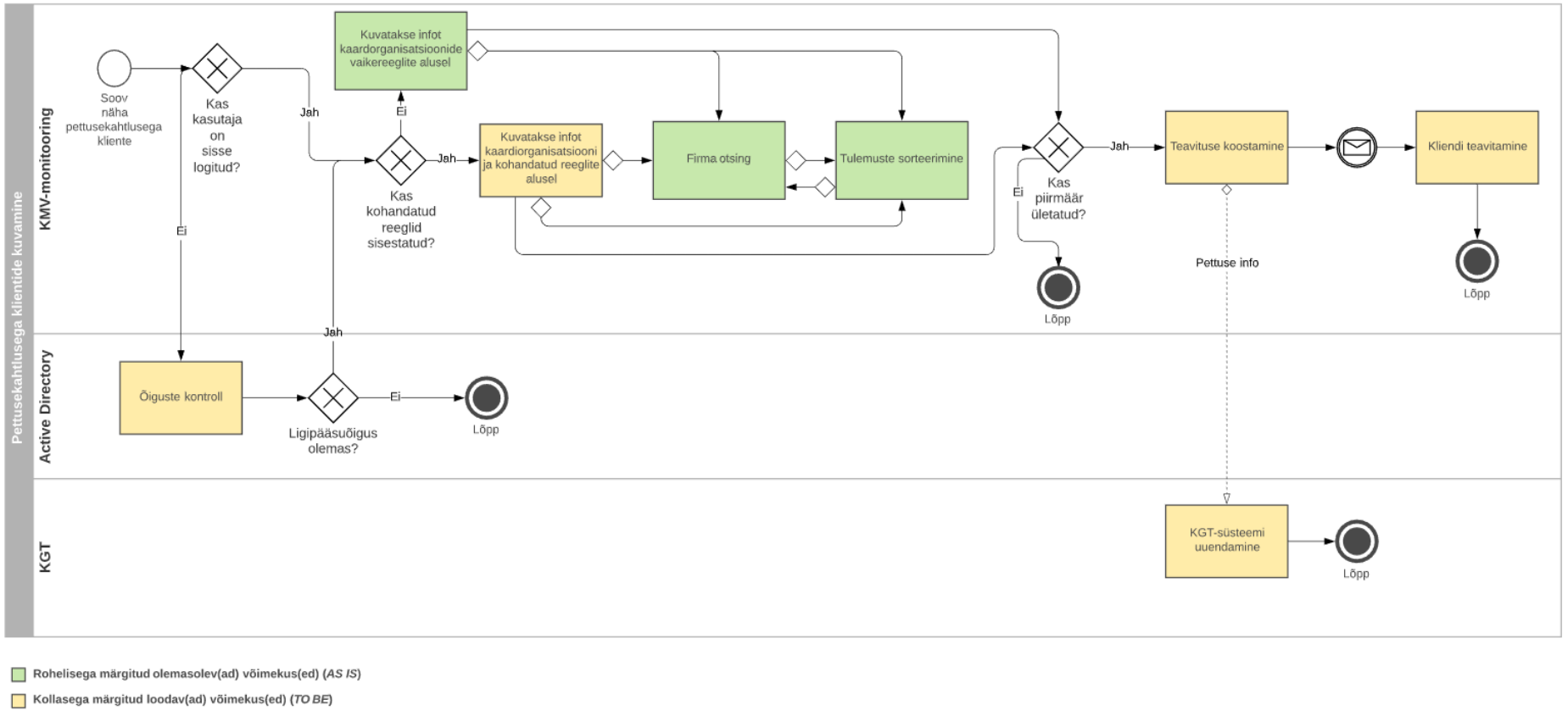


Joonis 9. KMV-monitooringu üldprotsess (Allikas: autori koostatud).

6.6.2 UC1: Pettusekahtlusega klientide kuvamine

Protsess algab kasutaja soovist näha pettusekahtlusega kliente puudutavat infot. Kasutaja läheb rakenduse pealehele ning süsteem kontrollib AD (*Active Directory*) kaudu, kas kasutajal on õigus lehele siseneda. Lehel kuvatakse info kaupmeeste kohta agregeeritud kujul – igal real eraldi kaupmees. Igal real kuvatakse alati kaupmeest puudutavad tunnused ning lisaks sellele erinevate aktiveeritud reeglite piirmäärad. Juhul, kui kaupmehele ei kohaldu kasutaja poolt loodud reegleid, kuvatakse kaupmehe juures vaid kaardiorganisatsioonide poolt määratud vaikereeglid. Tulemusi on võimalik sorteerida kõikide tunnuste alusel. Otsingut saab teostada sisestades kaupmehe ärinime või nime osa. Juhul, kui kaupmehele rakendunud reeglite piirmäär on ületatud, saab kasutaja

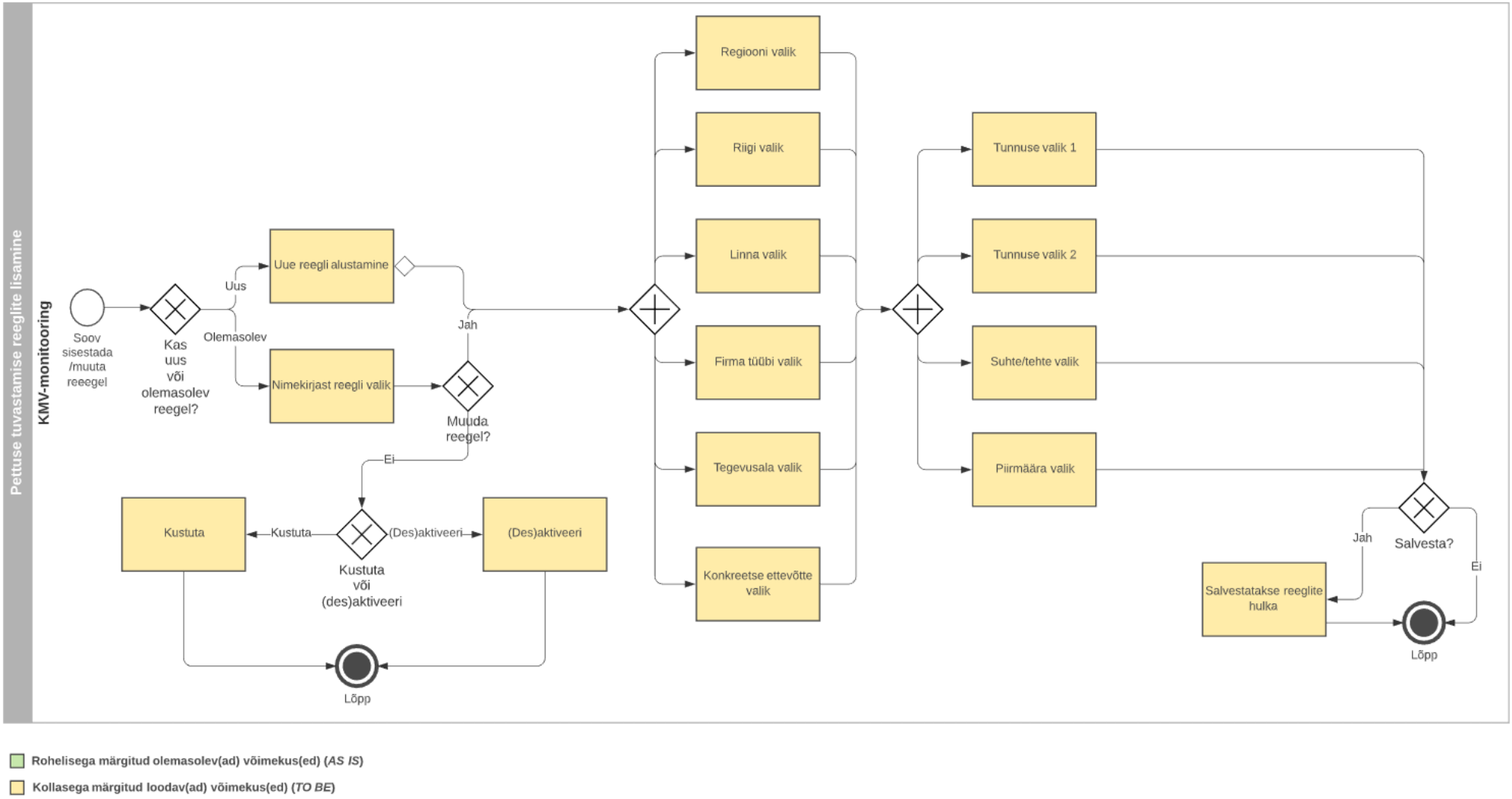
alustada teavituse koostamist, lisaks saadetakse info piirmäära ületamisest KGT-süsteemi.



Joonis 10. Pettusekahtlusega klientide kuvamine (Allikas: autori koostatud).

6.6.3 UC2: Pettuse tuvastamise reeglite lisamine

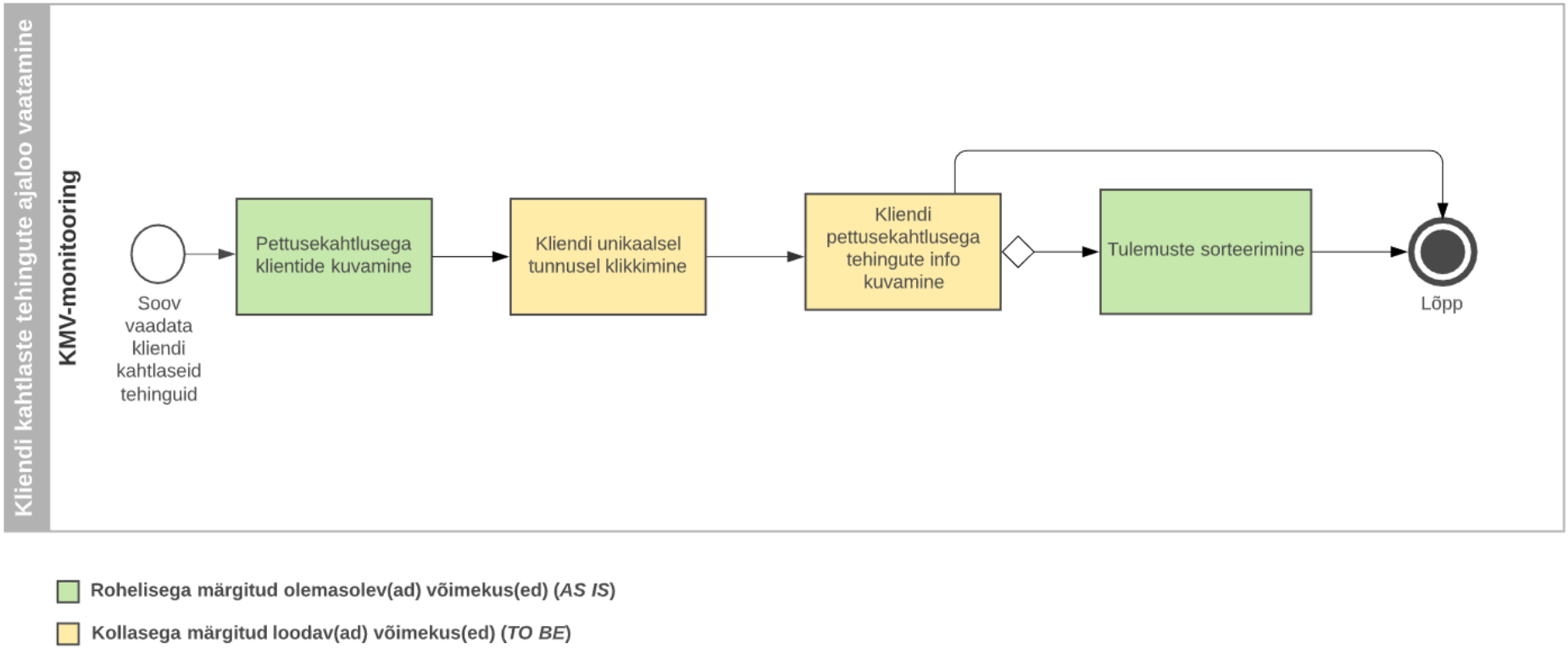
Protsess algab kasutaja soovist lisada või muuta pettuse tuvastamise reegleid. Kasutaja saab valida olemasolevate reeglite hulgast või alustada täiesti uue reegliga. Kasutaja saab määrata erinevaid reegli sihtgrupi parameetreid ehk kellele reegel kohaldub – kõik parameetrid on valikulised. Seejärel saab valida tunnused ja nende omavahelised suhted, mille alusel piirmäär arvutatakse. Näiteks tagasilükatud tehingute arv jagatud ööpäeva tehingute arvuga või tagasilükatud tehingute arv tunni jooksul. Kasutaja saab reeglit kustutada, aktiveerida ja desaktiveerida.



Joonis 11. Pettuse tuvastamise reeglite lisamine (Allikas: autori koostatud).

