TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tanel Joosep 176611

SEKUNDAARSETE MEDITSIINIANDMETE HAJUTATUD SÜSTEEMI ARHITEKTUUR JA DISAIN

Magistritöö

Juhendaja: Gunnar Piho
PhD

Tallinn 2020

Autorideklaratsioon

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

Autor: Tanel Joosep

13.05.2020

Annotatsioon

Käesoleva magistritöö eesmärgiks on välja pakkuda omapoolne andmevahetusplatvormi lahendus sekundaarsete andmete pärimiseks ja analüüsimiseks lähtudes OPEN-projektis tõstatatud probleemist.

Töös käsitletavad probleemide hulka kuuluvad arhitektuurimudeli loomine, lõpp-lahenduse komponentide valik ning väljapakutud rakenduse analüüsimine.

Töö tulemuseks on loodud arhitektuur ja disainitud võimalik lahendus (*proof-of-concept*) andmete turvaliseks vahetamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, 4 suuremat peatükki, 14 joonist, 5 tabelit.

Abstract

Architecture and design of a distributed system for secondary medical data

Secondary usage of clinical medical data helps to increase overall medical healthcare, reduce healthcare costs and carry out better targeted research. For example, secondary usage of medical data analysis has helped to predict diseases and patient's length of stay in the hospital.

Unfortunately, medical data is dispersed between different healthcare institutions. At the moment there is no good way to gather that kind of personal data from different medical institutions due to the lack of technical solutions and legal aspects.

The aim of this master's thesis is to design a secure distributed data exchange concept for requesting and analysing secondary medical data from different institutions. The problem was first introduced in the „*Open pseudonymised health data and distributed computing e- infrastructure for medical science and clinical practice"* back in 2017.

Problems addressed in the work include the creation of an architectural model, the selection of the components for the final solution and the analysis of the proposed application.

As a result of the work, an architecture has been created and a possible solution (*proof-of-work*) for secure data exchange has been designed.

The thesis is in Estonian and contains 46 pages of text, 4 bigger chapters, 14 figures and 5 tables.

Lühendite ja mõistete sõnastik

SOA	*Service-Oriented Architecture,* teenusorienteeritud arhitektuur
HL7	*Health Level Seven*, rahvusvaheline standard meditsiiniandmete edastamiseks.
EHR	*Electornic Health Record,* elektrooniline tervisekaart
FHIR	*Fast Healthcare Interoperability Resource,* kiire tervishoiuteenuste koostalitlusvõime ressursi protokoll
GDPR	*General Data Protection Regulation,* isikuandmete kaitse üldmäärus
POC	*Proof Of Concept*, kontseptsiooni tõestus.
ADL	*Archetype Definition Language,* arhetüüpide definitsiooni keel.
JSON	*JavaScript Object Notation*, JavaScripti keele objekti kirjeldus.
XML	*Extensible Markup language,* laiendatav märgistuskeel.
XHTML	*Extensible HyperText Markup Language,* veebilehtede loomiseks kasutatav keel. HTML keel, mis on teostatud XML-is.
URL	*Uniform Resource Locator*, üldine infoallika asukohamääraja.
API	*Application Programming Interface,*
AQL	*Archetype Query Language,* arhetüüpide pärimise keel
DSTU	*Draft Standard for Trial Use,* spetsifikatsioon, mis võimaldab rakendajatel standardit testida.
CEF	*Connecting Europe Facility*, Euroopa ühendatud keskus.
P2P	*Peer-to-peer*, võrdõigusvõrk, P2P-võrk, võrgusõlmede kogum.
IPFS	*InterPlanetary File System*, P2P-võrk failide jagamiseks.
cURL	*Client Uniform Resource Locator*, kliendi üldine infoallika asukohamääraja.

TCP	*Transmissioon Control Protocol*, edastusohje protokoll.
GNU	*GNU's Not Unix,* vaba tarkvara.
GPG	*GNU Privacy Guard*, vabavaraline krüptograafia raamistik.

Sisukord

Tabelite loetelu

Jooniste loetelu

1 Sissejuhatus

1.1 Taust

Meditsiinivaldkonnas räägitakse andmete plahvatuslikust kasvust, mille all peetakse silmas kõikvõimalike patsiendiga seotud kliiniliste andmete dokumenteerimist. Kliinilise informatsiooni teisene kasutamine aitab tagada kõrgetasemelist tervishoiuteenust, vähendada tervishoiukulusid ja läbi viia kliinilisi uuringuid.

Näiteks on võimaldanud sekundaarsete andmete analüüs parendada tervishoiuteenuse kvaliteeti ennustades patsiendi hospitaliseerimise pikkust ja parandada infektsioonikontrolli. Andmeid on kasutatud ka haiguste ennetamiseks ja inimeste kliinilistesse uuringutesse värbamiseks. Seega on informatsiooni teisesest kasutamisest kasu nii rahvatervise ametnikele, teadlastele, arstidele kui ka tööstusele üldisemalt.

Sekundaarsete andmete adekvaatsel kogumisel on siiski mitmeid takistusi, mille hulka kuuluvad näiteks andmete kvaliteet ja andmete hajutatus erinevate andmekogujate (haiglad, peremeditsiini keskused jm) vahel ning asutuste vahelised juriidilised kokkulepped. Oluliseks väljakutseks on ka üldstandardite, tõestatud metodoloogiliste raamistike puudumine ja turvalisuse tagamine. [1] [2]

