TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kenert Vaino 179221IADB

Rahvusvahelise menetlusprotsessi digitaliseerimine Prokuratuuri infosüsteemis

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

Autorideklaratsioon

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

Autor: Kenert Vaino

12.05.2023

Annotatsioon

Käesolevas bakalaureusetöös analüüsitakse kohtueelsete ja rahvusvaheliste kriminaalmenetluste menetlustoimingute dokumenteerimisprotsessi erisusi ja viimase erisustest tingitud probleeme menetlusprotsessi kestel. Lõputöö eesmärgiks on välja töötada vajalik lisamoodul Prokuratuuri infosüsteemile, mis võimaldaks selle integreerida rahvusvaheliseks menetlustoimingute edastamiseks kasutatava turvatud suhtluskanaliga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, viite peatükki, 14 joonist, ühte tabelit.

Abstract

Digitalization of International Procedures in the Prosecutor's Office's Information System

The thesis analyses the differences between the documentation process of pre-trial and international criminal proceedings and the shortcomings caused by the latter. The goal of this thesis is to develop an extra module for the Prosecutor's Office's Information System, which would allow it to be integrated with a secured communication channel used for exchanging information regarding international criminal proceedings.

The thesis is in Estonian and contains 25 pages of text, 5 chapters, 14 figures, 1 table.

Lühendite ja mõistete sõnastik

AAA	*Arrange, Act, Assert*; ülesseadmine, tegutsemine, valideerimine; muster testmeetodite struktureerimiseks
API	*Application Programming Interface*; rakendusprogrammliides; liides teiste rakenduste või muude teenustega suhtlemiseks
arenduskeskkond	Arendustöödeks eraldi ülesseatav süsteemide kogum, mis on eraldatud päris kasutajate andmetest
C#	Loetakse *C-Sharp*; ettevõtte Microsoft poolt arendatav programmeerimiskeel
CRON avaldis	Avaldis, millega defineeritakse protsessi käivitumissagedus, näiteks avaldis 0 */6 * * * tähistab käivitamist iga kuue tunni tagant
CRUD-operatsioonid	*Create, Read, Update, Delete* ehk andmete loomise, lugemise, uuendamise ja kustutamise tegevused
Dockerfile	Skriptifail, millega pakendatakse rakendus Docker konteinerisse
E-Codex	*e-Justice Communication via Online Data Exchange*; Euroopa Komisjoni projekt detsentraliseeritud taristu arendamiseks, et digitaalselt ühendada Euroopa Liidu õigusasutusi
EL	Euroopa Liit; Euroopa riikidest koosnev poliitiline organisatsioon
ET	E-Toimik; keskne infosüsteem, kuhu koondatakse andmed erinevatest sellega integreeritud infosüsteemidest
EXE-fail	*Executable file;* käivitatav programmifail
IDE	*Integrated Development Environment*; tarkvaraarenduseks loodud tarkvara, mis sisaldab endas tekstiredaktorit ja muid vajalikke mooduleid programmikoodiga töötamiseks
IIS	*Internet Information Services*; Windowsi operatsioonisüsteemil põhinev veebiserver
kohtueelne menetlus	Kohtus läbiviidava menetluse eelne uurimisprotsess, mille käigus selgitatakse välja toime pandud teo asjaolud ning otsustatakse edasised sammud
LTS	*Long-Term Support*; tarkvara versioon, millele on tagatud pikaajaline, keskmiselt 2-3 aasta pikkune, tugi selle arendanud ettevõtte poolt

PRIS	Prokuratuuri infosüsteem; infosüsteem, kus Prokuratuuri jt uurimisasutuste ametnikud registreerivad menetlustoiminguid ja teostavad päringuid teistesse X-teega liidestunud infosüsteemidesse
SOA	*Service-oriented architecture;* teenustel baseeruv tarkvaraarhitektuuri vorm
SOAP	*Simple Object Access Protocol*, lihtne objektipöörduse protokoll, XML-il põhinev veebiteenuste andmevahetusstandard
menetlusgrupp	Menetlusega seotud Prokuratuuri (või muu uurimisasutuse) ametnikud, kes teostavad menetlustoiminguid
menetlustoiming	Menetluse raames koostatav toiming, näiteks peale tunnistaja ülekuulamist koostatav ülekuulamisprotokoll
refaktoreerimine	Programmikoodi parendamine selle üldist loogikat muutmata
snapshot	Tõmmis andmeolemite mudelist kindlal ajahetkel
SQL	*Structured Query Language*, struktureeritud päringute keel, programmikeel andmebaasipäringute tegemiseks
XML	*Extensible Markup Language*, laiendatav märgistuskeel, struktureeritud andmete masinloetavaks tegemiseks
X-tee	andmevahetuse platvorm, mis võimaldab turvaliselt asutuste vahel teavet pärida ja vahetada [1]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Infotehnoloogia areng mõjutab inimkonda igapäevaselt aina enam ja enam. Väga suur osa inimeste elust leiab tänapäeval juba aset võrgus – üksteisega suheldakse kasutades mõnda digitaalset suhtluskanalit, arveid makstakse läbi finantsasutuste loodud online-lahenduste, toidu valmistamise asemel tellitakse toitu koju kasutades mõnda nutitelefoni rakendust ja veel palju muud. Ka Eesti Vabariigi kohtueelse menetluste uurimise juhil ning riiklike süüdistuste esitajal, milleks on Prokuratuur, on soov käia kaasas infotehnoloogiliste arengutega ning selle visiooni üheks etapiks on kriminaalmenetluste digitaliseerimine. [2]

Käesoleva töö kirjutamise hetkeks on eelmainitud visiooniga jõutud nii kaugele, et kohtueelsete menetluste läbiviijatele – prokuröridele – on arendatud spetsiaalne infosüsteem, mis kannab nime Prokuratuuri infosüsteem (lühend PRIS), läbi mille on neil võimalik dokumenteerida erinevaid siseriiklikke menetlustoiminguid, näiteks kellegi tunnistajana ülekuulamisele kutsumine, ning tänu integratsioonile keskse infosüsteemiga E-Toimik edastada neid kõikidele menetluskoosseisu kuuluvatele isikutele.

Paraku aga ei ole antud visiooni realiseerimisega jõutud veel rahvusvaheliste kriminaalmenetluste digitaliseerimiseni – neid dokumente edastatakse ning hoiustatakse reeglina endiselt paberkandjatel, mis aeglustab oluliselt menetluste läbiviimise protsessi. Seetõttu on Euroopa Komisjoni tellimusel arendamisel ühine platvorm ja keskkond, mille jaoks on liikmesriikidel võimalik välja arendada integratsioonid oma kasutatavatele infosüsteemidele.

Eesti Vabariigis integreeritakse eelpool mainitud platvormiga Prokuratuuri kasutatav infosüsteem PRIS. Käesoleva bakalaureusetöö eesmärgiks on vajalike tarkvaraliste arendustööde läbiviimine, et realiseerida kahe platvormi vaheline andmevahetus. Arendustööde teostajaks on Justiitsministeeriumi haldusalasse kuuluv ettevõte Registrite ja Infosüsteemide Keskus (RIK), mille vastava projekti arendustiimi koosseisu kuulub tarkvaraarendaja rollis ka käesoleva bakalaureusetöö autor.

2 Probleemi analüüs

Vastavalt Kriminaalmenetluse seadustikule on Eesti Vabariigis kriminaalmenetluste läbiviijateks uurimisasutused (nt Politsei- ja Piirivalveamet, Maksu- ja Tolliamet) ning Prokuratuur. Rahvusvaheliste kriminaalmenetluste kontekstis, millele käesolev töö keskendub, on kriminaalmenetluste läbiviijaks Prokuratuur. [3]

Prokuratuuriseadus viitab Prokuratuurile kui Justiitsministeeriumi valitsemisalasse kuuluvale valitsusasutusele, kes tuginedes Prokuratuuri- ja muudele seadustele peab sõltumatul moel läbi viima kohtueelseid kriminaalmenetlusi. [4] Menetlust läbiviivad prokuratuuri ametnikud ehk prokurörid peavad tagama menetlusprotsessi seaduspärasuse ning vajadusel kohtusse edastatava menetluse käigus täitma ka riikliku süüdistaja rolli.

2.1 Kriminaalmenetluste menetlustoimingute sisestamise protsessid

Käesolevas alampeatükis kirjeldatakse prokuröride poolt koostatavate menetlustoimingute dokumentide registreerimise protsess kohtueelsetes ja rahvusvahelistes kriminaalmenetlustes, et anda ülevaade nende protsesside erinevusest.

2.1.1 Kohtueelsete kriminaalmenetluste menetlustoimingute sisestamise protsess

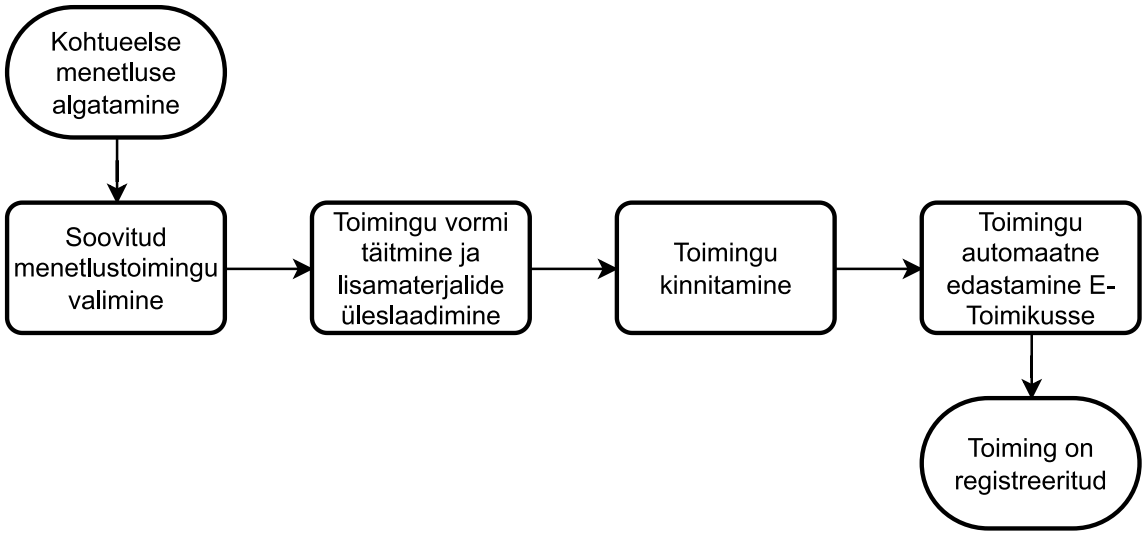
Kuna prokurör on oma töös kohustatud tuginema seadustele, siis eeldab ka kohtueelse kriminaalmenetluse alustamine, et menetluse läbiviimiseks on seaduslik ajend. Kui antud nõue on täidetud, registreerib prokurör menetluse alustamise digitaalselt, kasutades selleks prokuratuurile loodud infosüsteemi - Prokuratuuri infosüsteem ehk PRIS. Peale menetluse algatamist on kasutajal (ehk antud juhul prokuröril) võimalik teha loodud menetlusega erinevaid tegevusi – sisestada menetlustoiminguid, lisada menetlust läbiviivasse koosseisu uurijaid ja teisi prokuröre jm.