6.6.4 UC3: Kliendi kahtlaste tehingute ajaloo vaatamine

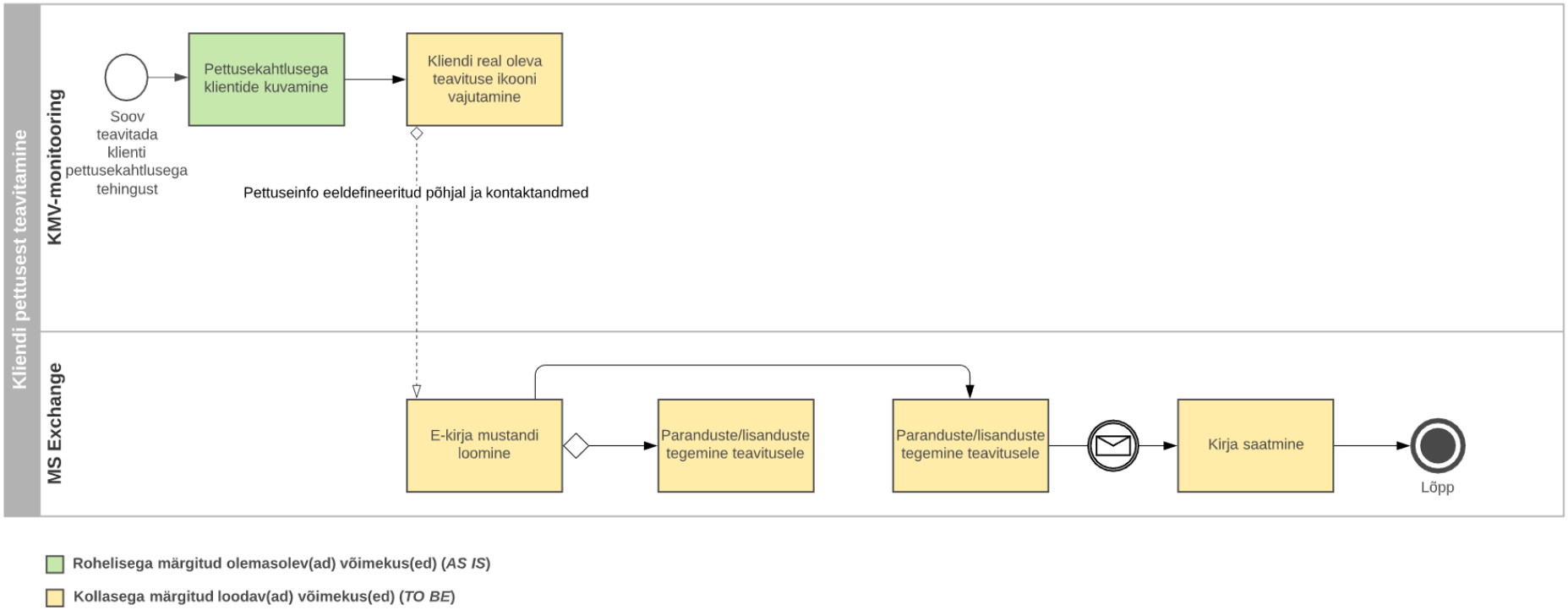
Protsess algab kasutaja soovist näha pettusekahtlusega klientide tehingute detailsemat infot. Kasutusmallis UC1 kirjeldatud vaates on kõik ettevõtet kirjeldavad näitajad agregeeritud – kui kasutaja soovib näha konkreetseid tehingud, mis on reeglite alusel filtrisse jäänud, siis ta vajutab kliendi unikaalse koodi peal ning rakendus viib ta detailsemasse tehingute vaatesse. Tulemusi saab filtreerida kõikide tunnuste alusel. Võimalik on liikuda tagasi agregeeritud vaatesse ning vaadata seejärel mõne teise kliendi detailvaadet.



Joonis 12. Kliendi kahtlaste tehingute ajaloo vaatamine (Allikas: autori koostatud).

6.6.5 UC4: Kliendi pettusest teavitamine

Protsess algab kasutaja soovist teatada kaupmeest kahtlusest, et osa tema maksepunktis tehtud tehingutest on kahtlased. Kasutusmalli alguses on kasutaja kasutusmallis UC2 kirjeldatud rakenduse avalehel. Nähes kaupmeest, kellel on üks või mitu piirmäära ületatud, vajutab kasutaja kliendi real olevat teavituse nuppu, misjärel monitooringusüsteemist laetakse eeldefineeritud teavituse põhi. See on automaatselt täidetud asjakohase infoga – need andmed edastatakse e-kirja põhjale, kus kasutajal on võimalik oma soovi järgi muudatusi teha. Kui teavitus on valmis, saadab kasutaja selle kliendile.



Joonis 13. Kliendi pettusest teavitamine (Allikas: autori koostatud).

7 Süsteemianalüüsi tulemused

Süsteemianalüüsi peatükis kirjeldatakse olulisemad kasutusmallid UML-kasutusmallide diagrammina *(Use Case Diagram)*. Seejärel kirjeldatakse funktsionaalsed nõuded kasutusmallidena.

- UC1: Pettusekahtlusega klientide kuvamine;
- UC2: Pettuse tuvastamise reeglite lisamine;
- UC3: Kliendi kahtlaste tehingute ajaloo vaatamine;
- UC4: Kliendi pettusest teavitamine.

Süsteemianalüüsi tulemuste kujutamiseks kasutatakse veel kontekstimudelit (*Context Diagram*) ja tegevusdiagrammi (*Activity Diagram*). Lisaks analüüsitakse olulisemate tehnoloogiate (konteinerid, veebiarendus, pidevvalmidus) valiku kriteeriumeid.

Süsteemianalüüsi käigus kirjeldatakse loodava rakenduse mittefunktsionaalseid nõudeid FURPS+ põhimõtetest lähtuvalt.

7.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded kirjeldavad süsteemile seatavaid ülesandeid. Ärinõuete kirjeldamisel antakse loodavast süsteemist üldisem kirjeldus, funktsionaalsed nõuded peavad olema niivõrd detailsed, et nende järgi oleks võimalik süsteemi implementeerida. Kasutusmallina saab kirjeldada funktsionaalseid nõudeid, millest on aktorile (inimene või väline süsteem) otsene kasu [21]. Et kirjeldada funktsionaalseid nõudeid võimalikult detailselt ning vältida dubleerimist, on funktsionaalsed nõuded selle töö raames esitatud kasutusmallidena.

7.1.1 Kasutusmallide diagramm

Kasutusmallide mudel kujutab infosüsteemi peamisi kasutusmalle:

Diagrammil on kujutatud viit aktorit:

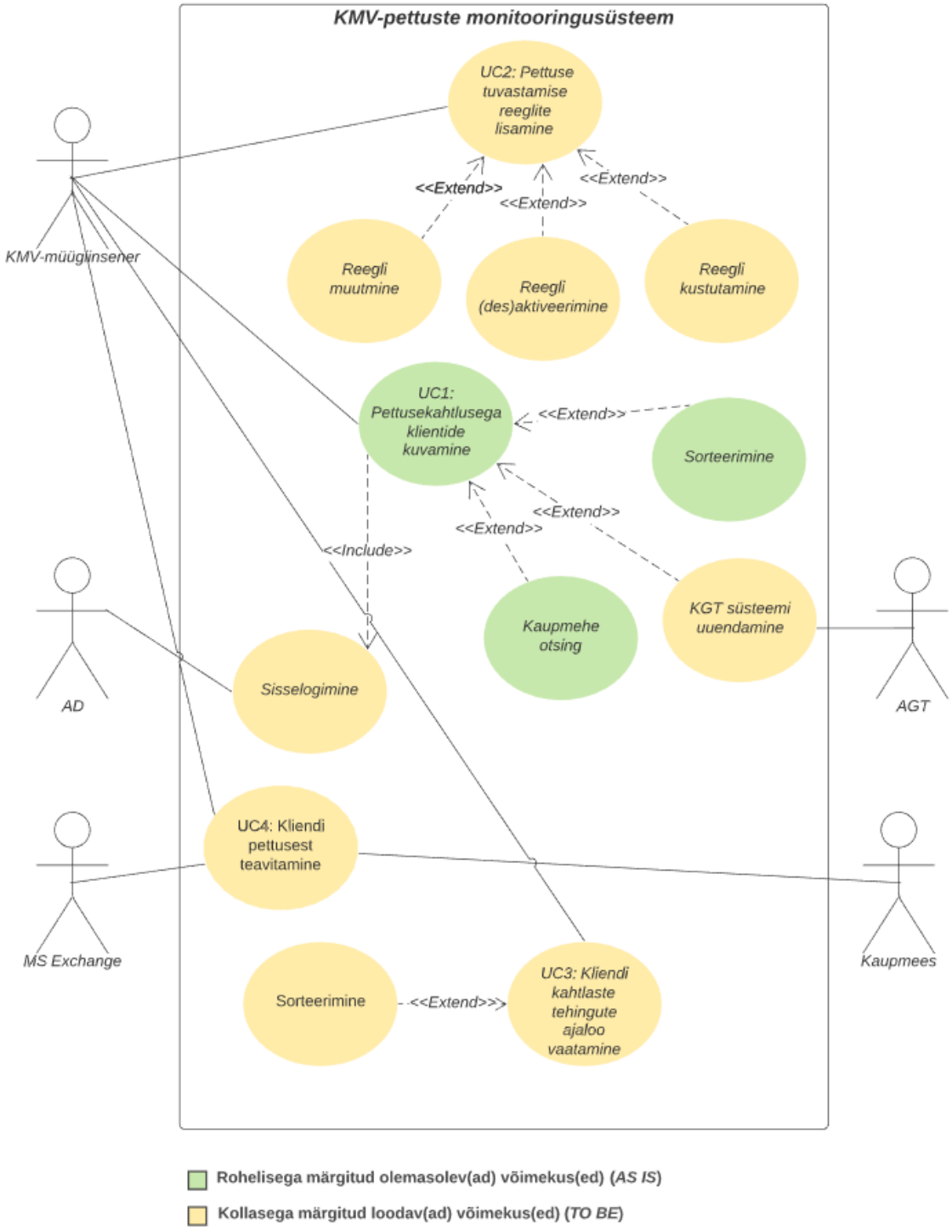
1. KMV-müügiinsener – Töötaja, kes igapäevaselt jälgib KMV-pettusemonitooringu tasemeid ning suhtleb kaupmeestega.

2. Kaupmees – Füüsiline või juriidiline isik, kes tarbib kaardimaksete vastuvõtmise teenust.

3. KGT – kaardimaksete vastuvõtmise taristu teenusepakkuja süsteem.

4. Active Directory (AD) – Microsofti domeenide ja kasutajate haldussüsteem. Käsitletava infosüsteemi puhul on see oluline ligipääsuõiguste haldamise kontekstis.

5. MS Exchange – Süsteem, mille kaudu saadetakse kaupmeestele e-kirja teel teavitusi.



Joonis 14. Kasutusmallide mudel (Allikas: autori koostatud).

7.1.2 UC1: Pettusekahtlusega klientide kuvamine

Pettusekahtlusega klientide kuvamise kasutusmallis on kirjeldatud tavatöövoog ja järgnevad alternatiivsed töövood:

- Kasutajal puudub ligipääsuõigus;

- Kaupmehele kohalduvad kasutaja poolt loodud reeglid

ID ja nimetus	**UC1: Pettusekahtlusega klientide kuvamine**
Peamine aktor	KMV-müügiinsener (kasutaja)
Teised aktorid	Active Directory, KGT-süsteem
Kirjeldus	Pettusekahtlusega kliendid kuvatakse rakenduse esilehel. Kui sisestatud reegli skooril täitub määratud limiit, värvub see lahter punaseks. Teave limiidi ületamisest saadetakse ka KGT-süsteemi.
Eeltingimused	Kasutaja (KMV-müügiinsener) on lisatud rakendusele ligipääsu võimaldavasse AD-gruppi.
Järeltingimused	1. Kasutaja saab alustada pettuseteavituse koostamist kaupmehele. 2. Kasutaja saab vaadata kaupmehe pettusekahtlusega tehingute detailvaadet.
Tavatöövoog	1. Kasutaja saabub rakenduse veebilehele. 2. Süsteem kontrollib AD kaudu, kas kasutajal on õigus lehele siseneda. 3. Kuvatakse reeglite piirmäärad kaardiorganisatsioonide poolt määratud vaikereeglite alusel. 4. Lisaks kuvatakse kliendi ettevõtte nimi, ärinimi, sisemine identifikaator, kaardiorganisatsiooni identifikaator, ettevõtte käive kuupõhiselt ning ettevõtte pettusekahtlusega käive kuupõhiselt. 5. Info piirmäära ületamisest saadetakse automaatselt KGT-süsteemi. 6. Kasutajal on võimalus konkreetset klienti otsida ettevõtte nime- või ärinimepõhiselt. 7. Kasutajal on võimalik andmeid kõikide veergude kaupa kasvavalt või kahanevalt järjestada. 8. Kasutajal on võimalik valida vaates kuvatavate ridade arvu.
Alternatiivsed töövood	**Kasutajal puudub ligipääsuõigus** 1. Kasutaja saabub rakenduse veebilehele. 2. Süsteem kontrollib AD kaudu, kas kasutajal on õigus lehele siseneda. 3. Kuvatakse teavitus ligipääsuõiguste puudumisest. **Kaupmehele kohalduvad kasutaja poolt loodud reeglid** 1. Kasutaja saabub rakenduse veebilehele.

	2. Süsteem kontrollib AD kaudu, kas kasutajal on õigus lehele siseneda. 3. Kuvatakse reeglite piirmäärad kaardiorganisatsioonide poolt määratud vaikereeglite ning kasutaja poolt defineeritud reeglite alusel. 4. Lisaks kuvatakse kliendi ettevõtte nimi, ärinimi, sisemine identifikaator, kaardiorganisatsiooni identifikaator, ettevõtte käive kuupõhiselt ning ettevõtte pettusekahtlusega käive kuupõhiselt. 5. Info piirmäära ületamisest saadetakse automaatselt KGT-süsteemi. 6. Kasutajal on võimalus konkreetset klienti otsida ettevõtte nime-või ärinimepõhiselt. 7. Kasutajal on võimalik andmeid kõikide veergude kaupa kasvavalt või kahanevalt järjestada. 8. Kasutajal on võimalik valida vaates kuvatavate ridade arvu.
Kasutussagedus	Pettusekahtlusega klientide kuvamine toimub alati enne kliendile pettusekahtluse teavituse saatmist.
Ärireeglid	1. Kaupmeeste koondinfo sorteeritakse vaikimisi kuupäeva järgi kahanevalt. 2. Koondinfo read on vaikimisi agregeeritud kuu ja kaupmehe kaupa. 3. Agregeerimise perioodi (päev, nädal, kuu, aasta) saab muuta reeglite vaate alt.