1.2 Probleemi kirjeldus

2017. aastal algatati Euroopa ülikoolide (Rostock, Lübeck, TalTech) ja ettevõtete (IBM, Fraunhofer, jt) konsortsiumi poolt OPEN-projekt (*Open pseudonymised health data and distributed computing e-infrastructure for medical science and clinical practice).* Projekti eesmärk on luua integreeritud, turvaline ja paindlik avatud e-infrastruktuur teiseste meditsiiniandmete käsitlemiseks ning anda inimestele kontroll nende terviseandmete terves Euroopas. Antud projekti puhul lähtutakse faktist, et praegu on suur hulk meditsiiniandmeid hajutatud erinevate andmekogujate vahel. Muuhulgas koguvad andmeid eraldi haiglad, perearstikeskused, meditsiinilaborid, teadusinstitutsioonid,

riigiinstitutsioonid ja isegi mobiilirakendused. Projektis on olulisel kohal ka andmete turvalisus, sest tegemist on delikaatsete andmetega. [3]

Seni kasutusel olevad lahendused keskenduvad peamiselt kontseptsioonile, kus andmete liigutamisel ja töötlemisel on osakaal kolmandatel osapooltel, näiteks suurkorporatsioonidel, sh Microsoft või Google. Delikaatsete andmete korral ei saa täielikult usaldada keskseid teenusepakkujaid. Selline lähenemine omab endas turvariski, sest usaldus välise teenuse serveri või rakenduse üle võib olla haavatav rünnakutele. [4] See viib olukorrani, kus meditsiiniasutused ei oma enam patsiendi andmete üle kontrolli ega vastutust.

OPEN-projektis tõstatatud vajadus hajusa ja turvalise andmevahetusplatvormi loomiseks, mis ühendaks endas erinevaid meditsiiniasutusi, puudub siiani tarkvaraline lahendus.

1.3 Eesmärk

Käesoleva töö eesmärk on luua hajus ja turvaline andmevahetusplatvormi lahendus sekundaarsete andmete pärimiseks ja analüüsimiseks lähtudes OPEN-projektis tõstatatud probleemist. Töö käigus esitletakse võimalikku lahenduse arhitektuuri ja disainitakse esmane lahendus (*proof-of-concept*), mis aitab teostada andmepäringuid nii, et andmed paiknevad erinevates asutustes. Muuhulgas ei liigutata andmeid ühest serverist teise, vaid liiguvad vahepealsed arvutused, mille põhjal ei ole võimalik tegelikke andmeid tuvastada. Ühelt poolt aitab antud meetod tagada delikaatsete andmete privaatsuse, sest kogu info jõuab kolmanda osapooleni peidetud kujul. Teisalt on kogu saadav info paikapidav edasiseks andmetöötluseks.
Töö lahenduskäigus on olulised viis alapunkti, mida võib pidada lahenduse nõueteks:

- Olemas peab olema süsteem, millega kliendid saavad liituda;
- Andmepäringute tegemine liitunud klientide vahel. Kliendid tähendavad antud töö raames asutusi, mis omavad meditsiinilisi andmeid;
- Tagastustulemuste hajutamine;
- Päringu standardite sidumine;
- Turvaline andmeedastus osapoolte vahel.

1.4 Ülevaade tööst

Lõputöö on jagatud neljaks suuremaks peatükiks, millest käesolev peatükk kirjeldab töö tausta ning kirjeldab probleemi, mida antud töös püütakse lahendada.

Teises osas kirjeldatakse töö metoodikat, mille käigus tuuakse välja töö kesksed objektid, võimalikud tööriistad probleemi käsitlemiseks ning valitakse meetodid töö protsessi koostamiseks.

Kolmandas peatükis esitab autor nii teoreetilise kui ka praktilise lahenduse tõstatatud probleemile.

Neljandas peatükis analüüsib autor töö tulemusi, tuues välja saadud lahenduse tugevad küljed ning töö puudujäägid. Samuti kirjeldatakse lühidalt alternatiivseid võimalusi probleemi lahendamiseks ning antakse ülevaade edasistest plaanidest.

2 Metoodika

Lõputöö väljund on eelneva analüüsi põhjal loodud meditsiiniandmete pärimise platvormi arhitektuur ning esialgne prototüüp.

2.1 Objekt

Käesolevas peatükis on täpsemalt kirjeldatud OPEN-projekti üldine struktuur ja funktsionaalseid ning mittefunktsionaalseid nõudeid, mis määravad antud töö praktilise osa raamistiku.

Probleemi lahendamisel tuleb arvestada ka meditsiiniasutustes kasutusel olevate erinevate standarditega. Nendeks võivad olla kohandatud lahendused, mis on loodud eritellimusel kolmandatelt osapooltelt või näiteks DICOM (*Digital Imaging and Communications in Medicine*), mis on peamiselt kasutusel pildiandmete edastamisel. Antud töös keskendub autor kahele enamlevinud avatud standardile – *openEHR* ja *HL7/FHIR.*

2.1.1 OPEN-projekti üldine struktuur

OPEN-projekti lahendus sisaldab kolme osa, mis on jagatud kasutajalugudeks:

a. Kodaniku vaade - tavakodaniku jaoks loodud vaade. Iga inimene saab enda andmeid kontrollida ja pärida erinevatest institutsioonidest.

b. Teadlase vaade, mis on mõeldud teadusuuringute läbiviimiseks teadlastele, kes ei tööta meditsiiniasutuses.

c. Arsti vaade, mis on mõeldud praktiseerivatele arstidele saamaks infot erinevate diagnooside ja nende puhuselt kasutatud raviviiside kohta.

Projekti tehniline lahendus baseerub *CEF Building Block* [5] lahendustel, mis püüavad lahendada identifitseerimis- ja kasutajaõigsuse komponente.