Menetlustoimingu sisestamiseks peab prokurör (või vastavat süsteemset privileegi omav muu prokuratuuri ametnik, nt. kantselei) valima konkreetsesse menetlusse lubatud toimingute seast ühe toimingu, mida ta soovib sisestada, mille peale suunatakse kasutaja toimingu sisestusvormile, kus on vaja täita ära vastavad väljad. Sõltuvalt toimingust on

kasutajal võimalik juurde lisada ka seonduvaid faile, nt dokumendid, helisalvestised, pildimaterjal jm.

Peale nõutud andmete sisestamist kinnitab kasutaja toimingu, mille käigus edastab infosüsteem sisestatud andmestiku ja üleslaetud failid kesksesse süsteemi E-Toimik ehk ET, kasutades selleks keskset infrastruktuuri X-tee, kus need talletatakse. Tänu integreeritusele E-Toimikuga muutub sisestatud menetlustoiming automaatselt kättesaadavaks ka teistele prokuratuuri töötajatele ja muudele menetlusgruppi kuuluvatele isikutele nende enda kasutatavates infosüsteemides, eeldusel, et on täidetud vajalike süsteemsete õiguste olemasolu.

Joonisel 1 on antud lühiülevaade uue kohtueelse menetluse ja sellega seonduva toimingu sisestamisest.



Joonis 1. Kohtueelse menetluse menetlustoimingu sisestamise protsess

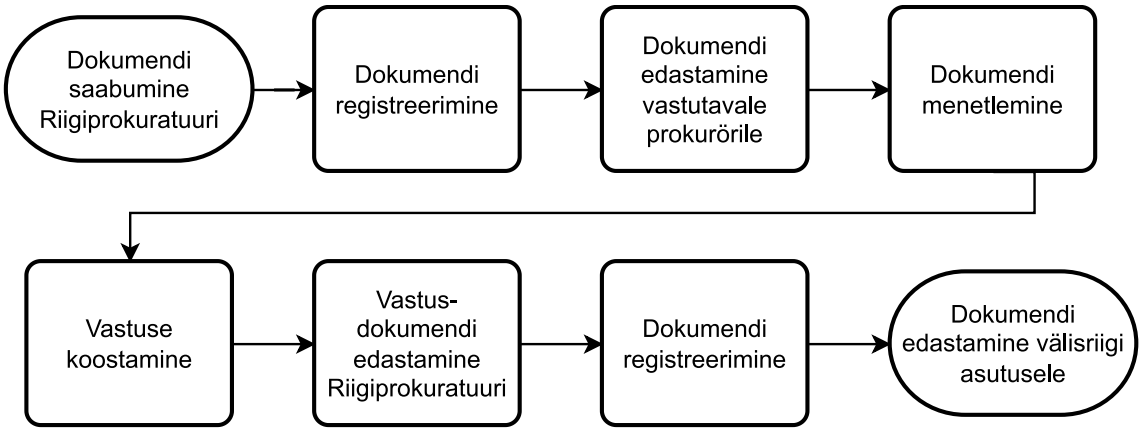
2.1.2 Rahvusvaheliste kriminaalmenetluste menetlustoimingute edastamise ja talletamise protsess

Erinevalt kohtueelsetest (ehk käesoleva töö kontekstis siseriiklikest) menetlustest ei ole rahvusvaheliste kriminaalmenetluste menetlustoimingute edastamiseks ja talletamiseks keskset süsteemi. Nagu varasemalt välja toodud, siis vahetavad erinevate riikide kriminaalmenetlusi läbiviivad asutused omavahel informatsiooni kasutades reeglina selleks posti-/kullerteenust. Selleks, et riik A (ehk taotleja) saaks vajalikku informatsiooni riigilt B (ehk täitja), tuleb tal koostada sõltuvalt taotluse eesmärgist kindlatele direktiividele vastava vormiga taotlus ning edastada see täitva riigi vastavale

uurimisasutusele. Selle taotluse peale peab täitja riigi asutuse vastutav ametnik kujundama oma otsuse, kas võtta asi menetlusse (informatsiooni kogumiseks) ning edastama oma otsuse taotluse esitanud asutusele, kasutades selleks jällegi paberdokumenti ja postiteenust.

Eestile esitatud taotluse korral (ehk siis Eesti Vabariigi prokuratuur on täitva asutuse rollis) registreerib Riigiprokuratuuri kantseleiametnik dokumendi saabumise nende asutuse kasutatavas dokumendihaldussüsteemis, peale mida see edastatakse menetlemiseks vastavat töökohustust täitma määratud prokurörile. Kuna dokument on paberkandjal ja prokuröri töökoht ei pruugi asuda aadressil, kuhu antud dokument välisriigi poolt edastati, siis tehakse seda füüsilisel meetodil (posti-/kullerteenus), et toimetada antud dokument prokurörini. Kui prokurör on koostanud edastatud taotlusele vastuse, liigub dokument uuesti samu etappe läbides tagasi taotlevasse asutusse. Originaaldokument arhiveeritakse arhiivis, kus sellega on vastavat õigust ja vajadust omavatel isikutel võimalik tutvuda.

Joonisel 2 on kujutatud saabuva rahvusvahelise menetlusdokumendi ja sellele vastamise protsess.



Joonis 2. Eestile edastatud rahvusvahelise menetluse toimingu ja sellele vastuse koostamise elutsükkel

2.2 Rahvusvaheliste kriminaalmenetluste läbiviimise protsessi puudused

Eelnevast nähtub, et kui kohtueelse kriminaalmenetlusega seonduvaid dokumente on võimalik talletada ja edastada digitaalselt, (siinkohal on oluline mainida, et antud töö kirjutamise hetkel on juriidiline õigus siiski veel pabertoimikul), siis rahvusvaheliste

kriminaalmenetlustega seonduvaid dokumente talletatakse, edastatakse ning töödeldakse endiselt füüsilisel teel, s.t paberkandjal ning kasutades füüsilisi edastusmeetodeid. See toob endaga kaasa järgnevaid probleeme:

1. Dokumentide edastamisest tingitud ajakulu - kui kohtueelsete kriminaalmenetluste puhul on kesksesse infosüsteemi andmete edastamiseks kuluv aeg mõõdetav sekundites, siis füüsiliste paberkandjatel olevate materjalide on sellest kahtlemata aeglasem, ulatudes ajaliselt päevadest kuni harvematel juhtudel ka kuudeni.

2. Dokumentide kättesaadavus – kuna dokumendid paiknevad füüsilises asukohas, siis ei ole teabevajadusega isikutele (nt. teine prokurör) tagatud nendele ligipääsu igal ajahetkel.

Lisaks eelnevatele peamistele probleemidele võib vaadelda ka, kuid ei kuulu käesoleva töö skoopi, paberkandjate ja nende füüsilisest edastamisest tingitud lisakulusid kui ka nendest tingitud mõju keskkonnale.

3 Probleemi lahendus

Eelnevas peatükis selgus, et kui Eestis on erinevate riigiteenuste vaheline suhtlus lahendatud üle X-tee platvormi, siis üleeuroopaliseks infovahetuseks sellist võimekust veel realiseeritud pole.

3.1 E-Codex platvorm

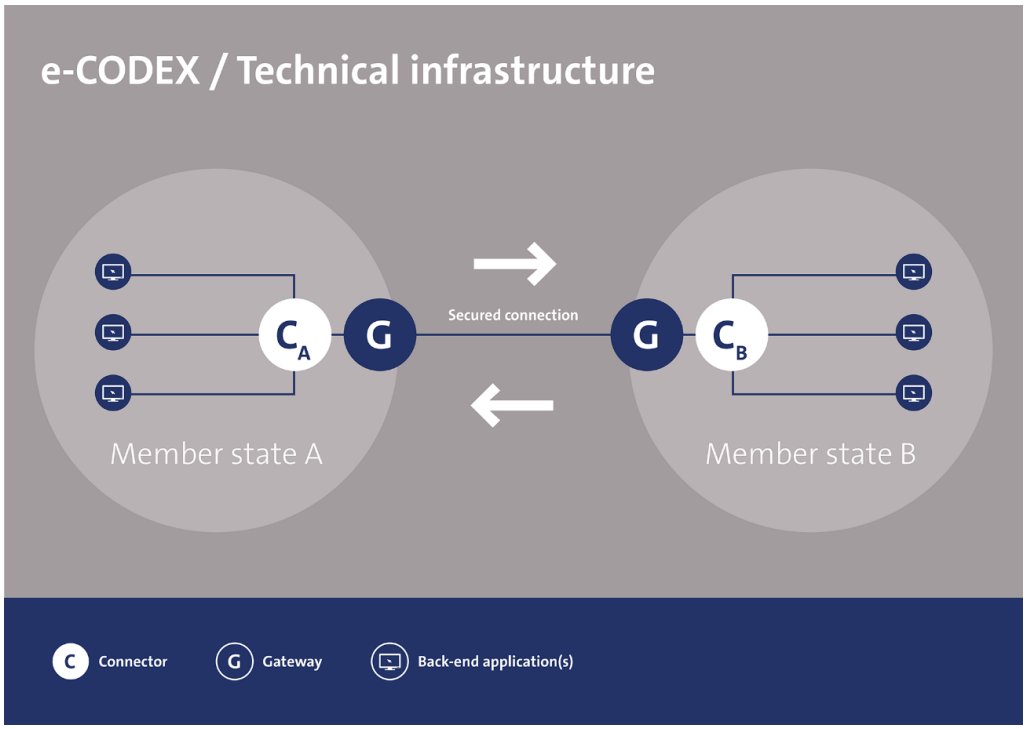
Rahvusvaheliste kriminaalmenetlustega seotud infovahetuse aeglusest tingitud mõjusid on täheldanud ka teised Euroopa Liidu liikmesriigid, mistõttu on Euroopa Komisjoni tellimusel välja arendamisel tehniline lahendus, mida nimetatakse e-Codex'iks - *e-Justice Communication via Online Data Exchange*. [5]

E-Codex'i eesmärgiks on kiirendada Euroopa Liidu liikmesriikide poolt läbiviidavate liikmesriikide vaheliste õigustoimingutega seotud dokumentide edastamist, luues detsentraliseeritud infovahetusplatvormi, mille külge on liikmesriikidel võimalik liidestada oma kasutatavad infosüsteemid.