Tabel 6. Kasutusmall – UC1: pettusekahtlusega klientide kuvamine (Allikas: autori koostatud).

7.1.3 UC2: Pettuse tuvastamise reeglite lisamine

Pettuse tuvastamise reeglite lisamise kasutusmallis on kirjeldatud tavatöövoog ja järgnevad alternatiivsed töövood:

- Olemasoleva reegli muutmine;
- Olemasoleva reegli kustutamine;
- Olemasoleva reegli aktiveerimine/desaktiveerimine

ID ja nimetus	**UC2: Pettuse tuvastamise reeglite lisamine**
Peamine aktor	KMV-müügiinsener (kasutaja)
Teised aktorid	Puuduvad
Kirjeldus	Kasutajal on võimalik sisestada lisaks vaikereeglitele ka enda loodud pettuse tuvastamise reegleid. Kõigepealt saab kasutaja valida uue reegli loomise või vana

	muutmise vahel. Seejärel on võimalik valida tunnuseid, mille alusel reegel kaupmeestele kohaldub. Viimasena on võimalik määrata reegli aluseks olevaid tunnuseid ning omavavahelisi suhteid, samuti reeglite kuvamise ühikuid või piirmäärasid.
Eeltingimused	Kasutajal on ligipääs rakenduse esilehele.
Järeltingimused	Kasutaja näeb pettusekahtlusega klientide kuvamisel lisaks vaikereeglitele ka enda koostatud reeglite tasemeid.
Tavatöövoog	1. Kasutaja on rakenduse esilehel. 2. Kasutaja vajutab lingile „reeglid“. 3. Süsteem kuvab kasutajale olemasolevad reeglid. 4. Kasutaja vajutab nupule „loo uus reegel“. 5. Süsteem kuvab reegli kohaldumise kriteeriumide valikud: regioon, riik, linn, firma tüüp, tegevusala, ettevõtte registrikood. 6. Kasutaja valib soovitud kohaldumise kriteeriumid. Kriteeriumidest kohustuslik on valida vaid üks. 7. Kasutajale kuvatakse nimekiri kaupmeestest, kes nendele kriteeriumidele vastab. 8. Kasutaja kinnitab valiku vajutades nupule „kinnita“. 9. Kasutajale kuvatakse tunnuste valik, tunnuste suhete valik (diskreetne, jagatis, liitmine, lahutamine), alamreeglite omavahelised tingimused (*if*, *or*, *and*, *else*), piirmäära andmetüüp (protsent, diskreetne, pidev, predikaat), agregeerimise periood (päev, nädal, kuu) ja piirmära tase. 10. Kasutaja valib soovitud väärtused. Tunnuste valikutest on kohustuslik valida vaid üks. 11. Kasutaja kinnitab valiku vajutades nupule „kinnita“. 12. Süsteem salvestab reegli.
Alternatiivsed töövood	**Olemasoleva reegli muutmine** 1. Kasutaja on rakenduse esilehel. 2. Kasutaja vajutab lingile „reeglid“. 3. Süsteem kuvab kasutajale olemasolevad reeglid. 4. Kasutaja vajutab nimekirjas topeltklõpsu soovitud reeglile. 5. Süsteem kuvab olemasoleva reegli kohaldumise kriteeriumid. 6. Kasutaja muudab soovitud kohaldumise kriteeriumid. 7. Kasutajale kuvatakse nimekiri kaupmeestest, kes nendele kriteeriumidele vastab. 8. Kasutaja kinnitab valiku vajutades nupule „kinnita“.

	9. Kasutajale kuvatakse olemasoleva reegli tunnused, tunnuste suhted, alamreeglite omavahelised tingimused, piirmäära andmetüüp, agregeerimise periood ja piirmära tase. 10. Kasutaja muudab soovitud väärtused. 11. Kasutaja kinnitab valiku vajutades nupule „kinnita“. 12. Süsteem salvestab reegli muudatused. **Olemasoleva reegli kustutamine** 1. Kasutaja on rakenduse esilehel. 2. Kasutaja vajutab nupule „reeglid“. 3. Süsteem kuvab kasutajale olemasolevad reeglid. 4. Kasutaja vajutab soovitud reeglile. 5. Kasutaja vajutab nupule „kustuta“. 6. Süsteem küsib: „Oled kindel?“ 7. Kasutaja vajutab nupule „kinnita“. 8. Süsteem kustutab reegli. **Olemasoleva reegli aktiveerimine/desaktiveerimine** 1. Kasutaja on rakenduse esilehel. 2. Kasutaja vajutab nupule „reeglid“. 3. Süsteem kuvab kasutajale olemasolevad reeglid. 4. Kasutaja vajutab soovitud reeglile. 5. Kasutaja vajutab nupule „aktiveeri/desaktiveeri“. 6. Süsteem küsib: „Oled kindel, et soovid aktiveerida/desaktiveerida?“ 7. Kasutaja vajutab nupule „kinnita“. 8. Süsteem aktiveerib/desaktiveerib reegli.
Kasutussagedus	Kasutussagedus sõltub kasutaja soovist.
Ärireeglid	1. Iga „kinnita“ sammu juures on võimalik ka muudatused tühistada. 2. Alamreeglite hulk ei ole piiratud, nende omavahelised suhted on määratud tingimuslausetega. 3. Ka vaikereeglid kuvatakse kehtivate reeglite nimistus. 4. Vaikereegleid ei saa kustutada. 5. Vaikereegleid ei saa desaktiveerida.

Tabel 7. Kasutusmall – UC2: pettuse tuvastamise reeglite lisamine (Allikas: autori koostatud).

7.1.4 UC3: Kliendi kahtlaste tehingute ajaloo vaatamine

Kliendi kahtlaste tehingute ajaloo vaatamise kasutusmallis on kirjeldatud tavatöövoog.

ID ja nimetus	UC3: Kliendi kahtlaste tehingute ajaloo vaatamine
Peamine aktor	KMV-müügiinsener (kasutaja)
Teised aktorid	Puuduvad
Kirjeldus	Kui kasutaja soovib näha konkreetseid tehingud, mis on reeglite alusel filtrisse jäänud, siis ta vajutab kaupmehe unikaalse koodi peal ning rakendus viib ta detailsemasse tehingute vaatesse. Tulemusi saab filtreerida kõikide tunnuste alusel.
Eeltingimused	Kaupmehel on vähemalt üks pettusekahtlusega tehing. Kasutajal on ligipääs rakenduse esilehele.
Järeltingimused	Puuduvad
Tavatöövoog	1. Kasutaja on rakenduse esilehel. 2. Kasutaja vajutab kaupmehe unikaalsele tunnusele. 3. Süsteem kuvab kliendi kahtlaste tehingute ajaloo. 4. Kasutajal on võimalik andmeid kõikide veergude kaupa kasvavalt või kahanevalt järjestada. 5. Kasutajal on võimalik valida vaates kuvatavate ridade arvu.
Alternatiivsed töövood	Puuduvad
Kasutussagedus	Kasutussagedus sõltub kasutaja soovist.
Ärireeglid	Kaupmeeste pettusekahtlusega tehingud sorteeritakse vaikimisi kuupäeva järgi kahanevalt.

Tabel 8. Kasutusmall – UC3: kliendi kahtlaste tehingute ajaloo vaatamine (Allikas: autori koostatud).

7.1.5 UC4: Kliendi pettusest teavitamine

Kliendi pettusest teavitamise kasutusmallis on kirjeldatud tavatöövoog.

ID ja nimetus	UC4: Kliendi pettusest teavitamine
Peamine aktor	KMV-müügiinsener (kasutaja)
Teised aktorid	MS Exchange, Kaupmees
Kirjeldus	Nähes kaupmeest, kellel on üks või mitu piirmäära ületatud, vajutab kasutaja kliendi real olevat teavituse nuppu, misjärel monitooringusüsteemist laetakse eeldefineeritud teavituse põhi, mis on automaatselt täidetud asjakohase infoga – need andmed edastatakse e-kirja põhjale, kus kasutajal on võimalik oma soovi järgi muudatusi teha. Kui teavitus on valmis, saadab kasutaja selle kliendile.
Eeltingimused	Kaupmehel on üks või mitu reegli piirmäära ületatud. Kasutajal on ligipääs rakenduse esilehele.
Järeltingimused	Puuduvad
Tavatöövoog	1. Kasutaja on rakenduse esilehel. 2. Kasutaja vajutab kaupmehe ärinime juures asuvale teavituse nupule. 3. Süsteem kuvab e-kirja põhjal eeldefineeritud teavituse põhja, mida on täiendatud asjakohase infoga süsteemi andmebaasist. 4. Kasutajal on võimalik e-kirjas muudatusi teha. 5. Kasutaja saadab e-kirja kaupmehele.
Alternatiivsed töövood	Puuduvad
Kasutussagedus	Kasutussagedus sõltub kasutaja soovist.
Ärireeglid	1. E-kirja eeldefineeritud põhi on kodeeritud süsteemi koodi ning seda ei saa muuta kasutajaliidese kaudu. 2. Süsteem otsib andmebassist automaatselt kaupmehe esindaja emaili aadressi ning lisab selle saaja väljale.

Tabel 9. Kasutusmall – UC4: kliendi pettusest teavitamine (Allikas: autori koostatud).

7.2 Mittefunktsionaalsed nõuded

Süsteemi omadustena käsitletavad mittefunktsionaalsed nõuded on sama kriitilised kui funktsionaalsed nõuded, tagades kogu süsteemi kasutatavuse ja tõhususe. Nende täitmata jätmise korral võib süsteem mitte vastata tellija, turu või regulatiivsetele nõuetele [6].

Mittefunktsionaalsete nõuete kaardistamine on oluline, et täita tellijate nõudmised süsteemi kvaliteedi osas.

Käesoleva töö käigus kategoriseeritakse mittefunktsionaalsed vastavalt FURPS põhimõtetele järgnevalt:

- Kasutatavus *(Usability)*
- Töökindlus *(Reliability)*
- Jõudlus *(Performance)*
- Toetatavus *(Supportability)*

7.2.1 Kasutatavus

Kasutatavuse punkti alla on koondatud peamiselt kasutajaliidese ja kasutajakogemusega seotud nõuded.

- KAMFN1 – kasutajaliideses kasutatav stiil, kirja font ja värvid peavad olema läbivalt ühesugused.
- KAMFN2 – andmete kustutamisel tuleb alati küsida kasutaja kinnitust.
- KAMFN3 – andmete sisestamisel tuleb kohustuslikud andmeväljad tähistada punase raamiga.
- KAMFN4 – süsteem peab kuvama teateid andmete salvestamise, kustutamise ja vigade korral.
- KAMFN5 – kõiki kasutusmalle peab olema võimalik alustada rakenduse esilehelt.
- KAMFN6 – lõppkasutajal ei tohi olla kasutuselevõtu testi (*User Acceptance Testing*) käigus kasutajakogemusele etteheiteid.
- KAMFN7 – andmete laadimine süsteemi peab toimuma automaatselt.

7.2.2 Töökindlus

Töökindluse punkti alla on koondatud nõuded, mis on seotud rakenduse kättesaadavuse ja taastamisega.

- TÖMFN1 – RTO (*Recovery Time Objective*) ehk maksimaalne süsteemi taastamisele kuluv aeg. E–R: 3h. L–P ja riigipühad: 24h.
- TÖMFN2 – RPO (*Recovery Point Objective*) ehk maksimaalne lubatav andmekadu. Andmete varundamine toimub kord ööpäevas ehk 24h.

RTO ja RPO tuleb kokku leppida IT-haldusega ning defineerida rollid ja ülesanded süsteemi taastamisel.

7.2.3 Jõudlus

Jõudluse punktis on kirjeldatud nõuded, mis on seotud rakenduse töökiiruse ja -võimekusega.

- JÕMFN1 – koondinfo (UC1) laadimisel maksimaalne lubatud kestus on 2 sekundit.
- JÕMFN1 – ettevõtte tehingute (UC3) laadimisel maksimaalne lubatud kestus on 1 sekund.

7.2.4 Toetatavus

Toetatavuse peatükis on kirjeldatud nõuded, mis on seotud rakenduse ja muudatuste haldamisega.

- TOMFN1 – rakenduse kohta peab olema dokumentatsioon, mis sisaldab minimaalselt kontekstimudelit, evitusdiagrammi, komponentdiagrammi, RPO- ja RTO-aega ning -protsessi, GET- ja POST-päringute logimise protsessi ning ligipääsude võimaldamise protsessi.
- TOMFN2 – iga muudatuse puhul süsteemis, peab eelnevalt uuendama rakenduse dokumentatsiooni kui see on vajalik.
- TOMFN3 – keskmine muutuste läbiviimise aeg (*Mean Time to Change – MTTC)* peab olema vähem kui 5 tööpäeva.

7.2.5 Turvalisus

Turvalisuse peatükis on kirjeldatud nõudeid, mis on seotud rakenduse infoturbe aspektidega. Turvalisus ei ole FURPSi järgi eraldi kategooria, kuid selle valdkonna nõuete olulisuse ja arvukuse tõttu on otstarbekam koondada need eraldi peatüki alla.

- TUMFN1 – pikk sessioon ei ole lubatud.
- TUMFN2 – mitteaktiivse kasutaja sessioon lõpeb automaatselt 30 minuti jooksul.
- TUMFN3 – kliendi ja serveri suhtlus käib üle HTTPS protokolli.
- TUMFN4 – rakendus tohib olla ligipääsetav vaid ettevõtte sisevõrgust.

- TUMFN5 – kasutajate õiguste haldus baseerub AD-rollidel ja -gruppidel.
- TUMFN6 – kasutaja autentimine toimub ühekordse sisselogimisega ehk SSO (*Single sign-on*) põhimõttel.
- TUMFN7 – kasutaja autentimine toimub veebiserveris kasutades Kerberose protokolli ja Django-auth-ldap teeki.