Antud plokkidest soovitakse kasutada peamiselt kaht osa:

1. *CEF eID* [6] plokk – võimaldab piiriülest autentimist turvalisel ja usaldusväärsel viisil, muutes riiklikud elektroonilised identifitseerimissüsteemid koostalitusvõimelisteks. Lahendus võimaldab ühe liikmesriigi kodanikel, teadlastel ja arstidel pääseda ligi e-infrastruktuurile, kasutades omaenda elektroonilisi identifitseerimise vahendeid, olgu selleks ID-kaart või mõni muu sarnane teenus. [3]

2. *CEF eSignature* [7] plokk – hõlbustab e-allkirjade vastastikust tunnustamist ja piiriülest koostalitlusvõimet liikmesriikide vahel. Selle abil on haldusasutustel ja ettevõtetel usaldus, et teenuse kasutajad on need, kes nad väidavad end olevat. [3]

Lisaks on OPEN-projektis välja pakutud veel *eInvoicing, eTranslation* ning *Human Interface* plokke, kuid nende praktiline väärtus ja vajalikkus peaks selguma projekti hilisemates etappides. CEF plokid baseeruvad peamiselt e-SENS teenustel. See tähendab, et kõiki teenuseid kasutatakse Euroopa reglementeeritud teenuste järgi.

Järgmiseks pakutakse välja terviseandmete semantiline koostalitlusvõime mudel. Selle eesmärk on lahendada probleem erinevate hajutatud terviseandmeallikate vahel, ühtlustades hajutatud andmeallikad ning muutes need ühtseks tervikuks. [3]

Edasi, tuuakse välja võimalused, kuidas lahendada turvalisuse aspekt. Eesmärk on tagada privaatsus- ja turvameetmed, mis takistaksid meditsiiniliste ja isikuandmete kasutamist väljaspool lubatud õigusraamistikku. Need meetmed ei tohiks süsteemi tõhusat kasutamist takistada.

Nendeks meetmeteks on:

a. Süsteemi lubatud kasutamist reguleerivad õiguslikud meetmed, mille väärkasutamisel on juriidilised karistused.
b. Protseduurilised ja arhitektuurilised meetmed, mis piiravad süsteemiga interaktsiooni primitiivide kasutamist.
c. Infrastruktuurimeetmed, mis keelavad, katkestavad ja kontrollivad peidetud toiminguid.
d. Täiustatud krüptograafilised meetmed, näiteks homomorfne krüptimine jms.

Tasub märkida, et punkti (d) osas öeldakse, et see on teadusuuringute seisukohast huvitav ja paljulubav, kuid mille töökindlust ja küpsust ei saa pidada iseenesestmõistetavaks. [3] Seega keskendub esildis peamiselt esimesele kolme punktile.

Lõpetuseks pakutakse välja võimalus, kuidas andmeid konsolideerida analüütilistel eesmärkidel. Siinkohal on eesmärk meditsiiniandmete koondamine, mis loob võimaluse mitmeotstarbeliselt analüüsida sekundaarseid terviseandmeid. Peamiseks takistuseks tuuakse asjaolu, et andmed, mis saadetakse, ei tohi originaalkujul asutustest väljuda. See tähendab seda, et andmed edastatakse juba peidetud kujul kolmandatele osapooltele. Selline lähenemine tagab usalduse osapoolte vahel, kus andmelekked ei ole võimalikud.

Tuuakse välja kaks võimalikku meetodit [3]:

1. Meetod baseerub eelnevalt arvutatud statistika põhjal, mis on ühendatud kõigi vastutavate töötlejate vahel. Meetod võimaldab mitmeid stohhastiliste omadustega lineaarseid analüüsimeetodeid. Samuti võimaldab kasutada andmete individuaalset varieeruvust ja ei piira valimi suurust ega vähenda vabadusastet. Funktsionaalsus võimaldab tõhusat andmeanalüüsi, mis säilitab haiglate töötlemata andmetes sisalduva teabe rikkuse, ilma et lõpptarbijale oleks võimalik individuaalset andmekirjet avaldada. Need meetodid on piiratud eelnevalt määratletud lineaarsete, stohhastiliste meetoditega.
2. Meetod baseerub viisile, kus esialgsete andmete põhjal koondatakse andmed asutuste tasandil vastavatesse rühmadesse ja seejärel kasutatakse neid asutuse tasandil või laiemalt ühtse andmekogumina. Päringumootor tagab sobitatud rühma minimaalse suuruse ja sisaldab eelnevalt määratletud läve, mis sõltub valimi suurusest ja rühmasisesest variatsioonist. Päringumootor väljastab andmed ainult siis, kui minimaalsed kriteeriumid on täidetud (näiteks andmehulga suurus on suurem kui miinimumlävi või rühma varieeruvus on väiksem kui eelnevalt seatud kriteeriumid). Süsteem määratleb eelnevalt sobivate muutujate kompleti, näiteks sugu ja vanus. Lõppkasutaja saab määratleda ülejäänud sobivad muutujad seni, kuni rühma suurus ja varieeruvus vastavad lävekriteeriumitele. Lõppkasutajal on teatav kontroll andmete grupeerimise üle, valides sobitamiseks kasutatavad muutujad. Selle meetodi puhul on stohhastilised hinnangud vähem efektiivsed võrreldes esimese meetodiga, kuna sobitatud rühmade moodustamine

vähendab teabesisu ja mõned analüütilised ülesanded võivad muutuda sobimatuks tänu sobimatutele vabadusastmetele.

Esimene meetod on tõhusam, kuid limiteerib analüütiliste meetodite arvu. Teise meetodi funktsionaalsus on vähem efektiivne, kuid võimaldab kasutajal määratleda grupeeritud andmete analüüsimist. Vastavalt analüütilistele eesmärkidele saavad lõppkasutajad mõlemat funktsionaalsust paralleelselt või järjestikku kasutada.