Infosüsteemi liidestamiseks e-Codex platvormiga tuleb igal liikmesriigil üles seada kaks komponenti - *gateway* (eesti k. lüüs) ja *connector* (eesti k. ühendusliides). Nimetatud komponente ei pea riik ise valmis arendama, vaid need on loodud eelpoolmainitud Euroopa Komisjoni tellimuse raames. Liikmesriigi, täpsemalt selle asutuse, kellel on vajadus antud platvormiga ühendus luua, ülesandeks on komponentide paigaldus ning konfigureerimine vastavalt etteantud dokumentatsioonile.

Lüüsi peamiseks funktsiooniks on kommunikatsioon teiste riikide paigaldatud lüüsidega ning oma ühendusliidesega. Ühendusliidese põhifunktsioonid on infovahetus riigisiseste infosüsteemidega ning edastatavate andmete valideerimine. Nimetatud komponentide omavaheline andmevahetus toimub kasutades turvatud SOAP veebiteenuseid.

Joonisel 3 on selgitatud e-Codex platvormi ülesehitust.



Joonis 3. e-Codex'i platvormi ülesehitus [6]

3.2 Kavandatav lahendus

Peatükis 2.2 toodi välja kaks peamist probleemi rahvusvaheliste kriminaalmenetlustega seotud dokumentidega. Nendeks olid andmete edastamise kiirus ning hilisem ligipääs andmetele, mis mõlemad taanduvad samale peamisele faktorile – dokument ei ole elektrooniline, vaid paberkandjal. See tähendab, et on vajalik toimingute digitaliseerimine.

Tulles tagasi algprobleemide juurde, siis võimalikud lahendused nendele oleks järgmised:

1. Dokumentide edastamisest tingitud ajakulu – arendada välja vajalik liidestus PRIS'ile, et võimaldada andmete edastamist üle e-Codex platvormi

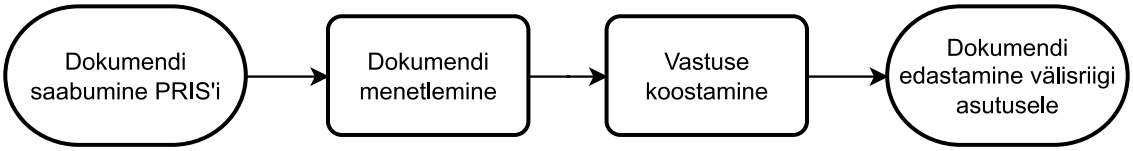
2. Dokumentide kättesaadavus – realiseerida rahvusvaheliste toimingute vormid PRIS'is, et oleks võimalik

    a. Välisriigist saabunud toiminguid kasutajale välja kuvada, hoiustada neid E-Toimikus ning neid töödelda

    b. Kasutajal ise koostada toiminguid ning edastada neid E-Toimikusse ja vastavale välisriigile

Kuna punktis 2 välja toodu on PRIS'is põhimõtteliselt juba realiseeritud siseriiklikult läbiviidavate menetlustoimingute näol ja arendustööd hõlmaksid peamiselt vajalike kasutajaliidese vaadete loomist, siis on käesoleva töö skoobiks punkti 1 lahendamine ehk vajalike arendustööde tegemine PRIS liidestamiseks E-Codex infovahetuskanaliga.

Võttes aluseks joonisel 2 kujutatud Eestile esitatud rahvusvahelise menetluse toimingu elutsükli, siis peale protsessi digitaliseerimist on selle dokumendi eeldatav elutsükkel vastavalt joonisel 4 kujutatule.



Joonis 4. Eeldatav Eestile esitatud rahvusvahelise toimingu elutsükkel peale selle digitaliseerimist

3.2.1 Funktsionaalsed nõuded loodavale lahendusele

Arendustööde tulemusena on vajalik täita järgnevad funktsionaalsed nõuded:

1. PRIS'il peab olema võimalik vastu võtta välisriikide õigusasutuste poolt edastatavaid menetlustoiminguid
2. PRIS'ist peab olema võimalik edastada rahvusvahelisi menetlustoiminguid välisriikide õigusasutustele

3.2.2 Mitte-funktsionaalsed nõuded loodavale lahendusele

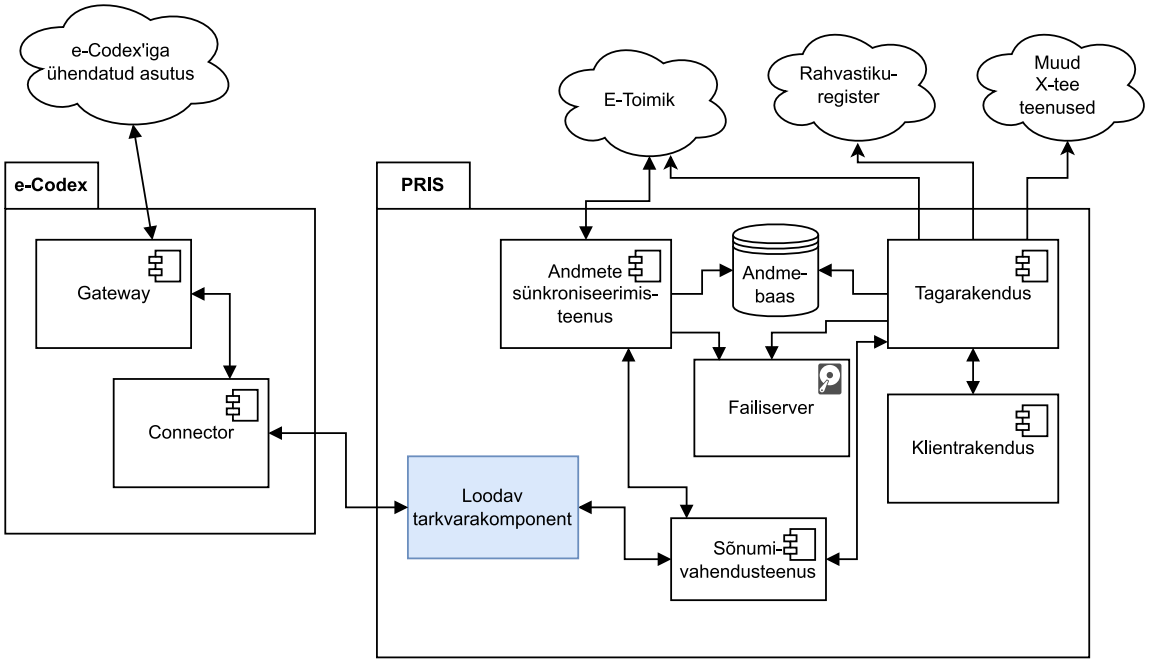
Loodav lahendus peab vastama järgnevatele ettevõtte üldistele ja ettevõtte peaarhitekti poolt käesolevale projektile seatud nõuetele:

1. Lahendus peab olema eraldiseisev tarkvarakomponent
2. Lahendus peab olema realiseeritud programmeerimiskeeles C# [7]
3. Lahendus peab olema realiseeritud .NET raamistikul [8], kasutades arendusprotsessi hetkel väljas olevat pikaajalise toega (*Long-Term Support* ehk LTS) versiooni
4. Lahendus peab olema juurutatav kasutades Docker [9] ja Kubernetes [10] tehnoloogiaid

Eelneva loetelu põhjal saab väita, et loodava tarkvarakomponendi kaheks oluliseks omaduseks on ettevõtte Microsoft poolt arendatavate programmeerimiskeele ja -raamistiku kasutamine. Antud nõuded tulenevad projekti teostava ettevõtte sisemisest kompetentsist – ettevõte on kasutanud Microsofti loodavaid tooteid pea kogu tegevusaja jooksul ning seetõttu on ka erinevate ettevõtte poolt teostatavate projektide arendusmeeskonnad üles ehitatud nõudes arendajatelt oskust eelpoolnimetatud tehnoloogiatega töötada. Ettevõtte näol on tegemist avaliku sektori ettevõttega ning samu tehnoloogiad kasutatakse suurel määral ka mitmetes teistes avaliku sektori infotehnoloogia ettevõtetes.

3.2.3 Kavandatava lahenduse arhitektuuriline paiknemine

Järgneval joonisel 5 on kujutatud loodava tarkvarakomponendi planeeritav paiknemine ja suhe Prokuratuuri infosüsteemi komponentidega.



Joonis 5. Arendatava tarkvarakomponendi paiknemine PRIS ja e-Codex suhtes

Esitatud jooniselt nähtub, et arendatava komponendi näol on tegemist eraldiseisva tarkvarakomponendiga, nagu oli ka sellele seatud üks mitte-funktsionaalsetest nõuetest. See tähendab, et esitatud funktsionaalsete nõuete täitmiseks ei kirjutata enam kogu lisanduvat loogikat olemasolevasse, monoliitse arhitektuurivormiga tagarakendusse, vaid luuakse eraldiseisev rakendus, mida olemasolev tagarakendus saab tarbida kui ühte

teenust, mis võimaldab infovahetust teiste, platvormiga liidestatud ametkondade infosüsteemidega.