7.2.6 FURPS+

FURPS+ peatükis on kirjeldatud infosüsteemi kavandamise, arendamise ja juurutamisega seotud nõuded.

- PLMFN1 – rakendus peab kasutama MS SQL Serveri andmebaasi.
- PLMFN2 – rakendus peab olema arendatud Django raamistikus.
- PLMFN3 – rakenduse juurutamisel peab kasutama pidevintegratsiooni ja pidevvalmiduse põhimõtteid.
- PLMFN4 – automaattestid peavad katma vähemalt rakenduse peamised kasutusmallid.
- PLMFN5 – automaattestid peavad põhinema Pytest teegil.
- PLMFN6 – rakendust peab arendama ja evitama Dockeri konteineris, et vältida ühildumisprobleeme toodanguserveriga.
- PLMFN7 – Dockerit käivitav toodanguserver peab kasutama Linuxi CentOS7 distributsiooni.
- PLMFN8 – süsteemi veebiserver peab põhinema Ngnix versioonil 1.18.
- PLMFN9 – süsteemi rakendusserver (WSGI ehk *Web Server Gateway Interface*) peab baseeruma Gunicorni versioonil 20.
- PLMFN10 – rakenduse töö jaoks vajalik kolmanda osapoole teekide haldus peab põhinema Pipenv teegil.
- PLMFN11 – rakenduse juurutamine peab järgima pidevintegratsiooni ja pidevvalmiduse praktikaid.
- PLMFN12 – 5 aasta keskmine kulu aastas peab jääma alla 15 000 EUR (arendus ja haldus).

7.3 Tehnoloogiate ja raamistike valik ja analüüs

Süsteemianalüüsi peatükis on oluline osa rakenduse arendamiseks ja juurutamiseks kasutavate tehnoloogiate ja raamistike analüüsil. Tehnoloogiate valikul peab silmas pidama nii arendajata oskuseid ja eelistusi, kui ka ettevõttes juba kasutusel olevaid tehnoloogiaid ja ettevõtte arhitektuuri.

7.3.1 Veebiarenduse raamistik

Töövahendite arenduse meeskond kasutab peamise programmeerimiskeelena Pythonit ja sellel põhinevaid raamistikke. See tähendab, et loodava lahenduse jaoks oli valida peamiselt Flaski- ja Django- kui peamiste Pythoni-põhiste veebiarenduse raamistike vahel.

Järgnevalt on välja toodud ülevaade Django ja Flaski erinevused: [22]

- Vormid – Djangole on vormide põhjad sisse ehitatud ning nad ühenduvad lihtsalt ORMi ja Django administreerimise funktsionaalsusega. Flaskil sisseehitaud vormide tugi puudub.

- Andmebaas – Djangole on sisse ehitatud ORM, selle peamine eesmärk on edastada andmeid relatsioonilise andmebaasi ja rakendusmudeli vahel. ORM automatiseerib andmete edastamise selliselt, et arendaja ei pea ideaaljuhul kirjutama SQLi päringuid. Flaskiga ei tule kaasa sisseehitatud andmebaasiga suhtlemise lahendusi, kuid on olemas kolmanda osapoole tööriistu nagu SQLAlchemy, mis pakuvad ORMile sarnast funktsionaalsust.

- Õigused ja ligipääsud – Djangoga on kaasas autentimisrakendus, millega on võimalik standardselt lahendada õiguste ja ligipääsude halduse. Flask pakub turvalisi küpsiseid, et kasutajal oleks võimalik implementeerida enda lahendus.

- Administreerimine – Djangoga on kaasas rakenduste andmete haldamiseks integreeritud administreerimise liides. Flask sellist liidest ei sisalda, kuid näiteks Flask-Admin kolmanda osapoole rakendusega saab luua sarnase administreerimise tööriista.

Eeltoodu näitel saab öelda, et Django sobib kõige paremini relatsioonilisi andmebaase kasutavate veebirakenduste jaoks. Flask on kasulik spetsiifilisemate juhtumite jaoks, mille puhul pole kasu Django tugevast põimitusest relatsiooniliste baasidega. Django kasutajate kogukond on aktiivsem. Flaski kasutajad on tihti sunnitud algusest arendama funktsionaalsusi, mis Djangoga juba kaasas. Mõlemad raamistikud paistavad silma efektiivse prototüüpimise vahendina [23].

Djangoga on andmepäringute (vaadete) loomiseks kolm põhimõttelist valikut: kasutada klassipõhiseid vaateid, kasutada klassipõhiseid geneerilisi vaateid (sisseehitatud standardvaated) või kasutada funktsioonidel baseeruvaid vaateid. Roy Greenfield *et al* [24] järgi on klassipõhised vaated küll kompaktsemad, kuid funktsioonipõhised vaated annavad andmete liikumisest parema ülevaate ning on neile arendajatele, kes Djangot igapäevaselt ei kasuta, paremini loetavad.

Kuna magistritöö tulemusel valmiva analüüsi alusel loodav monitooringulahendus on küllaltki standardne CRUD (*Create, Read, Update and Delete*) rakendus, siis on autor ülaltoodud infole tuginedes valinud rakenduse arendamiseks Django raamistiku, mis pakub oma sisseehitatud funktsionaalsuste abil kiiremat arendust ja selgepiirilisemat rakenduse struktuuri.

7.3.2 Konteinersüsteem

Rakenduse evitamisel toodanguserverisse on võimalik läheneda traditsioonilisel viisil ehk laadida lähtekood repositooriumist serverisse ning paigaldada toodanguserverisse rakenduse tööks vajalik tarkvara ja teegid. Tavaliselt tekib selle käigus palju ühilduvuse probleeme, eriti kui arenduseks kasutatakse Windowsi süsteemi ning toodanguserverina Linuxi süsteemi – mõni teek töötab vaid Linuxil ning suur osa tarkvarast käitub erinevatel operatsioonisüsteemidel või isegi sama operatsioonisüsteemi erinevatel versioonidel isemoodi. See tähendab, et kõikidel arenduse, toodangueelse, toodangu- ja testmasinatel peab olema sama konfiguratsioon – selle tagamine on keeruline ja ajamahukas.

Et ülaltoodud probleeme vältida ning muuta arendusprotsessi sujuvamaks, on mõistlik süsteeme arendada ja evitada konteinerites. Konteinerite eesmärk on hõlbustada isoleeritud keskkonna abil rakenduste juurutamist ja käitamist. Konteinerid võimaldavad arendajal pakkida rakenduse koos kõigi vajalike osadega, näiteks operatsioonisüsteemi komponentide, vajaliku taristu tarkvara (näiteks Python, Nginx, Gunicorn) ja teekidega,

ning tarnida see kõik ühe pakendina. See tähendab, et rakendus töötab sõltumata konfiguratsioonist igas masinas samamoodi [25].

Konteinerite asemel on rakenduste käitamiseks traditsiooniliselt kasutatud ühe füüsilise serveri peal töötavaid virtuaalmasinaid. Alloleval joonisel on näha nende kahe tehnoloogia võrdlust.

Virtuaalmasin	Dockeri konteiner
Riistvara tasemel protsesside isoleerimine.	Operatsioonisüsteemi taseme protsesside isoleerimine.
Käivitub minutitega	Käivitub sekunditega
Suurus on mõõdetav gigabaitides	Konteinerite suurus on mõõdetav pigem kilobaitides või megabaitides
Valmis virtuaalmasina konfiguratsiooni on keeruline leida	Eelnevalt loodud Dockeri konteineri põhjad on hõlpsalt kättesaadavad (Dockerhub)
Virtuaalmasina loomine võtab suhteliselt kaua aega	Konteinereid saab luua loetud sekunditega
Suurem ressursikasutus	Madalam ressursikasutus

Tabel 10. Virtuaalmasina ja Dockeri konteineri võrdlus [26].

Käesoleva töö raames analüüsitava projekti juures kasutatakse mõlemat tehnoloogiat ehk käivitatakse Dockeri konteinerites olev rakendus virtuaalmasina peal. Alloleval joonisel on lihtsustatult kujutatud rakenduse konteineriseerimist. Ruudud on konteinerid, ringid tähistatavad silda ehk ühendust konteinerite vahel, andmebaasi server on eraldiseisev ega asu konteineris. Kuna rakendus on küllaltki lihtsa arhitektuuriga, kasutatakse konteinerite suhtluse koordineerimiseks Docker-compose rakendust – Kubernetes ega Swarm ei ole

konteinerite väikest arvu ja rakenduse suhteliselt lihtsat arhitektuuri arvestades autori hinnangul vajalik.

Joonis 15. Konteinersüsteemi skeem (Allikas: autori koostatud).

7.3.3 Pidevintegratsioon ja pidevvalmidus

Selleks, et koodi muudatused jõuaksid arendaja arvutist repositooriumisse ning sealt testi- ja toodanguserveritesse, on võimalik valida mitme mooduse vahel. Üks võimalus on klassikaline juurutamise meetod, kus arendaja saadab koodi repositooriumisse ning IT-haldus laeb muudatused käsitsi toodangumasinasse. See tähendab, et reliisivälbad on pikad ja muudatuste läbiviimine aeglane ja kohmakas. Et koodi reliisimine toimuks automaatsemalt ja sujuvamalt on võimalik järgida pidevintegratsiooni ja pidevvalmiduse ehk edaspidi CI/CD (*Continuous Integration and Continuous delivery*) põhimõtteid. Järgnevalt on lahti kirjutatud CI/CD praktikad ja nende järgimise eelised.

Pidevintegratsioon [27]

Pidevintegratsiooni praktiseerivad arendajad ühendavad oma muudatused nii sageli kui võimalik peaharu külge. Arendaja tehtud muudatused läbivad igal laadimisel komponendi- ja integratsioonitestid. Sellega välditakse integratsioonikonflikte, mis on lihtsad tekkima, kui integratsiooniga oodatakse reliisipäevani. Pidev integreerimine paneb suurt rõhku automaatestimisele, et kontrollida rakenduse toimimist uute harude integreerimisel peaharru.

Pidevvalmidus [27]

Pidevvalmidus on pidevintegratsiooni laiendus, mis tagab, et muudatused on võimalik kiirelt kliendini toimetada. See tähendab, et lisaks testimise automatiseerimisele on automatiseeritud ka reliisiprotsess ja nupuvajutusega on võimalik rakendus juurutada.

Pidevvalmiduse kasutamise korral on võimalik reliisida iga päev, iga nädal, iga kahe nädala tagant või muul äri vajadustele vastaval perioodil. Pidevvalmiduse eeliseid ilmnevad kõige paremini, kui reliisida võimalikult tihti ja väikestes osades. Sel juhul on probleemide korral koodi lihtsam muuta ja parandada.

Pidevjuurutamine [27]

Pidevjuurutamine läheb sammu võrra kaugemale kui pidevvalmidus. Selle praktika kohaselt reliisitakse klientidele kõik muudatused, mis on läbinud kõik tarnevoo (*Deployment Pipeline*) etapid. Inimsekkumist ei toimu ja vaid ebaõnnestunud testid saavad takistada uue muudatuse reliisi. Pidevjuurutamise puhul saadakse kliendilt uuenduste kohta väga kiiresti tagasisidet, mis muudab arendustsükli lühemaks ja paindlikumaks. Arendajad saavad keskenduda tarkvara ehitamisele ja saavad näha töö tulemust kohe peale koodi üle laadimist.

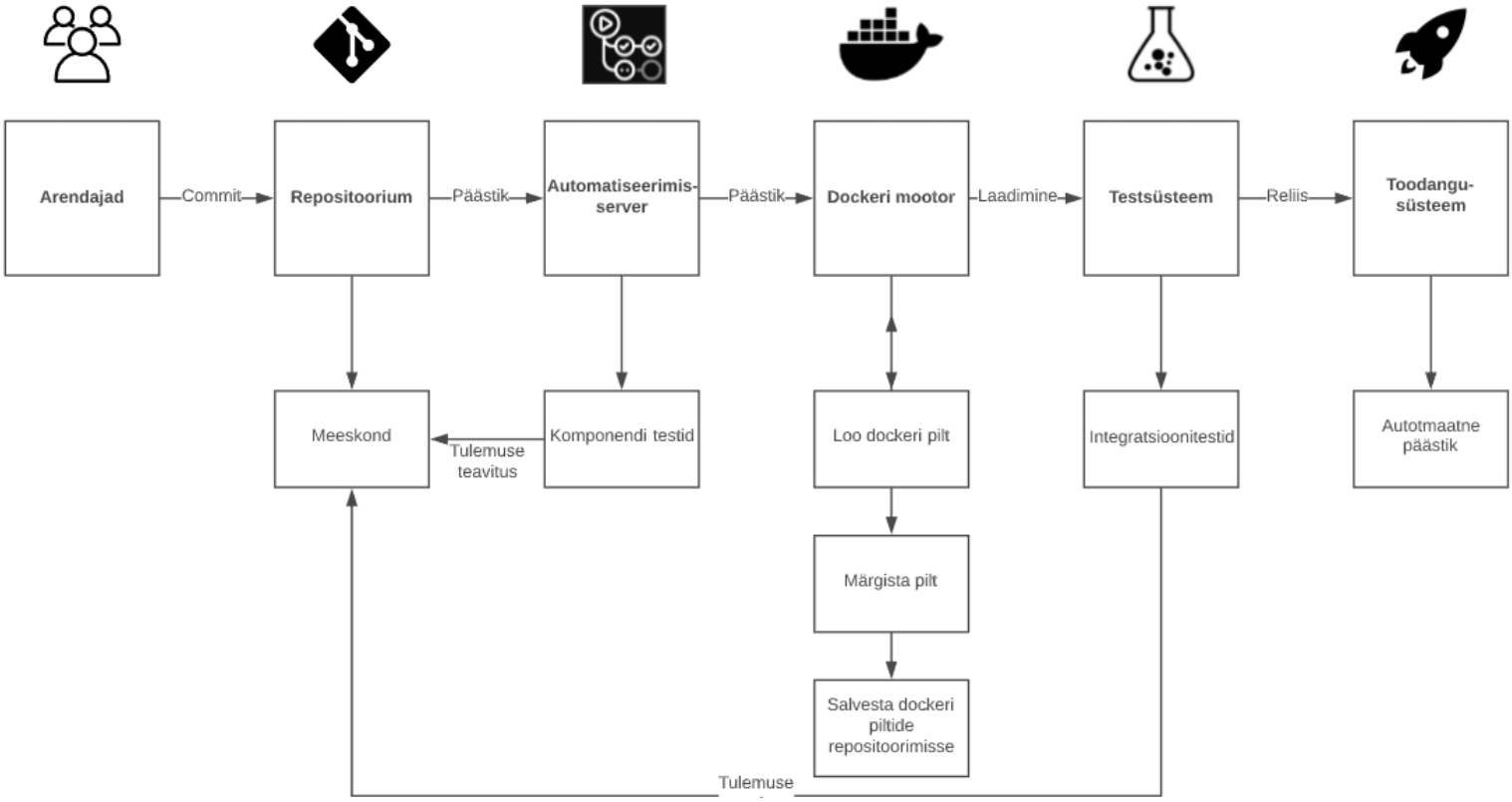
Docker ja CI/CD [25]

Docker on oluline CI/CD praktikate tagamiseks. Docker võimaldab seadistada arenduskeskkonda identselt toodanguserveriga. Lisaks saab käitada mitut arenduskeskkonda samal serveril – need võivad olla erineva tarkvara, operatsioonisüsteemide ja konfiguratsioonidega. Dockerit kasutades on võimalik kõigil arendajatel sõltumata lokaalsest keskkonnast töötada sama projekti kallal täpselt samade sätetega. See võimaldab arendajatel käivitada CI/CD jaoks olulisi automaatteste, et tuvastada koodi muudatustes potentsiaalseid vigu.