Esialgsed nõuded esimese meetodi päringumehhanismidele on järgmised [3]:

1. Arvutab kasutaja määratletud tingimusteta ja tingimuslikud keskmised ja dispersioonid ning tingimusteta või tingimuslikud keskmised vektorarvutused ja kovariatsioonimaatriksid (*covariance matrices*) mitmevariatiivsete analüüside jaoks.
2. Lõppkasutaja võib kujundada valimeid, valides vaatlusüksused tunnuse või mõne iseloomuliku muutuja järgi.
3. Seejärel saadetakse vaatluste arv konkreetsete muutujatega seotud küsitluste tulemuste kohta, mis on andmete kogumisel pakkuvate üksuste hulga suhtes suuremad.
4. Päringu struktuur peab võimaldama lõppkasutajal määratleda:
    a. Huvipakkuvad muutujad või muutujad või mõõdetud väljundmuutujad
    b. Ravimuutujad
    c. Tingimuslikud muutujad
5. Rühmitatud andmete põhjal arvutatud statistilised elemendid:
    a. Keskmised ja keskmised vektorarvutused
    b. Dispersiooni- ja variatsioonimaatriksid
    c. Korrutismaatriksid
    d. Koefitsiendi- ja ennustusvektorid ning hinnangute dispersioonivektorid
    e. Lihtne ja mitmekordne lineaarne regressioon

Esialgsed nõuded teise meetodi päringumehhanismidele on järgmised [3]:

1. Päringumootor sisaldab eelnevalt määratletud sobivate muutujate kompleti, näiteks sugu, vanus. Lõppkasutaja valib ülejäänud sobivad muutujad, eelistatavalt binaarsete / kategooriliste muutujate hulgast.

2. Kasutaja valib ühe tulemusmuutuja ja määrab süsteemi jaoks kas see on pidev või binaarne.
3. Lõppkasutaja saab ühendatud valimist haiglaid valida või nende hulgast valimise tühistada, kasutades haigla nime või teatud haiglataseme muutujaid.
4. Päringu tulemusel saadud vaatluste või dispersioonide arv on rühmitatud andmete koondamisel kaalu pakkuvate üksuste arvu vahel.
5. Päringumootor sisaldab lävireeglit minimaalse suuruse või sobivate rühmade varieeruvuse piires. Kui minimaalse läve reegel on täidetud, saab kasutaja juurdepääsu sobitatud rühma andmetele. Kui ei täideta reeglit, ei saa kasutaja juurdepääsu sobitatud grupi andmetele.

2.1.2 OpenEHR

OpenEHR on avatud mudel, mis kirjeldab terviseandmete haldamist, säilitamist, otsimist ja vahetamist elektroonilistes terviseandmetes. Viimane versioon on väljastatud 2019. aastal ning see määratleb kolm abstraktset spetsifikatsiooni: viitemudel, teenindusmudel ja arhetüübi mudel, mis omakorda vastavad vastavalt teabe-, arvutus-, ja teadmiste vaatepunktidele. [8] Nendest tuleneb teabemudelite, terminoloogiate ja domeenide sisumudelite selge ontoloogiline eraldamine, millel kõigil on täpselt määratletud ja piiratud ulatus hooldatavate ja kohandatavate süsteemide edendamiseks sõltumata programmeerimiskeelest, inimkeelest või aluseks olevast andmebaasitehnoloogiast. *OpenEHR* ühes peamiseks paradigmaks on kaheastmeline modelleerimine, mis kirjeldab stabiilset viiteteabe mudelit ja muutuva domeeni teadmiste mudelit. [9]

Infomudel peegeldab seda, kuidas terviseandmed on patsientide registris esindatud ning pakub põhilisi üksusi teabe esitamiseks terviseohutuse infosüsteemis, sealhulgas andmetüübid, andmestruktuurid, identifikaatorid ja kujundusmustrid. Ainult see osa on tarkvaras rakendatud. Teadmismudel pakub kliiniliste mõistete ametlikke määratlusi kahe eseme – arhetüüpide ja mallide – kujul, mis täidetakse töö ajal. Arhetüübid on *openEHRi* spetsifikatsiooni keskmes ning need on korduvkasutatavad terviseteabe diskreetsed mudelid, mis sisaldavad andmeklasse ja loogilisi andmegruppide määratlusi viisil, mida tervishoiutöötajad saavad kergema vaevaga mõista ja säilitada [10]. Arhetüüp on andmeklasside või andmeelementide kogumi juhtimisüksus ja see on operatiivselt esindatud ADL-is, kuigi eksisteerivad ka XML- ja JSON-vormingud. Andmebaasid, mis

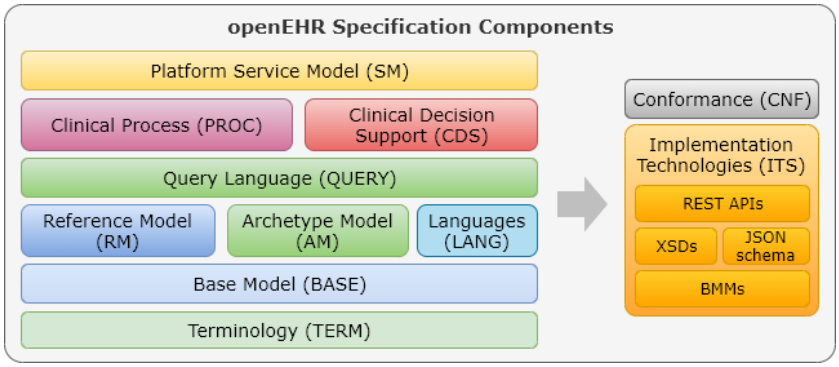
kasutavad kahte või enamat ühe või mitme arhetüübi andmepunkti, on loogiliselt esindatud mallidena.

Arhetüüp sisaldab:

a) metaandmete sektsiooni, mis kirjeldab selle autorit, eesmärki ja versiooni puudutavat teavet jms
b) määratluse sektsiooni, kus on loetletud andmeelemendid, terminoloogia seosed, toetatud andmetüübid ja nende väärtused (kui need on esindatud)
c) ontoloogia sektsiooni, mis sisaldab terminoloogia definitsioone.

Arhetüüpide ja mallidena määratletud sisu saab siduda ühe või mitme olemasoleva terminoloogiakomplektiga.

Joonisel 1 on toodud *openEHR* skeem.