Antud arhitektuurivormi näol on tegemist mikroteenustel põhineva arhitektuuriga, mis on üks teenustel põhinevatest arhitektuurivormidest (*service-oriented architecture, SOA*) välja arenenud alaliike [11]. Kuigi loodaval rakendusel on mitmeid ühiseid jooni mikroteenustele omaste omadustega, nagu näiteks sõnumivahendusteenuse kasutamine, kindla ülesande täitmine, siis ei pea autor õigeks kasutada loodava komponendi puhul terminit „mikroteenus“, vaid pigem iseloomustada seda kui „makroteenust“, kuna antud bakalaureusetöö skoobi väliselt hakkab antud komponent täitma ka teisi funktsioone, mis on seotud rahvusvahelise suhtluskanaliga integreerimisega.

Komponendi arendamiseks valiti teenusepõhine arhitektuurivorm, kuna võimaldab koondada kogu rahvusvahelise suhtlusega seonduva äri- ja andmepöördusloogika omaette tagarakendusse ning vajadusel teostada sellele arendustöid eraldiseisvalt ülejäänud rakendusest. See tähendab, et loodava tagarakenduse versioonide uuendamisel ei pea Prokuratuuri infosüsteemi administraatorid seiskama kogu infosüsteemi tööd, vaid on võimalik ajutiselt peatada ainult rahvusvaheliste toimingute tegemine ning peale uuendustööde teostamist need uuesti aktiveerida. Lisaks annab see juurde ka paindlikkust, mis kellaaegadel on võimalik teostada uuendustöid, kuna infosüsteemi kasutajad teostavad oma tööülesannetest tulenevalt valdavalt siiski siseriiklikke menetlustoiminguid.

4 Tarkvarakomponendi arenduskäik

Käesolevas peatükis kirjeldatakse bakalaureusetöö raames valmiva tarkvarakomponendi arendusprotsessi, järgides loodava komponendi arhitektuurilist ülesehitust. Kuna tegemist on tundlikku informatsiooni töötleva rakendusega, siis ei tooda välja loodava komponendi kõiki detaile ning samuti ei kuulu avalikustamisele arendusprotsessi käigus kirjutatav programmikood.

4.1 Kasutatavad arendustarkvarad

Loodava tarkvarakomponendi arendamiseks on autor kasutusele võtnud alljärgnevad arvutitarkvarad. Tarkvarade valikud tehti tuginedes projekti vajadustele ja seda teostava ettevõtte poolt omatavatele tarkvaralitsentsidele ning isiklikele kasutuskogemustele.

1. .NET SDK [12] – tarkvara C# programmikoodi kompileerimiseks ja kompileeritud koodi käivitamiseks

2. Visual Studio 2022 Professional [13] – IDE rakenduse lähtekoodi kirjutamiseks ja silumiseks

3. SQL Server Management Studio [14] – IDE andmebaasiga seotud operatsioonide teostamiseks, näiteks andmete pärimine või andmebaasi ülesehituse kontrollimine

4. Notepad++ [15] – multifunktsionaalne vabavaraline tekstiredaktor, millega on võimalik hõlpsalt avada erinevates arvutikeeltes kirjutatud faile ning neid loetaval kujul esitada

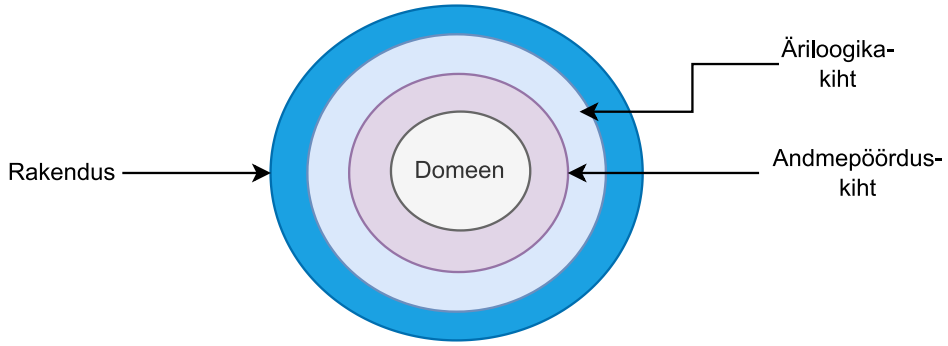
5. Docker Desktop [16] – võimaldab kasutajal oma arvutis virtualiseerida erinevaid teenuseid, näiteks lokaalne andmebaas

6. JetBrains Resharper [17] – lisamoodul Visual Studio arendustarkvarale, mis annab arendajale lisa-tagasisidet tema kirjutatava koodi kohta ning võimaldab hõlpsamini koodi refaktoreerida

Loetletud arendustarkvarad paigaldatakse ning arendustööd teostatakse läbi kasutades ettevõttepoolt tagatud Windows 10 operatsioonisüsteemiga arvutit.

4.2 Tarkvarakomponendi arhitektuur

Tarkvarakomponendi lähtekood on struktureeritud tuginedes *Domain-Driven Design* ehk rakenduse äriloogika põhikomponentideks – domeeniks – olevatel olemitel põhinevale arendusmustrile, järgides nn „sibulakujulist“ ehk kihilist arhitektuuri. Tegemist on arhitektuurivormiga, kus lähtekoodi põhiosad, mis moodustavad töötava rakenduse, on teineteisest eraldatud kihtidesse, millede loogikaklassid on koostatud olemi (või omavahel tihedalt seotud olemite) põhiselt. Tüüpiliselt on nendeks kihtideks andmemudeli kiht, loogika kiht ning rakenduslik kiht. Joonisel 6 on esitatud lihtsustatud kujul arendustööde raames valmiva tarkvara planeeritav arhitektuur.

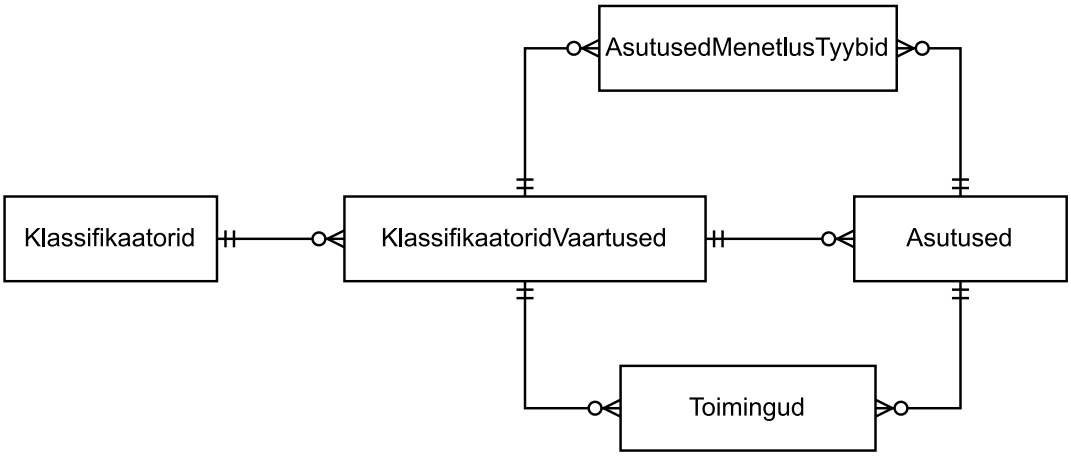


Joonis 6. Valmiva tarkvarakomponendi kavandatav arhitektuur

Lähtekoodi kirjutamisel üritatakse järgida Robert C. Martin'i raamatutes „Clean Code“ [18] ja „Clean Architecture“ [19] kirjeldatud arendusmustreid ja -printsiipe. Selle tulemusena loodab autor, et arendatav tarkvarakomponent on tulevikus hõlpsasti täiendatav ja hooldatav äriloogiliste nõudmiste muutumiste ning kasutatavate väliste tarkvarateekide uuendamise või väljavahetamise korral.

4.2.1 Andmeolemite kiht

Andmeolemite kiht kujutab endast andmemudelit ehk domeeni, mille olemitega teostatakse äriloogilisi protsesse. Järgnevalt on esitatud loodava rakenduse kontseptuaalne domeenimudel (joonis 7) ning kirjeldatud sellesse kuuluvate olemite funktsioone (tabel 1).



Joonis 7. Arendatava rakenduse olemi-suhte diagramm

Tabel 1. Domeenimudeli andmeolemite kirjeldused

Andmeolemi nimetus	**Andmeolemi kirjeldus**
Klassifikaatorid	Kindlate väärtuste ühine nimetaja või nende grupi nimetus, näiteks „riik“, „menetluse tüüp“
KlassifikaatoridVaartused	Klassifikaatorile vastavad väärtused, näiteks „Eesti“, „Euroopa uurimismäärus“
Asutused	Süsteemis registreeritud asutuste, kelle vahel toimub infovahetus, andmestik
AsutusedMenetlusTyybid	Süsteemis registreeritud asutuste seosed nende klassifikaatorite väärtustega, mis näitavad ära, milliseid menetluste tüüpe konkreetne asutus menetleb
Toimingud	Infosüsteemide vahel edastatav äriloogilise sisuga (s.t menetlustoimingute) andmestik

Rakenduse andmemudel on realiseeritud *Database first* põhimõttel ehk tuginedes andmebaasile. Rakenduse andmemudel on kirjeldatud päringukeeles SQL, mis omakorda on realiseeritud Microsoft SQL Server andmebaasina. Lokaalseteks arendustöödeks on

kasutusele võetud Docker keskkonnas konteinerina käivitatav virtuaalne andmebaasiserver, arenduskeskkonnas kasutatakse ettevõttepoolset andmebaasiserverit.

Andmemudeli muudatuste haldamiseks ehk versioneerimiseks on kasutusele võetud *snapshot'*id ehk tõmmised, mis genereeritakse arendaja poolt peale igat andmemudelisse tehtud muudatust. Nende põhjal on omakorda võimalik genereerida muudatuste- ehk migratsiooniskripte, mis viivad andmebaasis realiseeritud struktuuri vastavusse ühelt tõmmise versioonilt teisele uuendades.