Tarnevoo skeem

Alloleval joonisel on kujutatud käesoleva analüüsi tulemusel valmiva süsteemi loomisel kasutatav tarnevoog. Tarnevoog on automatiseeritud – arendaja saadab (*Commit*) koodi repositooriumisse, automatiseerimisserver käivitab komponendi testid, Dockeri mootor loob Dockeri pildid ja salvestab need, seejärel teostatakse integratsioonitestid

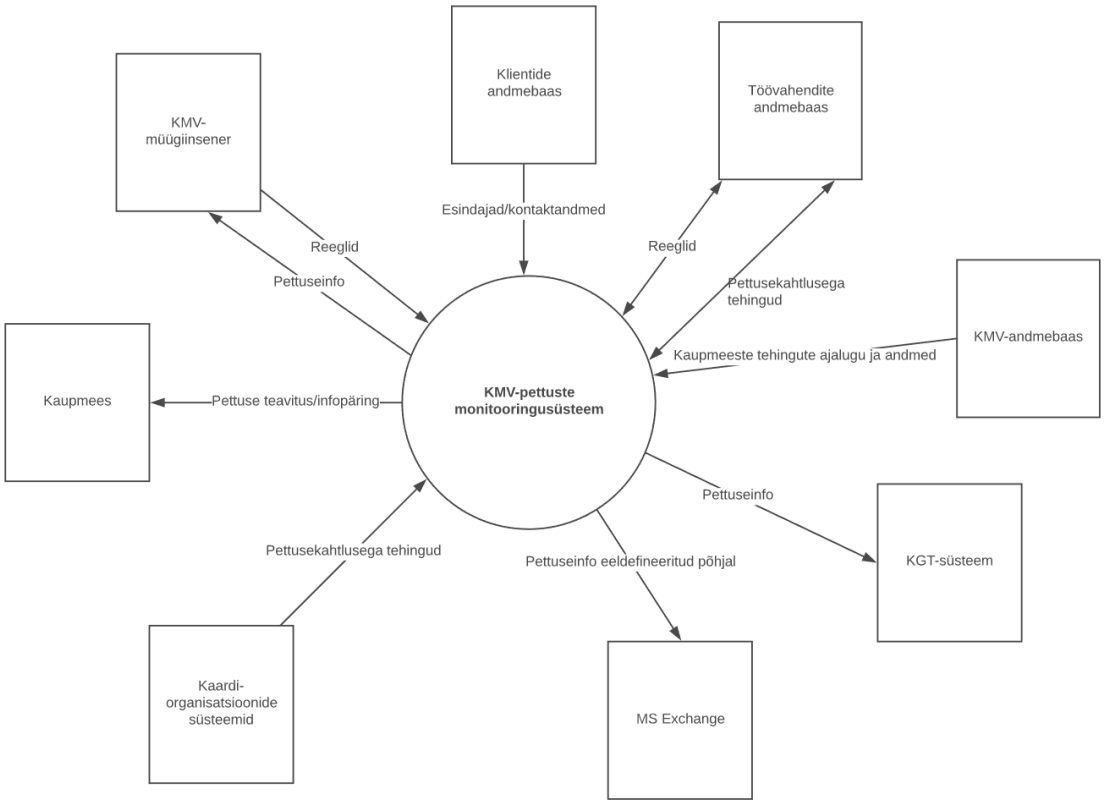
testsüsteemis ning kui kõik etapid on läbitud edukalt, reliisitakse kood automaatselt toodangukeskkonda.



Joonis 16. Tarnevoo skeem (Allikas: autori koostatud).

7.4 Kontekstimudel

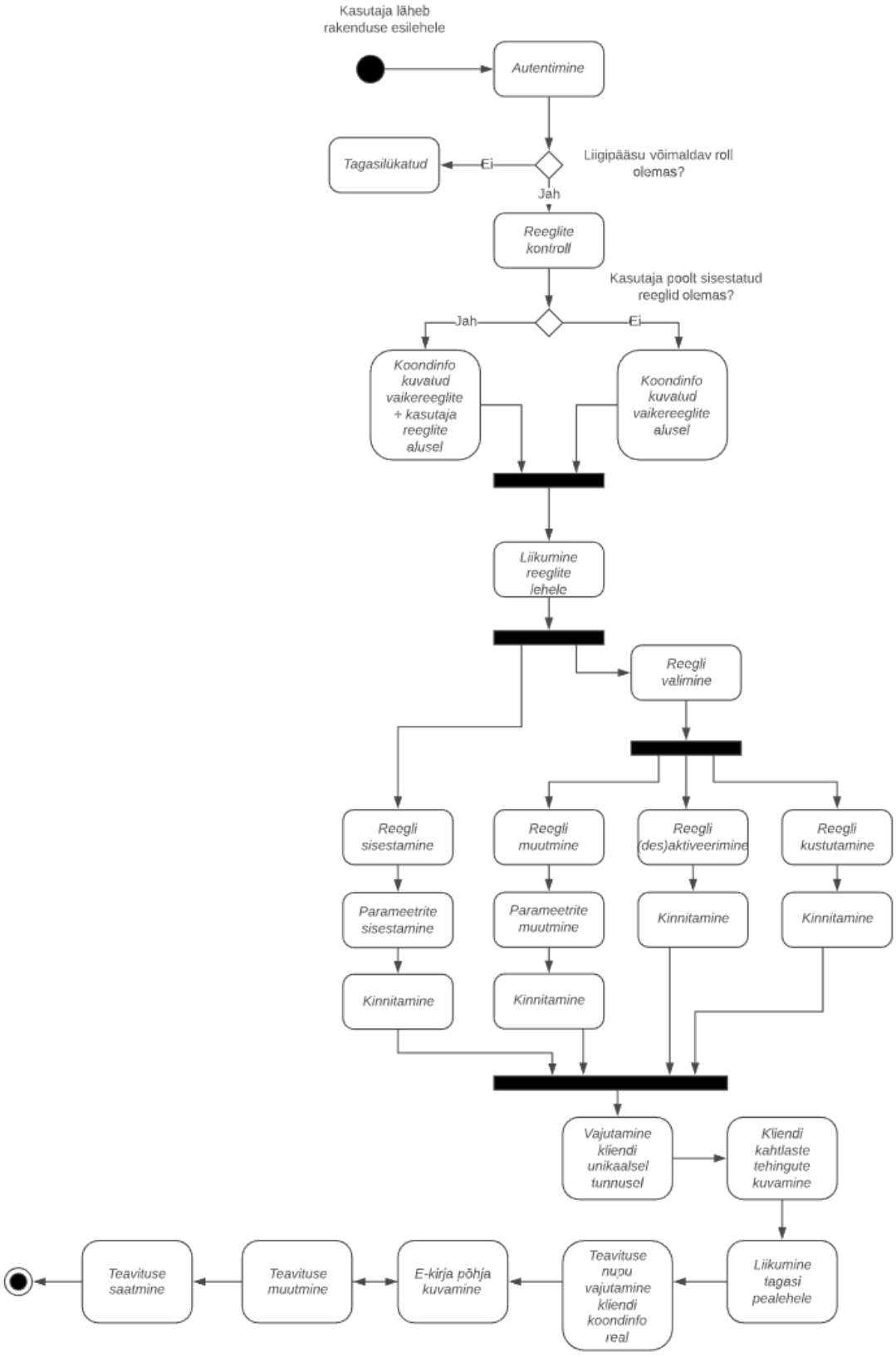
KMV-pettuste monitooringusüsteemi kontekstimudelil on väga üldisel tasemel näha planeeritav süsteem ning sellega suhestuvad ettevõttesisesed ja välised süsteemid ning aktorid. Samuti on üldises plaanis vaadeldav, milline info süsteemide ja aktorite vahel liigub.



Joonis 17. KMV-pettuste monitooringu kontekstidiagramm (Allikas: autori koostatud).

7.5 Tegevusdiagramm

Alloleval joonisel on kujutatud KMV-pettuste monitooringusüsteemi tegevusdiagrammi. Protsess algab kasutaja autentimisest, seejärel kuvatakse kasutajale kaupmeeste koondinfo. Peale seda liigutakse reeglitega seotud tegevuste juurde ning vaadatakse kliendi pettusekahtlusega tehinguid. Viimasena saadetakse kliendile teavitus pettusekahtluse kohta.

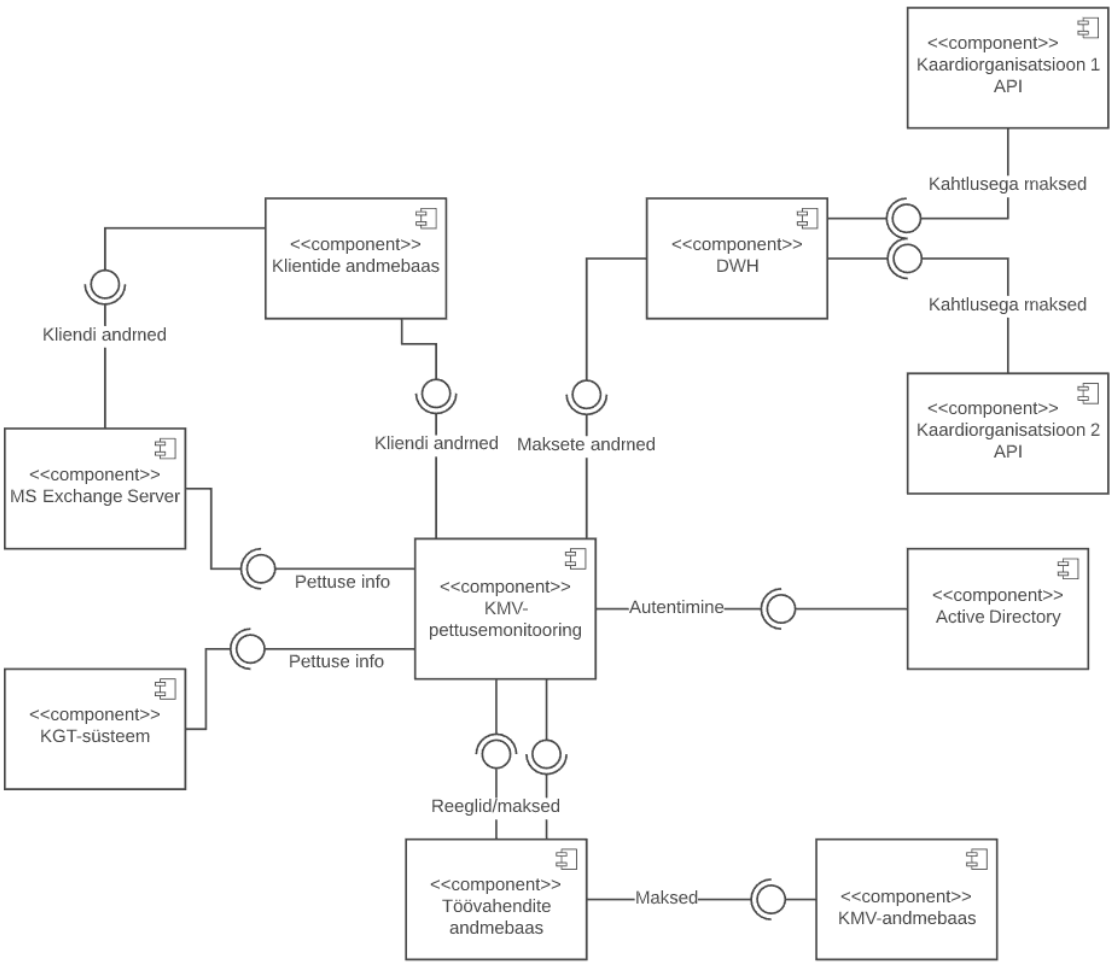


Joonis 18. KMV-pettuste monitooringu tegevusdiagramm (Allikas: autori koostatud).