Joonis 1. OpenEHR skeem. [11]

2.1.3 HL7 ja FHIR

Tervishoiu kiire koostalitusvõime ressursid ehk *FHIR* (*Fast Healtcare Interoperability Resources*) on spetsifikatsioon, mille on välja töötanud rahvusvaheline *Health Level Seven* (HL7) nimeline organisatsioon tervishoiuteabe elektrooniliseks vahetamiseks. See on standardi eelnõu, mis kirjeldab andmevorminguid ja elemente ja rakenduse programmeerimisliides (API) elektrooniliste terviseandmete vahetamiseks. FHIR-i ametlikult avaldatud versioon pärineb 2015. aastast DSTU 2 (*Draft Standard for Trial Use*) kujul ja ametlik STU 3-kandiaat on välja antud alates 2016. aasta maist. FHIR-il on tugev alus populaarsetel veebistandarditel (XML, HTTP, OAUTH jt) ning aitab kaasa

varasemate HL7 väljaannete, sh HL7 v2, HL7 v3 ja HL7 CDA20 parimate omaduste kasutuselevõtmisele [12].

FHIR-i põhiomaduseks on ressursid, mis on koostalituvõimelised. Need on analoogsed *openEHR*-i arhetüüpidega ja on modulaarsed komponendid, mille abil FHIR lahendused on üles ehitatud. Ressursid eksisteerivad XML- või JSON-vormingus tekstifailidena ja esindavad üldisi malle, mis sisaldavad andmeelemente eri tüüpi kliinilise haldusteabe jaoks inim-, ja veterinaarmeditsiinis. HL7 veebileht loetleb üles 93 erinevat ressursi klassi, sealhulgas näiteks konkreetsed allergiad, arsti suunamised ja retseptide väljastamised. Lisaks on võimalik leida näiteks pulsi, vererõhu, arteriaalse veregaaside näitajaid ja isegi sotsiaalse ajaloo väärtusi. Viimase tava kohasel on olemas kombineeritud ressursitüübid, mis aitavad väärtusparameetreid mõistlikuna ja kombineerituna hoida. Nende tüüpide abil saab komplekteerida (sarnaselt *openEHR*-i mallidega) erinevaid päringuid tervisekaartide, meditsiinidokumentide ja arstlike otsuste teostamiseks.

Andmeelemendid FHIR-ressursis kaasatakse ainult juhul, kui on oodata, et 80% rakendustest kasutab valitud elementi [13]. FHIR-ressurss on operatiivselt esindatud kas XML-is või JSON-is ning päringuid on võimalik teostada REST-stiilis. Joonisel 2 on toodud sellekohane näide. Päringu vastusel on globaalselt unikaalne identifikaator, erinevate metaandmete osad, URL identifikaator, inimestele loetav XHTML-i kokkuvõtlik jutustus ja standardselt määratletud andmeelemendid.

```
GET http://fhirtest.uhn.ca/baseDstu2/Observation?code=53484-
3&patient=26087&_elements=text&_format=json&_pretty=true&_sort:a
sc=date
```

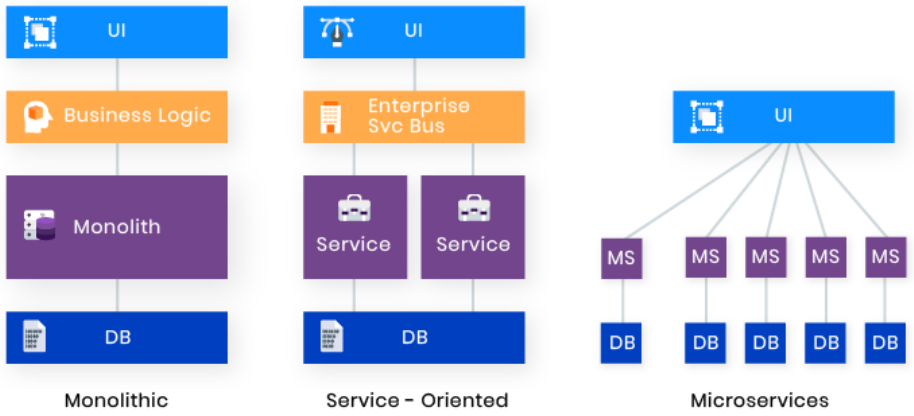
Joonis 2. FHIR REST päringu näide arteriaalse vere gaaside kohta.

2.2 Tööriistad

Antud peatükk kirjeldab ja analüüsib erinevaid arhitektuurilisi vahendeid, valmis tööriistu, serverlahendusi ja andmete maskeerimise võimalusi, mida on tulemuste saamiseks võimalik kasutada.

2.2.1 Arhitektuur

Antud alapeatükis kirjeldatakse erinevaid arhitektuurimustreid. Joonisel 3 on toodud erinevate arhitektuurimustrite põhistruktuurid. Autor ei keskendu oma töös täpsemalt monoliitse arhitektuuri kirjeldusse, sest selle kasutamine eeldab suuremahulist tarkvara projekti.



Joonis 3. Monoliidi, SOA ja mikroteenuse arhitektuuri struktuurid.

2.2.1.1 Mikroteenuste arhitektuur

Mikroteenuste arhitektuuris kasutatakse igat põhilist funktsionaalsust kui eraldiseisvat rakendust. Selline rakendus töötab iseseisvalt ning neid teenuseid saab sõltumatult kasutusele võtta, uuendada ning vahetada. Rakenduste haldamine ei ole osa rakendusest vaid täiesti iseseisev protsess. Samuti võivad olla need rakendused väga erinevad või isegi kirjutatud erinevates keeltes. Mikroteenuseid soovitatakse kasutada pigem konkreetsete projektikomponentide osade lahendamiseks, sest nende negatiivseks osaks on nende haldamise keerukuse kasvukõver ning nende testimise raskus. Mikroteenuseid kasutades peab arvestama ka mälumahu suurenemisega ja lähenemine nõuab arendustiimidelt suuremat koordineerimise ressurssi. [14]

2.2.1.2 *SOA* ehk teenus-orienteeritud arhitektuur

Teenus-orienteeritud arhitektuuri eesmärk on jagada süsteem mitmeks erinevaks äriloogiliseks osaks. Jagatud teenused omavad kindlat sorti ülesandeid, mida on võimalik kohandada varjates rakendamise aluseks olevaid andmestruktuure. Peidetud andmestruktuuride kasutamiseks suhtlevad teenused omavahel läbi integratsioonikomponentide.