4.2.2 Andmepöörduskiht

Andmepöörduskiht ehk *Data-Access Layer* koondab endasse kogu loogika, mis on seotud väliste teenuste (andmebaas, failiserver ja muud välised (veebi-)teenused) tarbimisega. Käesoleva töö skoobis on nendeks andmebaas, kus on realiseeritud rakenduse andmemudel ning hoitakse nendega seonduvaid andmeid, ning failiserver, kus on hoiustatud edastatavate andmetega seonduvad failid.

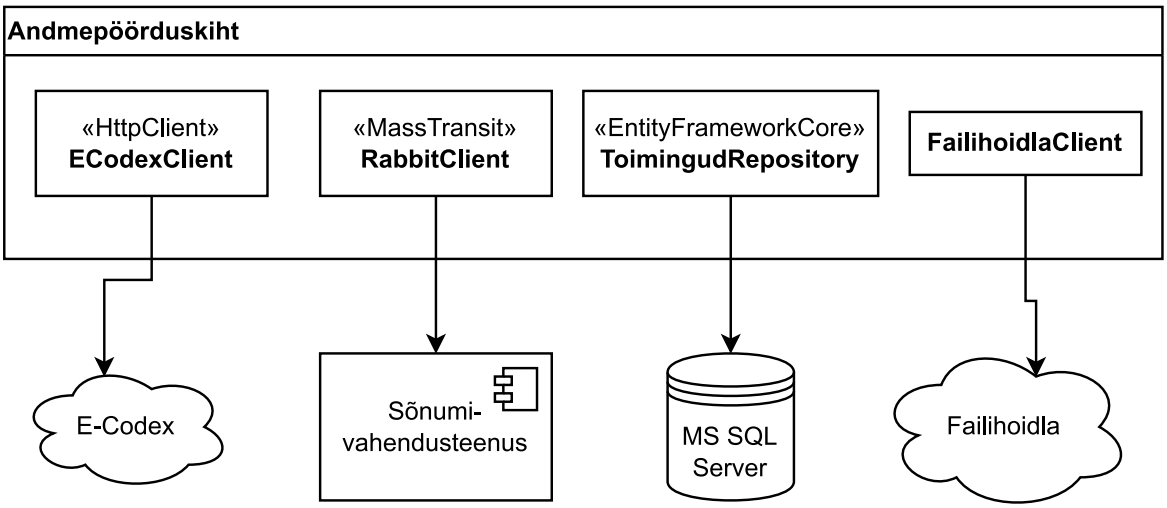
Selleks, et rakendusel oleks võimalik andmebaasiga liidestuda, on vajalik kasutusele võtta väline teek. Loodava rakenduse tarbeks on valitud selleks teegiks Entity Framework Core [20]. Tegemist on Microsoft poolt arendatud *object-relational mapping* ehk objekt-relatsioonvastenduse teegiga, mis võimaldab arendajal rakenduse koodi kirjutades kasutada andmebaasis realiseeritud andmeolemeid. Entity Framework Core näol on antud töö kirjutamise hetkel tegemist kõige populaarsema ORM tüüpi teegiga, mis on saadaval NuGet [21] teegihoidlas. Kuna loodaval rakendusel on lihtne andmemudel ning ei ole vajadust sooritada komplekssseid andmebaasipäringuid, siis osutus valik Entity Framework Core kasuks, kuna tänu kasutatavatele arendustööriistadele – võimalus genereerida andmemudel Entity Framework Core jaoks sobivasse C# koodi – saab seeläbi hõlpsalt loodud integratsioon rakenduse andmepöörduskihi ja andmebaasi vahel.

Andmeolemite ja andmebaasiga seonduva loogika ning äriloogika selgemaks omavaheliseks eraldamiseks jälgitakse *Repository pattern* arendusmustrit ehk on võetud kasutusele olemite põhised klassid, mille ülesandeks on konkreetse andmeolemiga seotud andmebaasipäringud. Kuna rakenduses on kasutusel ka Entity Framework Core teek, siis on antud otsuse näol tegemist valikuga, mille osas on palju vastakaid arvamusi. Peamine põhjus, miks sellele vastu ollakse, on see, et eelpoolmainitud tarkvarateek rakendab juba oma ülesehituses antud arendusmustrit. Samas põhjustab see autori hinnangul liiga

tugevat seost kasutatava teegi (ja seeläbi pöördumiste andmebaasi poole) ja äriloogika vahel. Eraldades need kaks, on esiteks tulevikus lihtsam antud tarkvarateek vahetada välja mõne teise samataolise vastu (mis antud hetkel on küll vähetõenäoline, arvestades andmemudeli lihtsust) aga peamise kasutegurina näeb töö autor koodi ühiktestimise lihtsustumist – kui andmepäringute loogika on eraldatud muust äriloogikast, on neid kahte võimalik testida nii sõltumatult teineteisest kui ka integreerituna.

Kuna rakenduse poolt töödeldavas andmestikus sisaldub ka faile, mida nende mahukuse tõttu ei ole reeglina mõistlik hoiustada andmebaasiserveris, siis on kasutusele võetud eraldiseisev failihoidla. Failihoidla näol on tegemist ettevõttesisese failiserveri-teenusega. Failihoidlaga ühendumiseks ning failide töötlemiseks on käesolevas kihis realiseeritud kõik CRUD-operatsioonid. Failide talletamisel failihoidlasse saadakse sealt failiviide, mille alusel on võimalik faili hiljem teenuse kaudu tarbida. Failiviited talletatakse andmebaasi koos ülejäänud andmestikuga.

Joonisel 8 on kujutatud andmepöörduskihi ülesehitus. Et vältida kordusi, on joonisele kandmata korduvad, samalaadsed loogikaklassid.



Joonis 8. Andmepöörduskihi lihtsustatud ülesehitus

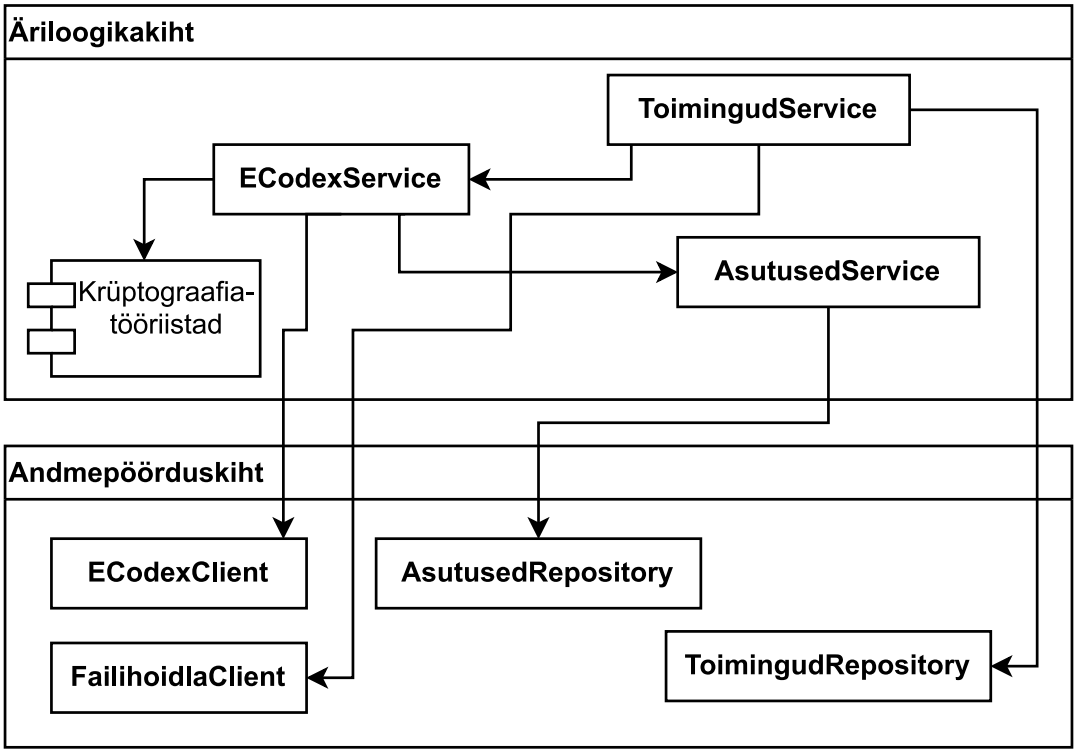
4.2.3 Äriloogikakiht

Äriloogikakihis on realiseeritud rakenduse äriloogilised nõudmised. Vastavalt peatükis 4.2 mainitud arendusmustrile, on äriloogika üles ehitatud vastavalt selle peamistele andmeolemitele.

Mõned näited realiseeritud äriloogilistest operatsioonidest:

1. Toimingute andmestiku pärimine ja salvestamine vastavatest hoidlatest
2. Toimingute andmestiku edastamine välisesse teenusesse
3. Asutuste andmestiku uuendamine välisest teenusest

Joonisel 9 on esitatud äriloogika kihi ülesehitus ning selle osade seosed andmepöörduskihiga.



Joonis 9. Äriloogikakihi ülesehitus

4.2.4 Rakenduse kiht

Rakenduse kiht on loodava tarkvarakomponendi kõige välisem kiht, kus on realiseeritud rakenduse käivitamisloogika ning selle muud sisemised äriloogilised (tausta-)protsessid, moodustades seeläbi käivitatava ja funktsionaalse tarkvara.

Selleks, et rakendusel oleks võimalik vastu võtta välismaa infosüsteemidelt saabuvaid andmeid, on vaja realiseerida API otspunkt, mis põhineb SOAP sõnumivahetuse protokollil. Kui vanemates .NET raamistike versioonides (enne .NET Core versioone välja antud .NET Framework nimetusega versioonid) olid antud protokolli kasutusele

võtmiseks vajalikud teegid sisseehitatud, siis uuemates .NET versioonides on SOAP tugi küllaltki puudulik. Rakenduse kasutatavas .NET versioonis, milleks on vastavalt mitte-funktsionaalsetele nõuetele valitud .NET 6 [12], ei ole sisseehitatud tuge antud protokollile ning vajalik on võtta kasutusele mõni väline teek. Enamlevinumad teegid selle tarbeks on CoreWCF [22] ja SoapCore [23], millest mõlemat katsetati ka käesoleva rakenduse arendusprotsessis.