8 Rakenduse arhitektuur

Järgenvas peatükis on kujutatud analüüsitava rakenduse arhitektuur. Selleks on kasutatud komponentdiagrammi (*Component Diagram*) ning seisundimuutuste diagramme (*Statechart Diagram*). Lisaks on visualiseeritud rakenduse juurutamise mudel evitusdiagrammina (*Deployment Diagram*).

8.1 Komponentdiagramm

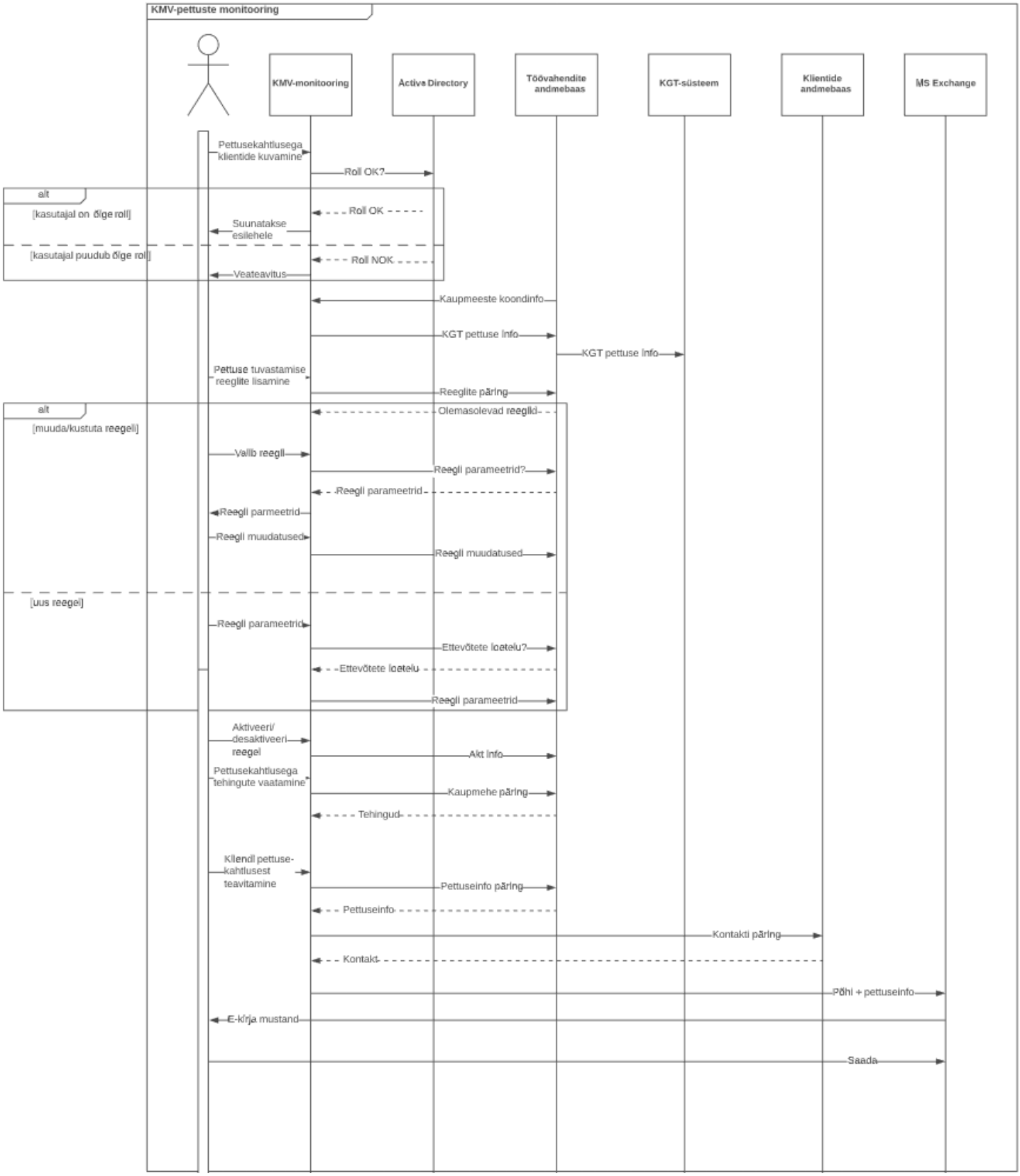


Joonis 19. KMV-pettuste monitooringusüsteemi komponentdiagramm (Allikas: autori koostatud).

8.1.1 Monitooringusüsteemi järgnevusdiagramm

Alloleval joonisel on kujutatud KMV-pettuste monitooringusüsteemi järgnevusdiagrammi. Diagramm kirjeldab üldistatud kujul süsteemide vahelist info liikumist. Diagrammil ei ole kujutatud alternatiivseid töövooge, kui need ei ole andmete

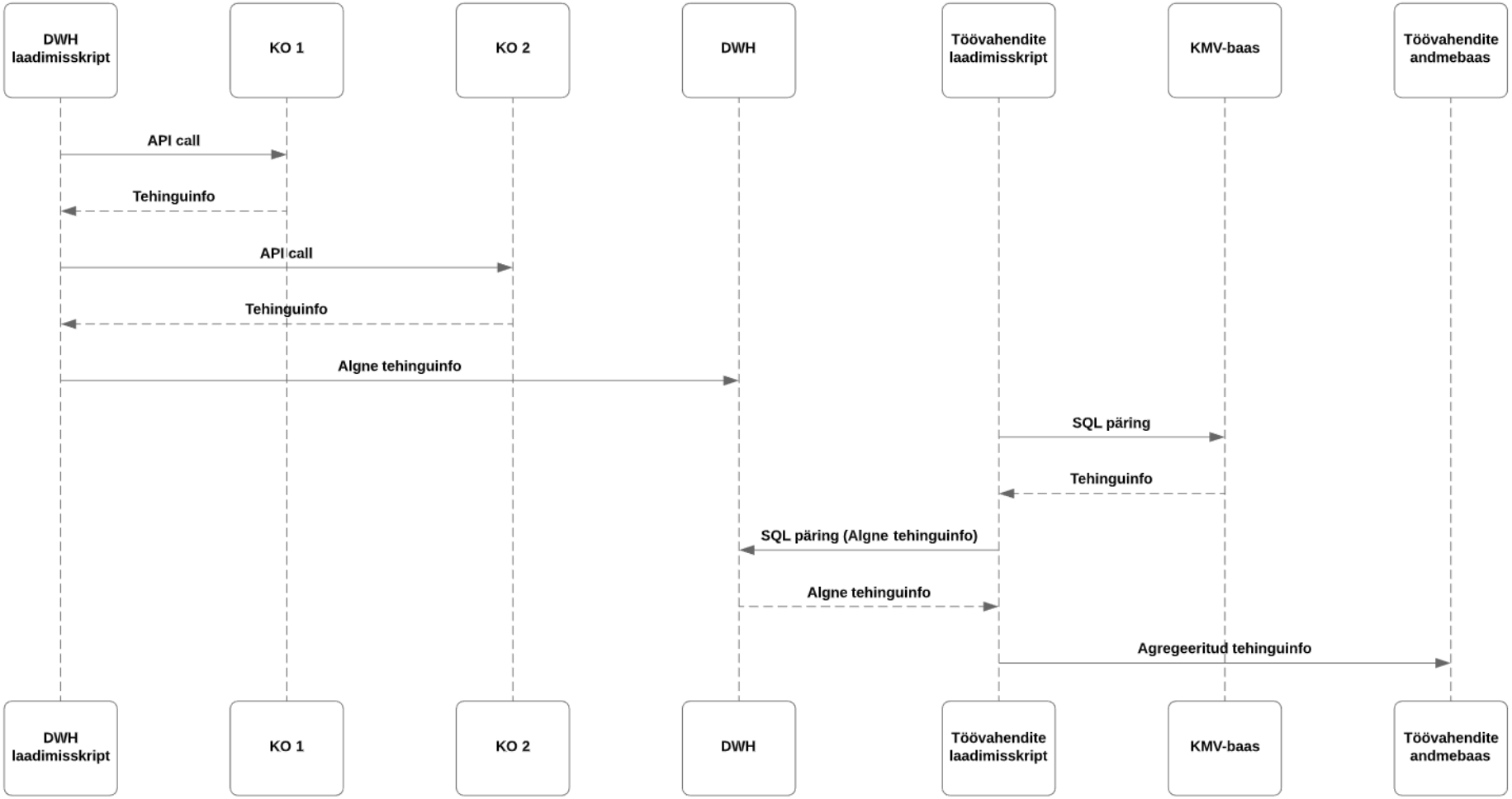
liikumise seisukohalt olulised. Samuti ei ole kujutatud kõiki kinnitamise ja tühistamise valikuid, kuna see muudaks joonise halvemini loetavaks.



Joonis 20. Monitooringusüsteemi järgnevusdiagramm (Allikas: autori koostatud).

8.1.2 Andmete laadimise järgnevusdiagramm

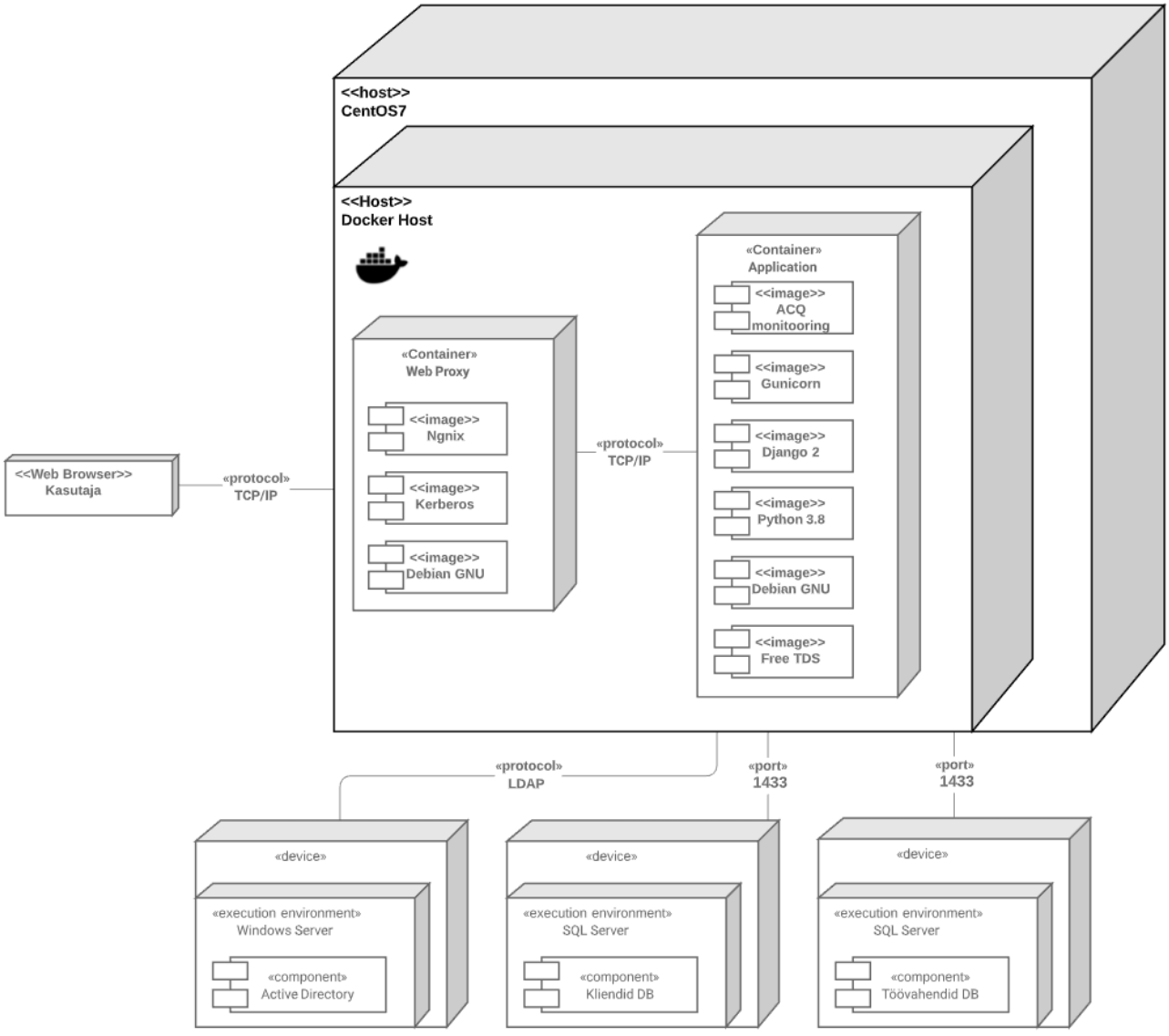
Järgneval joonisel on esitatud andmete laadimise järgnevusdiagramm. See kirjeldab seda, kuidas andmeid välistest süsteemidest andmebaasidesse laetakse ja hiljem töödeldakse. Kaardiorganisatsioon on joonisel lühendatud KO. Plaanilised laadimisskriptid luuakse, et vältida suurest andmehulgast tulenevat pikka ooteaega rakenduse erinevate päringute laadimisel. Esialgu on planeeritud laadimisskripte käivitada kord poole tunni tagant, kuid vajadusel on võimalik seda intervalli lühendada.



Joonis 21. Andmete laadimise järgnevusdiagramm (Allikas: autori koostatud).

8.2 Evitusdiagramm

Alloleval joonisel on kujutatud rakenduse lihtsustatud evitusdiagramm. Diagrammilt on välja jäetud väliste süsteemide komponendid, mis ei ole vajalikud kõnealuse rakenduse toimimiseks.