Võrreldes mikroteenuste arhitektuuriga pole teenused ja äriloogika üksteisest täiesti eraldatud, vaid andmebaasid on tsentraalsed. SOA-le iseloomulik keskne arhitektuur muudab selle haldamise lihtsamaks ja ühtlasi sobilikumaks suuremate projektide jaoks. Teenus-orienteeritud arhitektuuri negatiivseks osaks on see, et teenusesse muudatuse sisseviimisel tuleb kogu teenust uuendada. Tsentraliseeritud integratsiooni platvormid on seetõttu tavaliselt keerulised ning võivad ajas kasvada mahult suureks. [15]

2.2.2 Vabavaralised lahendused

Prototüübi loomisel on abiks kaks vabavaralist platvormi, mis imiteerivad reaalseid servereid. Nendeks on *CaboLabs EHRServer* [16] ja *HAPI FHIR* [17].

2.2.2.1 CaboLabs EHRServer

Tegemist on *openEHR* standardil ja spetsifikatsioonil põhineva testserveriga, mille seadistamiseks on vaja kasutada avatud lähtekoodiga kliiniliste andmete hoidlat. Tehnilise poole pealt on vajalik rakenduse seadistamiseks *Grails* v2.5.3; andmebaasina kasutatakse *MySQL* andmebaasi ning rakendus on üles ehitatud Java programmeerimiskeeles. Päringute tegemiseks kasutatakse rakenduses juba olemasolevat REST teenuste funktsionaalsust. Päringute tegemiseks saab kasutada ka AQL (*Archetype Query Language*) keelt, mis on oma olemuselt arhetüüpide pärimiseks mõeldud struktuurpäringukeel. Päringu tulemused on saadaval nii XML-, kui ka JSON formaadis. [18]

2.2.2.2 HAPI FHIR

FHIR-põhise testserver kasutab HAPI-FHIR teeki, mis on Java baasil kirjutatud ning avatud lähtekoodiga. HAPI-FHIR on DSTU2 ja DSTU3 toega. Vabavaralises teegi abil on kasutajal enda käejärgi luua testandmeid. Kui *EHRServer* toetab AQL päringukeelt, siis FHIR-ist päringute tegemiseks saab kasutada ainult REST teenuseid. Päringute tulemused kuvatakse nii XML- kui ka JSON-vormingus.

2.2.3 Pilve- ja serverteenused

Võimaliku lahendusena uuris autor pilve- ja serverteenuste võimalusi. Alljärgnevates tabelites 1 ja 2 kirjeldatakse server-tüüpi ja pilveteenuste lahendusi nende positiivsete ja negatiivsete omaduste kaudu. [19]

Positiivsed omadused	**Negatiivsed omadused**
Füüsiline kontroll (varu)koopia üle.	Nõuab investeeringuid riistvarasse ja infrastruktuuri.
Hoiab kriitilisi andmeid ettevõttesiseselt. Ühelgi kolmandal osapoolel pole andmetele juurdepääsu.	Vajalik on IT-toe personali ning serveriruumi olemasolu.
Andmete juurdepääsuks pole vaja internetiühendust.	Hädaolukordades (nt tulekahju) võivad andmed kaduda.
Kulutõhusam väikese- ja keskmise suurusega ettevõtete jaoks.	Puudub töö- või taastumisaeg.

Tabel 1. Server-teenuse eelised ja puudused.

Positiivsed omadused	**Negatiivsed omadused**
Pole vaja kohapealset riistvara ega kulutusi. Sobib hästi väiksematele ettevõtetele, mis võivad andmemahtude kasvamisega kiiresti välja kasvada.	Andmete taastamise kulud võivad kaaluda üles nendest saadava kasu.
Salvestusruumi saab vastavalt vajadusele lisada. Lahendused on sageli tellitavad nii, et makstakse selle eest, mis vaja on.	Organisatsioonil võib olla andmete salvestamisel piirangud, mida tohib turvalisuse tõttu pilve serverisse salvestada.

Taastamist saab algatada ükskõik millisest arvutist.	Kui organisatsioonil ligipääs internetile kaob pole andmetele enam juurdepääsu.
Andmeid saab pilves hoida regulaarselt 15-minutiliste intervallidega, minimeerides andmekaod hädaolukordades. Väiksemate andmekogumite taastamise aeg on paranenud.	Andmete täielik taastamine võib osutuda kalliks ja aeganõudvaks.

Tabel 2. Pilveteenuse eelised ja puudused.

2.2.4 Andmete maskeerimise võimalused

Turvalisuse tagamiseks on rakenduses vajalik andmed maskeerida. Andmete peitmiseks on olemas hulganisti erinevaid algoritme ja meetodeid. Autor analüüsib käesolevas peatükis nelja erinevat meetodit: asendamine, segamine, krüpteerimine ja kustutamine.

2.2.4.1 Asendamine

Asendamist peetakse üheks tõhusaimaks meetodiks. Antud funktsionaalsus baseerub olemasoleva väärtuse asendamisel mõne teise autentse väärtusega, mis asub otsingutabelis *(look-up table)* või ka sama andmestiku piires. Selline lähenemisviis võimaldab salvestada andmeid nii, et neid pole lihtne tuvastada, kuid ühtlasi säilitades realistliku väljanägemisega andmekirjed kogu andmestikus. Samas on võimalik andmeid vastavalt otsingutabeli väärtustele tagasiühilduvalt määrata.