CoreWCF on avatud lähtekoodiga .NET Foundation'i projekt, mille eesmärgiks on tuua uude .NET versiooni üle vanades .NET versioonides (.NET Framework aegsed) eksisteerinud SOAP funktsionaalsused. Paraku ei olnud antud teeki võimalik kasutusele võtta, kuna arendusfaasi hetkel ei olnud antud teegi arendamisel veel jõutud niikaugele, et oleks võimalik kasutada ka WS-Security spetsifikatsioonile [24] vastavaid krüptograafiafunktsioone.

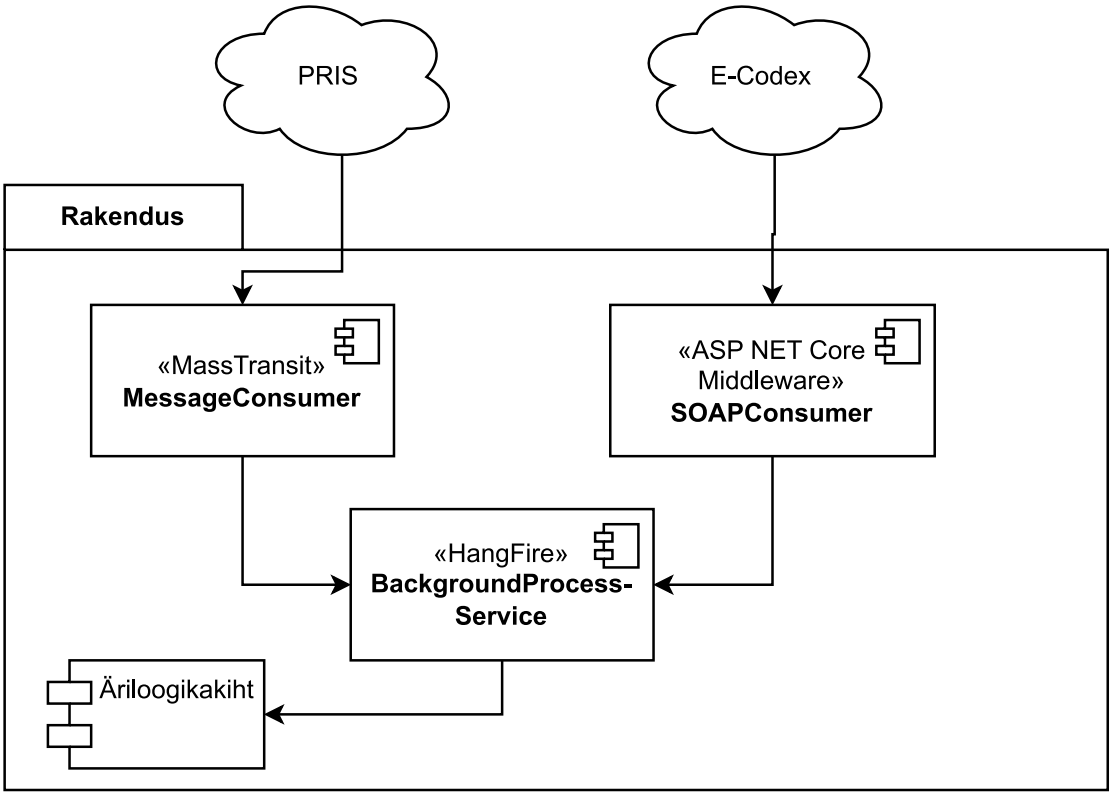
Teine levinud teek, SoapCore, on samamoodi avatud lähtekoodiga teek, ent sellel ei ole taga suurt organisatsiooni, vaid arendatakse üksikisikute poolt. Antud teegiga sai lihtsa vaevaga tekitada rakendusele vajaliku teenuse otspunkti, ent esmaseid testpäringuid tehes ilmnes, et teek ei oska töödelda *multi-part* ehk mitmest osast koosnevaid päringuid. Sarnaselt CoreWCF teegiga on jällegi puudu WS-Security tugi. Kuna SoapCore'ga oli võimalik tekitada rakendusele vajalik teenuse otspunkt, siis sai vastu võetud otsus jätta see selleks otstarbeks kasutusele ning luua ülejäänud loogika ise, tuginedes ASP.NET Core raamistiku pakutavale *middleware* [25] ehk vahelüli funktsionaalsusele.

Selleks, et rakendusel oleks võimalik Eesti poolselt infosüsteemilt, milleks on Prokuratuuri infosüsteem, võtta vastu andmeid ja muid päringuid, on vaja see ühendada nimetatud infosüsteemi kasutatava sõnumivahendusteenusega RabbitMQ [26]. Antud liidestus on mõistlik realiseerida kasutades mõnda olemasolevat tarkvarateeki. Antud rakenduse tarbeks sai valitud teek MassTransit [27], kuna see on käesoleva töö kirjutamise hetkel on üks populaarsemaid ning seda kasutab ka teine suhtluspool.

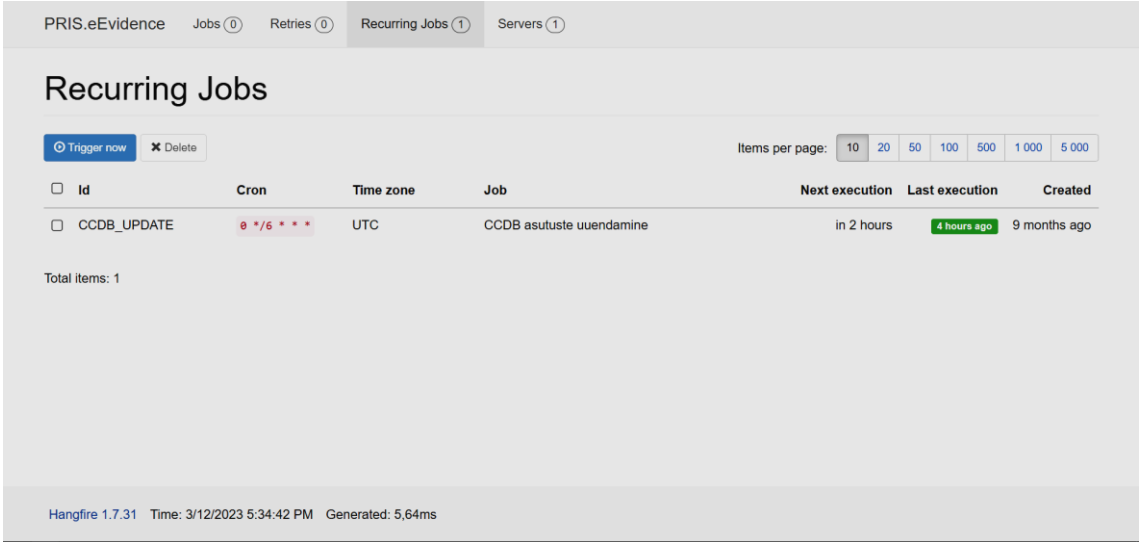
Rakenduses toimuvate taustaprotsesside, milleks on vastuvõetud andmete edastamine teisele süsteemile ning muud andmete sünkroniseerimisprotsessid, realiseerimiseks uuriti erinevaid võimalusi ASP.NET Core raamistikul ühekordsete ja korduvate taustaprotsesside loomiseks ja käivitamiseks. Selle tulemusel saab öelda, et antud eesmärki on võimalik saavutada nii raamistiku enda pakutavate vahenditega, luues kogu

vajaliku loogika ise, kui ka kasutades väliseid teeke, näiteks HangFire [28] või Quartz.NET [29]. Kuigi eelistatum variant oleks olnud vastav lahendus ise luua, langes valik lõpuks siiski välise teegi HangFire kasuks. Nimetatud teek pakub valmis lahendust nii erinevate protsesside käivitamiseks (s.h ka ajastatud käivitamist, mis on konfigureeritav CRON avaldiste [30] abil) kui ka seireks tänu sisseehitatud kasutajaliidesele.

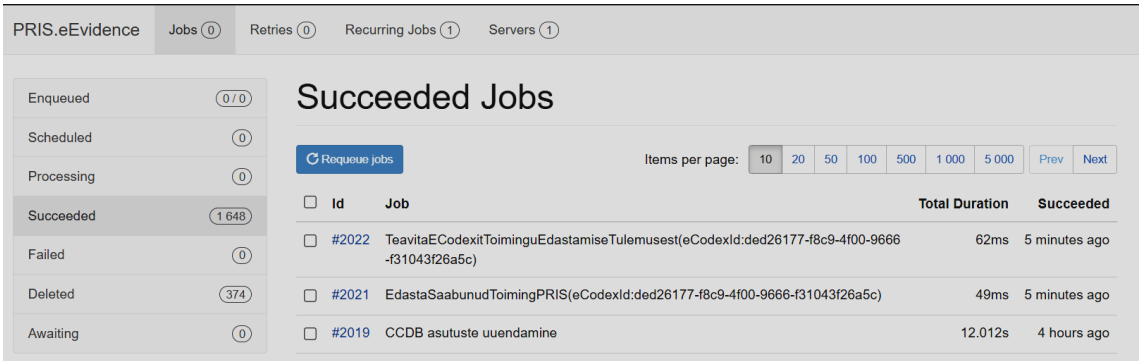
Järgnevatel joonistel on kujutatud rakendusliku kihi lihtsustatud ülesehitus (joonis 10) ning kasutatud tarkvarateegi – HangFire – poolt genereeritud kasutajaliides rakenduse äriloogiliste-protsesside haldamiseks (joonis 11, joonis 12).