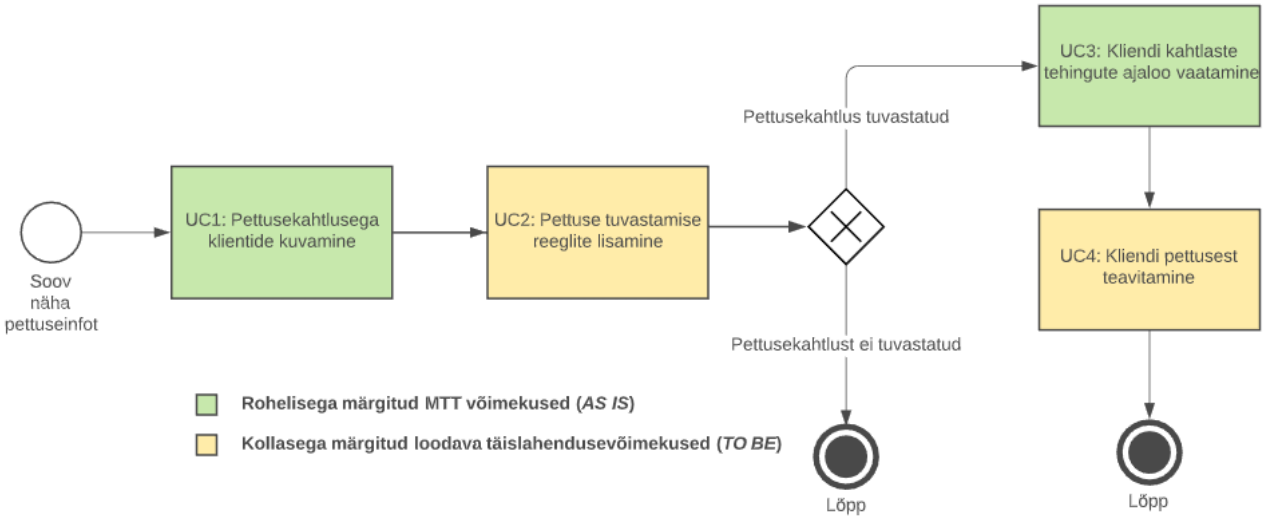


Joonis 22. Monitooringusüsteemi evitusdiagramm (Allikas: autori koostatud).

9 Minimaalne töötav toode

MVP (*Minimum Viable Product*) ehk eesti keeles öelduna minimaalne töötav toode, edaspidi MTT, kujutab endast esimest versiooni tootest, mida klient saab kasutada. Selle eesmärk on katsetada, kas idee päriselus toimib ja lõppkasutajale toode meeldib. Lisaks saab kiiresti tagasisidet vajalike muudatuste kohta, mida lõppkasutaja tootes näha soovib. Halvimal juhul võib teada saada, et toode ei olegi sellisel kujul teostatav. Selle hea külg on aga säästetud aeg ja raha, mis oleks võinud kuluda täieliku tootevisiooni realiseerimisel [28].

Alloleval joonisel on näidatud, millised kasutajalood on MTT loomisel kaetud. Kõik kasutajalugudes kirjeldatud punktid ja alternatiivsed töövood ei ole täidetud, samuti ei ole täidetud kõik mittefunktsionaalsed nõuded. MTT eesmärk on olnud kasutaja võimalikult kiire vabastamine andmete manuaalsest sisestamisest ning monitooringusüsteemi kiiremaks muutmine, samuti on nüüd võimalik samast süsteemist saada ülevaade kliendi kahtlaste tehingute ajaloost.



Joonis 23. Minimaalse töötava toote BPMN diagramm (Allikas: autori koostatud).

Kui lõppkasutaja oli MTT-d kasutanud ja andnud tagasisidet, täiustati rakenduses selle põhjal filtreerimis- ja otsingulahendust. Samuti on paranenud andmete kvaliteet – 2019 II poolaastal oli käsitsi sisestatud andmeridadest kuni 3% vigased, automaatselt laetud andmete puhul on veaprotsent 0. Varasema lahenduse keskmine laadimise aeg oli ligikaudu 1,5 tundi, MTT laeb andmeid alla 2 sekundi.

MTT on loodud Django ja JQuery DataTables raamistikke kasutades. Selle detailsem kirjeldus ja viide Githubi repositooriumile on leitav Lisast 1. Et mitte rikkuda ärisaladuse hoidmise nõuet, on koodi ja kasutajaliidest muudetud ning rakenduse andmebaasi on loodud geneerilised andmeväljad ja näidisandmed, samuti ei ole jagatud reeglite arvutamise ning andmete laadimise ja agregeerimise mooduleid. See tähendab, et MTT rakendus on muudetud geneeriliseks monitooringulahenduseks, millega saab soovi korral reegleid muutes jälgida erinevaid protsesse.

10 Järeldused

Magistritöö käsitles KMV-pettuste monitooringu süsteemi analüüsi teostamist ja minimaalse töötava toote loomist. Magistritöö eesmärk on täidetud, MTT on loodud ja lahenduse täisversiooni arendamist on võimalik alustada.

Magistritöö võiks olla aktuaalne finantssektoris kaardimaksete või riskivaldkonnas töötavate inimeste jaoks. Samuti võiksid sellest olla huvitatud inimesed, kellel on vaja analüüsida või luua monitooringulahendus ükskõik millise tehingu- või summapõhise protsessi jälgimiseks.

Autori hinnangul on MTT loomine sellise rakenduse puhul osutunud paremaks valikuks võrreldes madala täpsusega prototüübi loomisega. MTT kasutuselevõtuga on juba lahendatud neljast funktsionaalsest nõudest kaks, ning ajakulukas ja veaaldis manuaalne andmesisestus on automatiseeritud. See tähendab, et andmete sisestamise veamäär on langenud 3% pealt 0% juurde. Samuti toimivad MTT päringud võrreldes VBA-lahendusega suurusjärgu võrra kiiremini – eelneva 1,5 tunni asemel alla 2 sekundi.

2020. aasta suvel hindavad finantsettevõte juhatus ja huvitatud osapooled, kas alustada rakenduse täisversiooni arendamist. Juhul, kui lahenduse arendamisega soovitakse edasi liikuda, jõuab rakendus toodangukeskkonda hinnanguliselt 2020. aasta viimases kvartalis.

Autor toob välja potentsiaalsed rakenduse edasiarenduse soovitused tulevikuks, mis selle magistritöö skoobist välja jäid.

Lisada süsteemile tehingute blokeerimise reeglid

Blokeerimise reeglite jõustamine nõuab, et süsteem peab toimima reaalajas. Igat sisenevat makset hinnataks reeglite tingimustega ning tehing teostuks ainult kõikide blokeerivate reeglite läbimise puhul. See eeldab, et reeglid on väga täpsed ja valepositiivsete arv äärmiselt madal.

See edasiarendus on autori ja tellija arvates kindlasti vajalik ning tuleb lähitulevikus teostada. See nõuab põhjalikumat analüüsi andmeanalüütikute poolt, kes peavad aitama reeglid välja töötada ning hindama nende veamäära.

Töötada välja mudelipõhine pettuste tuvastamise süsteem

Masinõppe mudelil põhinev pettuste tuvastamise süsteem on kindlasti üks perspektiivikas edasiarendus, mida loodava monitooringusüsteemiga liidestada. Selle analüüsimise ja treenimisega on andmeanalüütikud magistritöö kirjutamise ajal juba alustanud, kuid selle töö skoopi ei ole see ajaraami sobimatuse ning ärisaladuse kaitse tõttu mahtunud. Mudelipõhine lahendus võib asendada või täiendada reeglitele tuginevat lähenemist.

Magistritöö tulemusel valminud MTT ja tulevikus valmiv täislahendus on korduvkasutatavad ka teiste protsesside monitoorimiseks. Selleks tuleb teha kohandused andmeväljades ja kuvamise reeglites ning ühendada rakendus uue andmebaasiga.

11 Kokkuvõte

Magistritöö tulemusel valmis äri- ja süsteemianalüüs, minimaalse töötava toote loomiseks ning seejärel tuleb teha otsus soovitud lahenduse täisversiooni arendamise osas. Töö tulemusel valminud minimaalne töötav toode on asendanud varem kasutusel olnud lahenduse.

Töö esimeses osas kirjeldas autor kaardipettuste valdkonda ja defineeris probleemipüstituse.

Teises osas sõnastati töö eesmärk – teostada äri -ja süsteemianalüüs KMV-pettuste monitooringusüsteemi arendamiseks ning luua minimaalne töötav toode. Samuti kirjeldatakse autori roll ning defineeritakse töö skoopi kuuluvad ja töö skoobist välja jäävad teemad.

Kolmandas osas valiti arendusmetoodika ja kirjeldati ning analüüsiti rakenduse äri- ja süsteemianalüüsil kasutatavate metoodikate valikuid.

Neljandas osas analüüsiti erinevaid kaardipettuste tuvastamise ja ennetamise meetmete liike ning kirjeldati ja põhjendati käesoleva töö jaoks tehtud valikuid. Samuti võrreldi alternatiivseid lahendusi lõppkasutaja esitatud olulisemate nõuetega.

Viiendas osas viidi läbi ärianalüüs ja kirjeldati huvitatud osapoolte rolle RACI-põhimõtte järgi. Intervjuude põhjal loodi rakenduse ärikirjeldus ja ärireeglid ning koostati äriinfo mudel olemi-suhte diagrammina. Samuti kirjeldati ja visualiseeriti äriprotsessid BPMN diagrammidena.

Kuuendas osas viidi läbi süsteemianalüüs, seejuures kirjeldati süsteemi kasutusmalle:

1. UC1: Pettusekahtlusega klientide kuvamine;
2. UC2: Pettuse tuvastamise reeglite lisamine;
3. UC3: Kliendi kahtlaste tehingute ajaloo vaatamine;

4. UC4: Kliendi pettusest teavitamine.

Kasutusmallid visualiseeriti kasutusmallide mudelina. Samuti loodi tegevusdiagramm süsteemi kõikide ülesannete osas. Lisaks põhjendatakse rakenduse arendamiseks ja juurutamiseks kasutavate tehnoloogiate ja raamistike valikut. Tehnoloogiate valikul peeti silmas arendajata oskuseid kui ka ettevõttes juba kasutusel olevaid tehnoloogiaid ja ettevõtte arhitektuuri.

Seitsmendas osas defineeriti rakenduse arhitektuur. Süsteemid kujutati komponentdiagrammil ning andmete liikumine toodi välja monitooringulahenduse ja andmete laadimise järgnevusdiagrammidel. Lisaks visualiseeriti rakenduse juurutamise mudel evitusdiagrammina.

Kaheksandas osas kirjeldati magistritöö tulemusel loodud minimaalse töötava toote funktsionaalsust. Samuti kirjeldati lõppkasutajalt minimaalse töötava toote kohta saadud tagasisidet.

Üheksandas osas anti hinnang töö käigus saavutatud eesmärkide täitmisele. Autor tõi välja projekti edasise tegevuskava, andis soovitusi edasiarendusteks ning hindas lahenduse korduvkasutuse potentsiaali.

Magistritöö probleem on lahendatud ja seatud eesmärgid täidetud.

Kasutatud allikad

[1] Finextra, „Three Types of Merchant Fraud: A Guide For Merchant Acquirers," [Võrgumaterjal]. Available: https://www.finextra.com/blogposting/14769/three-types-of-merchant-fraud-a-guide-for-merchant-acquirers. [Kasutatud 12 veebruar 2020].

[2] U.S. Department of Justice, „Fugitive Pleads Guilty In $200 Million Credit Card Fraud Scam," [Võrgumaterjal]. Available: https://www.justice.gov/usao-nj/pr/fugitive-pleads-guilty-200-million-credit-card-fraud-scam. [Kasutatud 12 veebruar 2020].

[3] European Central Bank, „Fifth report on card fraud," September 2018. [Võrgumaterjal]. Available: https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport201809.en.html#toc1. [Kasutatud 12 aprill 2020].

[4] I. Sommerville, Software Engineering, 10th Edition, Essex: Pearson Education Limited, 2016.

[5] Atlassian, Inc., „What is kanban?," [Võrgumaterjal]. Available: https://www.atlassian.com/agile/kanban. [Kasutatud 4 aprill 2020].

[6] IIBA, „A Guide to the Business Analysis Body of Knowledge V3.0," 2015. [Võrgumaterjal]. Available: http://www.the-aba.com/administrator/components/com_event/uploads/59014e456ca677.92343092BABOK_Guide_v3_Member.pdf. [Kasutatud 13 veebruar 2020].

[7] Association of Business Process Management Professionals, „Business Process Management Body of Knowledge," [Võrgumaterjal]. Available: https://cdn.ymaws.com/www.abpmp.org/resource/resmgr/Docs/ABPMP_CBOK_Guide_English.pdf. [Kasutatud 5 aprill 2020].

[8] Ernst & Young Baltic AS, Protsessianalüüsi käsiraamat, 2012.

[9] Kaitseministeerium, „Protsesside analuus ja kaardistamine,“ 2009. [Võrgumaterjal]. Available: https://www.slideshare.net/kaidopalu/protsesside-analuus-jakaardistamine?from_action=save. [Kasutatud 10 aprill 2020].

[10] A. Cockburn, Writing Effective Use Cases, Indianapolis: Addison-Wesley, 2001.

[11] M. C. P. J. j. G. O. I. Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, New York: Addison-Wesley, 1992.

[12] A. B. Al-Badareen, M. H. Selamat ja M. A. Jabar, „Software Quality Models: A Comparative Study,“ %1 *Software Engineering and Computer Systems: Second International Conference*, Kuantan, 2011.

[13] Professional Services Plus, „Think you’ve got your requirements defined? Think FURPS!,“ [Võrgumaterjal]. Available: http://www.psplus.ca/articles/think-youve-got-your-requirements-defined-think-furps/. [Kasutatud 25 aprill 2020].

[14] P. Eeles, „Capturing Architectural Requirements,“ [Võrgumaterjal]. Available: https://www.ibm.com/developerworks/rational/library/4706.html. [Kasutatud 16 aprill 2020].