On mitmeid andmevälja tüüpe, kus selline lähenemisviis pakub optimaalset kasu kogu andmekogumi või alamkogu maskeerimisel. Näiteks võib tuua analüüsitava andmestiku, kus lähteandmetes on olemas nii ees-kui ka perekonnanimed ja isikute sugu. Turvalisuse tagamiseks on antud andmestikus võimalik teostada asendamist näiteks kahel erineval viisil. Esimesel juhul saab asendada ees-ja perekonnanimed otsingutabeli genereeritud väärtustega ning teine võimalus lubab vahetada omavahel soolised väärtused, salvestades tehtud muudatused otsingutabelisse. Asendamise negatiivseks küljeks on vaja, et

otsingutabelid oleksid küllaltki ulatuslikud, et oleks olemas võimalikult suur varieeruvus. [20]

2.2.4.2 Segamine

Segamine on küllaltki levinud andmete hägustamise viis. See sarnaneb asendamismeetodiga, kuid otsingutabeli kasutamise asemel tuletab see uue väärtuse samast andmeveerust, mida varjatakse. Lihtsustatult võib öelda, et andmed ühes ja samas veerus on juhuslikult segatud.

Segamist kasutatakse sageli finantsandmete peitmisel. Näiteks varjamaks ettevõte tarnijate nimesid ja kontosid on võimalik need väärtused omavahel segada. Praktikas kasutatakse asendamist ja segamist siiski koos- esmalt asendatakse andmeväärtused ja seejärel alles segatakse, mis tagab selle, et algandmete leidmine on keerulisem.

Kasutades segamist iseseisva meetodina on võimalik üsna algupäraste teadmistega rakendada pöördprojekteerimist ja reaalsed andmed algkujul kokku panna. Meetod on veelgi ebaturvalisem, kui turvatav andmestik on väike, sest sellisel juhul on võimalik katse- eksitusmeetodiga leida andmete originaalkuju. [21]

2.2.4.3 Krüpteerimine

Krüpteerimist peetakse üheks keerulisemaks viisiks andmete peitmisel ja hägustamisel. Krüpteerimisalgoritmid nõuavad enamasti seda, et andmete kasutamiseks oleks kasutajal või rakendusel olemas võtmekombinatsioon avalikust- ja privaatsest võtmest. Seda nimetatakse avaliku võtme krüptograafiaks. Võtmete ülesanne on kontrollida ja tagada seda, et kasutajal, kes andmeid soovib vaadata, oleks selleks õigus.

Võtmed töötavad nii, et saatja šifreerib sõnumi kasutades vastuvõtja avalikku võtit, vastuvõtja dešifreerib sõnumi enda privaatse võtmega. Privaatne võti on alati sõnumi vastuvõtja valduses, seevastu avalik võti on mõeldud avalikuks levitamiseks.

Sensitiivsete andmete puhul püütakse enamasti sensitiivsete andmete puhul kasutada krüpteerimist. Krüpteerimise kasutamine on võrreldes teiste meetoditega ressursikulukam ning juhul kui andmestikus on palju väärtusi, siis võib kogu protsess muutuda aeglaseks. [22]

2.2.4.4 Kustutamine

Kustutamine on andmete peitmise meetoditest kõige lihtsam ja käepärasem kui tegemist pole oluliste kirjetega ja algandmeid ei ole vaja hilisemas faasis uuesti kasutada. Andmete kustutamise korral antakse andmetabeli konkreetsele väärtusele nullväärtus.

Enamikel juhtudel vähendab kustutamine andmekogumis säilitatava andmete terviklikkuse taset. Nullväärtus võib moonutada analüüsi tulemusi, sest tegu ei ole realistliku väärtusega. Kustutamisel ei ole algväärtusi võimalik valideerida. Samuti välistab kustutamine paljudel juhtudel andmete tagasipööramise ehk algandmeid ei ole võimalik taastada. [23]

2.3 Töö koostamise protsess

Käesolevas peatükis kirjeldab lõputöö autor töö koostamise protsessi, alustades arhitektuurilise valiku tegemisest ja lõpetades rakenduses alakomponentide meetodite valikuga.

2.3.1 Töö arhitektuurne valik

Antud magistritöö praktilise osa arhitektuurse lahenduse valimisel välistas autor esimesena monoliitse arhitektuuri. Monoliitse arhitektuuri haldamine on suurte andmemahtude ja rohkete lisafunktsionaalsuste arendamisel keerukas ning hilisemas faasis ka kulukas.

Mikroteenuseid ja teenus-orienteeritud arhitektuuri võrreldes selgub, et SOA arhitektuur on sobilikum rakendada suurtes ettevõtetes ning mikroteenused sobivad paremini projektipõhiste teenuste loomiseks. Kiire ja väikeste osade kaupa arendamine muudab mikroteenuste kasutamise paindlikumaks kui SOA arhitektuuri.

Antud lahenduse kirjutamisel eelistab autor mikroteenuste põhimõtet, sest süsteemi osad on erinevad ja kohati üksteisest sõltumatud ning iga teenuse arendamine nõuab erinevas mahus ressurssi. Seega vastab mikroteenuste arhitektuur paremini loodava lahenduse nõuetele ja eripäradele.

2.3.2 Päringutüübi valik

Magistritöö praktilise osa üheks olulisemaks valikuks on kliendile saadetava päringu tüüp, mis vabavaralist lahendust arvesse võttes saab olla kas AQL või REST- päring. Antud tööd kirjutades lähtus autor päringu tegemises viisi universaalsusest. Meditsiiniinfosüsteemide standardid openEHR ja FHIR toetavad mõlemad REST päringu teostamist ning seetõttu valis töö autor lahenduse realiseerimiseks REST-päringu. AQL-päringul on teatud juhtudel omad eelised, millest on pikemalt juttu töö analüüsi osas.