Joonis 10. Rakendusliku kihi lihtsustatud ülesehitus



Joonis 11. Kuvatõmmis rakenduse ajastatud protsesside töölauavaatest



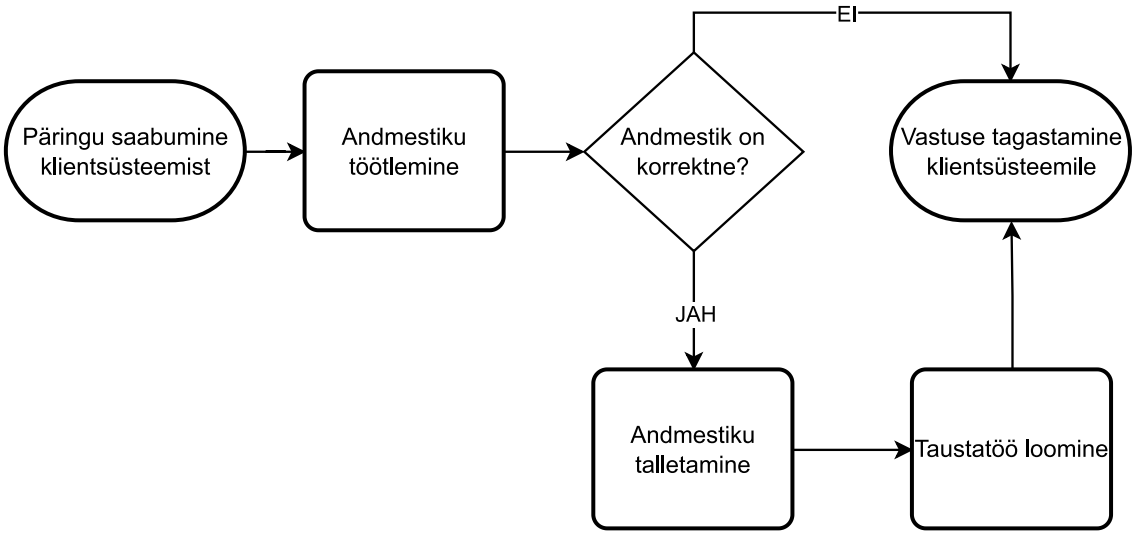
Joonis 12. Kuvatõmmis rakenduses loodud ühekordsete taustaprotsesside vaatest

4.3 Valminud rakenduse põhifunktsionaalsus

Käesoleva bakalaureusetöö skoobis põhjal arendati valmis tagarakendus, mille põhiline töövoog on järgmine:

1) Klientsüsteemist tehakse rakendusele päring, milles sisaldub edastatav andmestik
2) Rakendus töötleb saadud andmestiku ning talletab saadud informatsiooni vastavatesse andmehoidlatesse
3) Rakendus registreerib taustatöö, mille ülesandeks on saadud andmestiku edastamine vastavale asutusele
4) Rakendus tagastab klientsüsteemi päringule, kas andmete vastuvõtmine õnnestus

Joonisel 13 on kujutatud valminud tagarakenduse põhitöövoog.

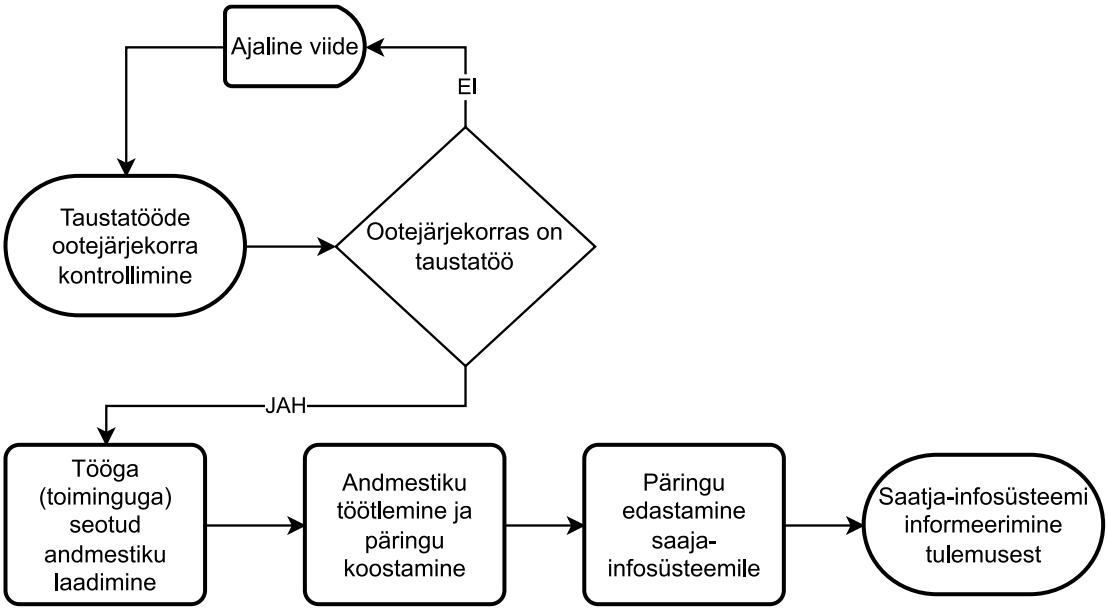


Joonis 13. Rakendusele tehtavate päringute üldine töötlemisloogika

Samal ajal jookseb rakenduses taustaprotsess, mis perioodiliselt kontrollib genereeritud taustatööde nimekirja:

1) Taustaprotsess leiab järjekorras registreeritud taustatöö
2) Taustaprotsess loeb andmehoidlast sisse edastatava toimingu andmestiku
3) Taustaprotsess töötleb andmestiku ning koostab ja edastab päringu vastavale klientsüsteemile
4) Taustaprotsess teavitab toimingu edastanud infosüsteemi andmete üleandmise (eba)õnnestumisest

Selgitav illustratsioon antud protsessi tööst on kujutatud joonisel 14.



Joonis 14. Rakenduse andmestiku edastamise taustaprotsessi põhiloogika

Võrreldes hetkel kliendi poolt kasutusel oleva andmete edastamise meetodiga, milleks on dokumentide edastamine paberkandjal, on tänu loodud tagarakendusele andmete edastamine kahe asutuse (infosüsteemide) vahel vaieldamatult kiirem, jäädes arenduskeskkonnas ajaliselt alla ühe minuti, sõltuvalt edastatavate andmete, eelkõige manustena kaasapandud failide, hulgast ja kogumahust.

4.4 Rakenduse testimine

Arendustööde tulemusena valminud rakenduse lähtekoodi testimiseks, nii arendustööde käigus kui ka peale käesoleva töö raames kirjeldatud skoobi valmimist, kasutati nii automaatseid kui manuaalseid testimisviise.

4.4.1 Automaattestid

Automaattestid on mõnes programmeerimiskeeles (käesoleva töö kontekstis C#) kirjutatud testid, mida on võimalik arendajal käivitada arendusprotsessi käigus käsitsi läbi IDE kui ka automaatselt, seadistades nende käivitamine rakenduse ehitusprotsessi ühe osana.

Rakenduse automaattestid on kirjutatud NUnit testimisraamistikus. Antud raamistik valiti autori ja meeskonna eelneva kokkupuute põhjal. Enne testprojektide loomist uuriti ka teisi

populaarsemaid testimisraamistikke nagu näiteks MSTest ja xUnit, aga ei leitud erinevusi, mis käesoleva rakenduse kontekstis oleksid kandnud määravat rolli.

Kirjutatud tarkvarakoodi testimiseks kirjutati kaht eri liiki automaatteste:

1. Ühiktestid – testitakse koodi ühe spetsiifilise osa ehk ühiku – konkreetse meetodi – vastamist antud meetodile seotud ülesande täitmisega, välised sõltuvused on *mocked* ehk imiteeritud

2. Integratsioonitestid – testitakse kindla lähtekoodi osa toimimist koos erinevate väliste sõltuvustega, näiteks andmebaas või muu väline API

Automaattestide kood paigutati eraldiseisvatesse projektifailidesse, struktureerides need analoogselt loodud rakenduse lähtekoodiga.

Testmeetodite nimetamisel järgiti Microsofti poolt kirjeldatud nimetamispõhimõtteid, kus testmeetodi nimi koosneb kolmest alakriipsuga eraldatud osast, milleks on testitava meetodi nimi, stsenaarium ja oodatud tulemus, näiteks testmeetod nimetusega „GetFile_ValidFileId_ReturnsFileWithSameId“ testib, et faili pärimisel korrektses vormingus failiviitega tagastab faili, mille viide vastab sisendis antud viitele.

Testmeetodite koostamisel järgiti AAA ehk *Arrange, Act, Assert* mustrit. Mustri nimetus ja põhimõte tuleneb sellest, et meetodi sisu on jaotatud kolmeks osaks, mis, tuginedes eelpoolmainitud testmeetodile, on järgmised:

1) *Arrange* ehk ülesseadmine – selles faasis luuakse vajalikud eeltingimused meetodi testimiseks, näitemeetodi puhul luuakse imitatsioon failihoidlast, kus paikneb kindlaksmääratud viitega fail

2) *Act* ehk tegevus – testitava meetodi väljakutsumine ning vajadusel tagastatavate andmete hoiustamine muutujatesse, näitemeetodi puhul kutsutakse välja failihoidla CRUD loogikaklass ning päritakse läbi selle eelpoolmääratud viitega fail

3) *Assert* ehk valideerimine – kontrollitakse tegevusfaasi tulemusena toimunut, mille järel loetakse testmeetod kas õnnestunuks või mitte, näitemeetodi puhul kontrollitakse, et tagastatud fail oleks nõutud failiviitega

4.4.2 Manuaalne testimine

Rakenduse manuaalset testimist viidi läbi inimese poolt (arendaja või muu projektiga seotud isik), kes katsetas toimingute edastamist kahe infosüsteemi vahel. Selle jaoks loodi kummaski infosüsteemis menetlustoiminguid, järgides konkreetsetele toimingutele seatud ärilisi nõudmisi, ning edastati neid teise infosüsteemi. Selle tegevuse käigus jälgiti ühes infosüsteemis sisestatud andmete õiget kuvamist ka teises infosüsteemis ning rakenduse logidesse tekkivat informatsiooni, mis võivad viidata võimalikele vigadele.