[15] Riigi Infosüsteemi Amet, „Relatsiooniliste andmemudelite koostamise juhend ver.1.0,“ 2015. [Võrgumaterjal]. Available: https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/relatsiooniliste_andmemudelite_koostamise_juhend_ver._1.0.pdf. [Kasutatud 2 aprill 2020].

[16] M. Blaha, Object - Oriented Modeling and Design with UML, Pearson Education, 2004.

[17] J. G. J. Valacich, Modern Systems Analysis and Design, Pearson Education, Inc, 2017.

[18] Visual Paradigm, „What is Component Diagram?,“ [Võrgumaterjal]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/. [Kasutatud 5 märts 2020].

[19] M. Pragya, R. K. Sharma, R. Rastogi ja . Kumar, „Credit Card Fraud Detection System,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/341178406_CREDIT_CARD_FRAUD_DETECTION_SYSTEM. [Kasutatud 15 mai 2020].

[20] G. Sparks, „Database Modelling in UML,“ [Võrgumaterjal]. Available: http://www.methodsandtools.com/archive/archive.php?id=9. [Kasutatud 25 aprill 2020].

[21] K. Bittner ja I. Spence, Use Case Modeling, Boston: Addison Wesley, 2002.

[22] Kite, „Flask vs. Django: Choose Your Python Web Framework,“ [Võrgumaterjal]. Available: https://kite.com/blog/python/flask-vs-django-python/. [Kasutatud 15 märts 2020].

[23] Meta Garden LLC, „Django vs Flask - A Practictioner's Perspective,“ [Võrgumaterjal]. Available: https://devel.tech/features/django-vs-flask/. [Kasutatud 15 märts 2020].

[24] D. Roy Greenfeld ja A. Roy Greenfeld, Two Scoops of Django 1.11: Best Practices for the Django Web Framework, Two Scoops Press, 2017.

[25] N. Gift, K. Behrman, A. Deza ja G. Gheorghiu, Python for DevOps: Learn Ruthlessly Effective Automation, O'Reilly Media, 2019.

[26] Geekflare, „Docker vs Virtual Machine – Understanding the Differences,“ [Võrgumaterjal]. Available: https://geekflare.com/docker-vs-virtual-machine/. [Kasutatud 12 mai 2020].

[27] Atlassian Inc., „Continuous integration vs. continuous delivery vs. continuous deployment,“ [Võrgumaterjal]. Available: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment. [Kasutatud 12 mai 2020].

[28] Veebimajutus.ee, „Mis on MVP ja milleks seda tarvitada?,“ [Võrgumaterjal]. Available: https://www.veebimajutus.ee/blogi/mvp-toode. [Kasutatud 14 mai 2020].

[29] McKinsey, [Võrgumaterjal]. Available: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Where%20machines%20could%20replace%20humans%20and%20where%20they%20cant/Where-machines-could-replace-humans-and-where-they-cant-yet.ashx. [Kasutatud 12 veebruar 2020].

[30] Worldline, [Võrgumaterjal]. Available: https://worldline.com/content/dam/worldline/documents/publications/brochures/onlinewatcher-en.pdf. [Kasutatud 12 veebruar 2020].

[31] Object Management Group, Inc, „Business Process Model and Notation,“ [Võrgumaterjal]. Available: http://www.bpmn.org/. [Kasutatud 16 12 2018].

[32] S. A. White ja D. Miers, „BPMN Modeling and Reference Guide,“ 2008. [Võrgumaterjal]. Available: http://media.techtarget.com/Syndication/ENTERPRISE_APPS/BPMNModeling_and_Reference_Guide_Digital_Edition_G360.pdf. [Kasutatud 22 12 2018].

[33] K. Wiegers ja J. Beatty, Software Requirements (3rd Edition), Microsoft Press, 2013.

[34] T. Hathaway ja A. Hathaway, Data Flow Diagrams - Simply Put!: Process Modeling Techniques for Requirements Elicitation and Workflow Analysis, Createspace Independent Publishing Platform, 2016.

[35] A. V. Aho, J. D. Ullman ja J. E. Hopcroft, Data Structures and Algorithms, Pearson, 1984.

[36] Visual Paradigm, „What is Data Flow Diagram?,“ [Võrgumaterjal]. Available: https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/. [Kasutatud 11 Aprill 2020].

[37] IEEE, „Guide to the Software Engineering Body of Knowledge V3.0,“ 2014. [Võrgumaterjal]. Available: https://www.computer.org/web/swebok/v3. [Kasutatud 27 01 2019].

[38] M.-C. Lee, „Software Quality Factors and Software Quality,“ *British Journal of Applied Science & Technology,* 2014.

[39] S. Wagner, Software Product Quality Control, Springer-Verlag Berlin Heidelberg, 2013.

[40] SmartDraw, LLC, „Data Flow Diagram,“ [Võrgumaterjal]. Available: https://www.smartdraw.com/data-flow-diagram/. [Kasutatud 11 aprill 2020].

Lisa 1 – MTT ülevaade

Minimaalse töötava tootena on loodud Django rakendus, millega on võimalik monitoorida erinevaid reegleid. Kui reegli piirmäär on ületatud, värvub lahter punaseks. Andmete laadimise ja reeglite arvutamise mooduleid ei ole ärisaladuse tõttu jagatud koodile lisatud. Siiski saab hea ülevaate, kuidas monitooringulahenduse kasutajaliides ja tööprotsess toimib.

MTT kuvatõmmised

1. **Pettusekahtlusega klientide kuvamine – koondvaade**

   Kuvatõmmisel on kujutatud rakenduse avakuva, kus on ettevõtete info agregeeritud kujul ning erinevate reeglite tasemed. Kui piirmäär on ületatud, värvub lahter punaseks.

2. **Otsing**

   Otsida on võimalik ettevõtte nime või nime osa järgi.

3. **Järjestamine**

   Järjestada saab kõikide tunnuste alusel. Kuvatõmmisel on selleks valitud summa.

4. **Pettusekahtlusega tehingute vaatamine – detailvaade**

   Vajutades ettevõtte unikaalsele tunnusele, on võimalik kuvada kliendi kõik kahtlased tehingud eraldi real.

Monitoring

- MONITORING

Show 25 entries

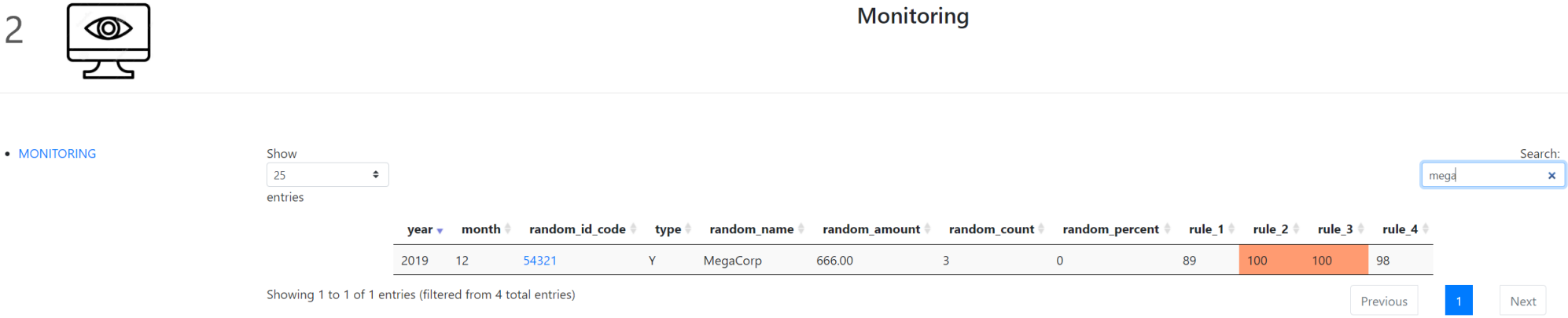
Search:

year	month	random_id_code	type	random_name	random_amount	random_count	random_percent	rule_1	rule_2	rule_3	rule_4
2020	1	12345	X	Company Inc	123.00	1	1	12	17	89	100
2020	2	12345	X	Company Inc	555.00	1	0	45	100	23	11
2020	2	234567	Y	Business Ltd	321.00	1	0	100	77	32	21
2019	12	54321	Y	MegaCorp	666.00	3	0	89	100	100	98

Showing 1 to 4 of 4 entries

Previous 1 Next

Joonis 24. MTT pettusekahtlusega klientide kuvamine – koondvaade (Allikas: autori koostatud – kuvatõmmis).



Joonis 25. MTT otsing (Allikas: autori koostatud – kuvatõmmis).

Joonis 26. MTT järjestamine (Allikas: autori koostatud – kuvatõmmis).

4 Monitoring

- MONITORING

Company Transactions

random_name	random_amount	date	type	random_id_code
MegaCorp	10.00	Dec. 11, 2019	Y	54321
MegaCorp	636.00	Dec. 3, 2019	Y	54321
MegaCorp	20.00	Dec. 2, 2019	Y	54321

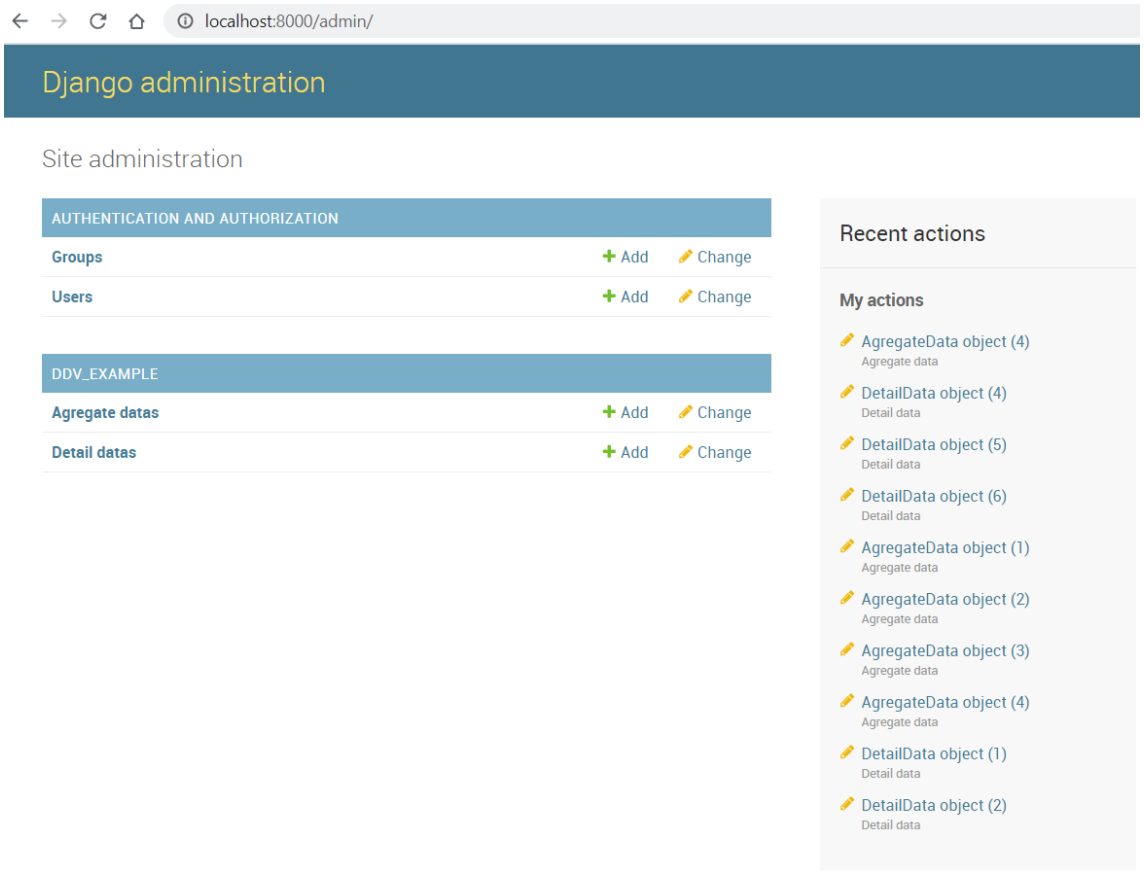
Joonis 27. MTT pettusekahtlusega klientide vaatamine (Allikas: autori koostatud – kuvatõmmis).

Käivitamise juhised

Järgnevalt on kirjeldatud, kuidas MTT-rakendust lokaalselt käivitada. Käivitamiseks on vajalik, et arvutisse oleks paigaldatud Dockeri rakendus koos docker-compose funktsionaalsusega.

1. Klooni või lae alla kood Githubist. https://github.com/Jennosian/djangoapp
2. Liigu käsureal Monitoring_Docker_MVP kaustale.
3. Käivita rakendus käsuga docker-compose up
4. Liigu brauseris http://localhost:8000/

Kui lisaks andmete vaatamisele on ka soov andmeid sisestada või muuta, siis selleks tuleb teha Django admin konto käsuga „python manage.py createsuperuser“. Peale seda on võimalik luua kasutajanimi ja parool. Admin on ligipääsetav http://localhost:8000/admin.



Joonis 28. MTT admin (Allikas: autori koostatud – kuvatõmmis).

Magistritöös esitletav MTT on ühendatud kaasas oleva SQLite näidisbaasiga, et geneerilisi andmeid oleks hõlpsam rakendusega koos edastada. Dockerfile kirjelduses on olemas kõik seadistused, et oleks võimalik luua ühendus üle FreeTDS protokolli ka enda valitud MS SQL Serveriga. Selleks tuleks muuta Django rakenduse settings.py faili DATABASE seadeid järgnevalt:

```
DATABASES = {
    'default': {
        'ENGINE': 'sql_server.pyodbc',
        'NAME': 'andmebaasi_nimi',
        'HOST': 'host IP',
        'USER': 'kasutajanimi',
        'PASSWORD': 'parool',
        'PORT': '1433',
        'OPTIONS': {
            "driver": "FreeTDS",
            "host_is_server": True,
            "extra_params": "tds_version=7.4",
          }
        }
    }
```