4.5 Rakenduse turve

Arendatud rakenduse näol on tegemist eraldiseisva tagarakendusega, mis, kommunikeerides infosüsteemi tagarakendusega, võimaldab selle kasutajatel tarbida teatuid teenuseid. See tähendab, et infosüsteemi kasutajad ei puutu kunagi otseselt kokku arendatud rakendusega ning arendatud rakenduse ainsateks otsesteks kasutajateks saavad olema rakenduseadministraatorid, kellele jääb ligipääs erinevateks haldustegevusteks.

See võimaldab arendatud rakenduse paigaldada avalikust võrgust eraldiseisvasse võrgutaristusse, millele ligipääs on vaid kontrollitud ja dokumenteeritud isikutel. Antud isikute ülesandeks jääb ka vastava taristu seadistamine selliselt, et rakendusel oleks võimalik teostada vajalikke äriloogilisi operatsioone, see tähendab ligipääs peatükis 3 kirjeldatud lüüsile kui ka sõnumivahendusteenusele ja peatükis 4.2.2 kirjeldatud andmete hoidmiseks kasutatavatele välistele sõltuvustele, vastavalt ettevõtte infoturbeosakonna poolt paika pandud reeglistikule.

4.6 Rakenduse lansseerimine

Käesoleva bakalaureusetöö kirjutamise hetkel on antud projekt veel arendusfaasis, mistõttu antud töö skoobi raames arendatud rakendus veel lõpp-kasutajateni jõudnud ei ole.

Arendusfaasis on rakenduse jooksutamiseks kaks võimalust:

1) Lokaalses arenduskeskkonnas on rakendus jooksutatav tavalise konsoolirakendusena, kasutades selleks IDE või käivitades kompileerimise tulemusena väljastatud EXE-faili otse käsurealt.

2) Arendusserveris, kuhu paigaldatakse jooksvalt valmivaid arendustöid, on rakendus käivitatav IIS veebiserverina. Peale veebiserveri esmast seadistust on arendajal peatükis 4.1 mainitud IDE abil võimalik genereerida skript ning kompileerida valmis pakett, mis paigaldatakse vastavasse IIS veebiserveri kausta.

Vastavalt peatükis 3.2.2 kirjeldatud etteantud mittefunktsionaalsetele nõuetele peab valmiv rakendus olema käivitatav kasutades ka konteinerlahendusi. Nagu eelnevalt mainitud, siis antud hetkel ei ole loodava tarkvara arendusprotsessis veel jõutud esmase versiooni valmimise ja tarnimiseni, ent on loodud võimalused arendusfaasi hetkeseisu põhjal pakendada rakendus Docker tehnoloogial põhinevasse konteinerisse. Vastav võimalus on loodud kasutades Dockerfile skriptifaili, mis loeb sisse erinevad lähtekoodi kuuluvate projektide osad ning kompileerib ja pakendab nendest valmis konteinerfaili, mida arendaja saab soovi korral jooksutada ka enda isiklikul tööarvutil, kasutades selleks Docker Desktop [16] tarkvara.

5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli arendada välja vajalik tarkvarakomponent Prokuratuuri infosüsteemile võimaldamaks rahvusvaheliste kriminaalmenetlustega seotud menetlustoimingute digitaalset edastamist ja vastuvõtmist teiste Euroopa Liidu liikmesriikidega kasutades selleks keskset platvormi e-Codex.

Bakalaureusetöö analüütilises pooles anti ülevaade praegusest kriminaalmenetluste läbiviimise protsessist nii siseriiklike kui ka rahvusvaheliste kriminaalmenetluste näol, analüüsiti viimase puudujääke ning kirjeldati ära funktsionaalsed ja mitte-funktsionaalsed nõuded tehtavatele arendustöödele võimaldamaks digitaalset andmete vahetamist.

Bakalaureusetöö põhiosa praktilise poolena valmis tarkvaraline komponent Prokuratuuri infosüsteemi tagarakendusele, mis võimaldab infovahetust eelpoolmainitud Euroopa Liidu platvormiga. Antud sisus on ära selgitatud tehtud arendustööde käik järgides loodud tarkvarakomponendi lähtekoodi arhitektuuri ning valminud rakenduse testimist ning käivitusvõimalusi.

Tänu töö tulemusele valminud komponendile on oodata rahvusvaheliste kriminaalmenetluste läbiviimisprotsessi olulist kiirenemist, seda eelkõige tänu menetluse teise osapoole – välisriigi uurimisasutusega – kiirenenud andmevahetuse tõttu.

Kasutatud kirjandus

[1] Riigi Infosüsteemi Amet, „X-tee,“ [Võrgumaterjal]. Kättesaadav: https://x-tee.ee/. [Kasutatud 28 10 2023].

[2] „Digitaalse menetluse tulevik,“ Riigiprokuratuur, [Võrgumaterjal]. Kättesaadav: https://aastaraamat.prokuratuur.ee/prokuratuuri-aastaraamat-2018/digitaalse-menetluse-tulevik. [Kasutatud 28 10 2022].

[3] „Kriminaalmenetluse seadustik,“ Riigi Teataja, [Võrgumaterjal]. Kättesaadav: https://www.riigiteataja.ee/akt/106012016019. [Kasutatud 28 10 2022].

[4] „Prokuratuuriseadus,“ Riigi Teataja, [Võrgumaterjal]. Kättesaadav: https://www.riigiteataja.ee/akt/102062020008. [Kasutatud 28 10 2022].

[5] „e-Codex,“ [Võrgumaterjal]. Kättesaadav: https://www.e-codex.eu/. [Kasutatud 3 11 2022].

[6] „e-Codex / Technical infrastructure,“ [Võrgumaterjal]. Kättesaadav: https://www.e-codex.eu/sites/default/files/inline-images/e-CODEX_Connector-Gateway%20infographic_1_0.png. [Kasutatud 4 11 2022].

[7] Microsoft Corporation, „C# | Modern, open-source programming language for .NET,“ [Võrgumaterjal]. Kättesaadav: https://dotnet.microsoft.com/en-us/languages/csharp. [Kasutatud 09 12 2022].

[8] Microsoft Corporation, „.NET,“ [Võrgumaterjal]. Kättesaadav: https://dotnet.microsoft.com. [Kasutatud 07 12 2022].

[9] Docker Inc., „Docker Docs,“ [Võrgumaterjal]. Kättesaadav: https://docs.docker.com/get-started/. [Kasutatud 25 03 2023].

[10] The Linux Foundation, „Kubernetes,“ [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/. [Kasutatud 14 03 2023].

[11] S. Newman, Building Microservices, O'Reilly Media Inc., 2021.

[12] Microsoft Corporation, „.NET 6.0,“ [Võrgumaterjal]. Kättesaadav: https://dotnet.microsoft.com/en-us/download/dotnet/6.0. [Kasutatud 01 12 2022].

[13] Microsoft Corporation, „Visual Studio 2022 IDE - Programming Tool for Software Developers,“ [Võrgumaterjal]. Kättesaadav: https://visualstudio.microsoft.com/vs/. [Kasutatud 24 2 2023].

[14] Microsoft Corporation, „SQL Server Management Studio,“ [Võrgumaterjal]. Kättesaadav: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms. [Kasutatud 24 2 2023].

[15] D. Ho, „Notepad++,“ [Võrgumaterjal]. Kättesaadav: https://notepad-plus-plus.org/. [Kasutatud 24 2 2023].

[16] Docker Inc., „Docker Desktop,“ [Võrgumaterjal]. Kättesaadav: https://www.docker.com/products/docker-desktop/. [Kasutatud 24 2 2023].

[17] JetBrains s.r.o., „ReSharper: The Visual Studio Extension for .NET Developers,“ [Võrgumaterjal]. Kättesaadav: https://www.jetbrains.com/resharper/. [Kasutatud 24 2 2023].

[18] R. C. Martin, Clean Code, Pearson Eductation, Inc., 2009.

[19] R. C. Martin, Clean Architecture, Pearson Education, Inc., 2018.

[20] Microsoft Corporation, „Nuget Gallery | Entity Framework Core 7.0.3,“ [Võrgumaterjal]. Kättesaadav: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore. [Kasutatud 20 2 2023].

[21] Microsoft Corporation, „Nuget Gallery,“ [Võrgumaterjal]. Kättesaadav: https://www.nuget.org/. [Kasutatud 2023 2 18].

[22] .NET Foundation, „Github - CoreWCF,“ [Võrgumaterjal]. Kättesaadav: https://github.com/CoreWCF/CoreWCF. [Kasutatud 01 12 2022].

[23] Digital Design, „GitHub - DigDes/SoapCore,“ [Võrgumaterjal]. Kättesaadav: https://github.com/DigDes/SoapCore. [Kasutatud 01 12 2022].

[24] IBM Corporation, „WS-Security - IBM Documentation,“ [Võrgumaterjal]. Kättesaadav: https://www.ibm.com/docs/en/app-connect/11.0.0?topic=security-ws. [Kasutatud 01 12 2022].

[25] R. Anderson ja S. Smith, „ASP.NET Core Middleware | Microsoft Learn,“ [Võrgumaterjal]. Kättesaadav: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/. [Kasutatud 28 02 2023].

[26] VMWare Inc., „RabbitMQ,“ [Võrgumaterjal]. Kättesaadav: https://www.rabbitmq.com. [Kasutatud 22 2 2023].

[27] C. Patterson, „MassTransit,“ [Võrgumaterjal]. Kättesaadav: https://masstransit.io/. [Kasutatud 22 02 2023].

[28] HangFire OÜ, „GitHub - HangFireIO/HangFire,“ [Võrgumaterjal]. Kättesaadav: https://github.com/HangfireIO/Hangfire. [Kasutatud 1 3 2022].

[29] M. Lahma, „Quartz.NET,“ [Võrgumaterjal]. Kättesaadav: https://www.quartz-scheduler.net/. [Kasutatud 1 3 2023].

[30] Oracle, „Cron Expressions,“ [Võrgumaterjal]. Kättesaadav: https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm. [Kasutatud 1 3 2023].

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

Mina, Kenert Vaino

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Rahvusvahelise menetlusprotsessi digitaliseerimine Prokuratuuri infosüsteemis“, mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

12.05.2023

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