2.3.3 Server- komponendi valik

Lahenduse valik sõltub sellest, missugustel ärilistel eesmärkidel ja millise võimekusega lahendust on kliendil vaja. Nagu 2.2.3 alapeatüki tabelitest selgub, siis on server-lahendus eelistatum variant, kui on tegemist sensitiivsete andmetega, sest pilveteenuse korral on võimalik andmeid kergemini kuritarvitada. Näiteks, 2017 aastal jagas Ühendkuningriikide haigla *Royal Free* enam kui 1.6 miljoni patsiendi andmeid Google DeepMind teenusega nende endi teadmata. [24]

Kuna käesoleva töö eesmärk on töödelda sensitiivseid andmeid, siis on töö autor valinud lahenduseks server- tüüpi teenuse. Teatud juhtudel kasutatakse ka nende kahe teenuse kooslust, hoides näiteks mitteväärtuslikke andmeid pilves ja diskreetsemaid andmeid kohalikus serveris. Kombineeritud meetodi jätab autor kõrvale lihtsuse tagamiseks.

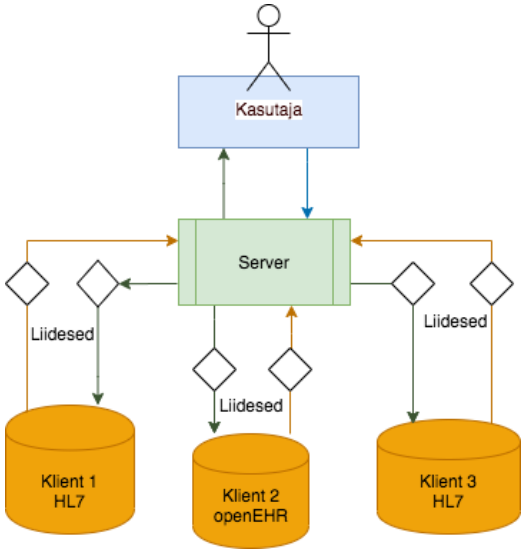
2.3.4 Andmete maskeerimise valik

Vaatamata sellele, et krüptoalgoritmide kasutamine võib olla aja- ja ressursimahukad, kasutab autor antud töös just nende rakendamist. Krüptograafiliste meetodite kasutamine on ajas järjest rohkem muutunud töökindlamaks ja paremaks ning paljudel juhtudel lihtsamini kasutavamaks. Lisaks leiab autor, et teised meetodi, näiteks asendamine ja segamine, ei ole sensitiivsete andmete korral piisavalt turvalised.

Samuti näitab autor rakenduses, et teatud juhtudel tuleb kõne alla ka andmeväljade kustutamine. Näiteks analüüsides kvantitatiivseid tulemusi, ei oma aadress-, või tööstaatuse väljad mingisugust rolli.

3 Autoripoolne lahendus

Autor pakub antud peatükis lähtuvalt eelpool teostatud meetodite valikust välja omapoolse lahenduse sekundaarsete meditsiiniandmete töötlemiseks. Lahenduskäigu saab jagada kaheks osaks: andmete pärimine ning andmete tagastamine. Joonisel 4 on toodud lahenduskäigu lihtsustatud skeem.



Joonis 4. Autoripoolse lahenduskäigu skeem

Alljärgnevates alapeatükkides on toodud rakenduse töövoog ning iga osa täpsem kirjeldus.

3.1 Andmete töövoog

Järgnevalt on kirjeldatud protsess, kuidas rakenduse abil andmeid päritakse.

1. Iga kasutaja saab rakendusse sisenedes oma privaatse- ja avaliku võtme. Võtmete abil veendutakse kasutaja autentsuses ning andmete liigutamisel kontrollitakse kasutaja õigusi toimingute sooritamisel.

2. Kasutaja edastab päringu serverile, millele lisatakse võtmete abil loodud kasutajale omane signatuur. Signatuuri on vaja selleks, et:

    a. Tagada terviklikkus – sõnumit ei saaks edastamise ajal muuta

b. Autentsus – sõnumi autor on tegelikult see, kes ta enda väitel on

c. Tagasilükkamine – sõnumi autor ei saa hiljem eitada, et ta ei olnud küsija.

3. Server töötleb antud töö raames päringu kaheks erinevaks standardiks – *openEHR* ja *FHIR*.

4. Server saadab päringud igale kliendile töötlemiseks.

5. Liides kontrollib kliendi infosüsteemi standardi olemasolu ning valib seeläbi millisel kujul päring edastada.

6. Vastavalt kindlaksmääratud tüübile edastatakse päring liitunud klientidele.

Andmete tagastamise protsessi kirjeldus:

1. Klient realiseerib serveri poolt saadetud päringu.

2. Saadud tulemusread muudetakse räsipuu (*hash tree*) abil peidetud sõnumiks. Iga puu juurtipp signeeritakse, mis tagab andmete korrektsuse. Samuti on selle abil võimalik kindlaks määrata, mis ajahetkel andmeid tagastati.

3. Tagastusliides kontrollib, kas nõudmised, mille hulka kuulub ka signatuur ja räsifunktsiooni juurtipu olemasolu, on lisatud. Kui ei ole, siis keeldutakse andmeid edastamast.

4. Serveris kogutakse kõikide klientide andmed kokku ja agregeeritakse üheks tervikuks vastavalt standarditele (*openEHR* ja *FHIR*). Andmed, mida server ei oska agregeerida, kustutatakse. Standardite abil on võimalik valida personaalne info ning eemaldada või peita vastavad tulemused.

5. Serveris tehakse esmane statistiline analüüs, mille käigus püütakse analüüsida esmajärgus kvantitatiivseid väärtuseid.

6. Analüüsi tulemused edastatakse kasutajale.

3.2 Lahenduskäigu komponendid

Autor jagab lahenduskäigu neljaks alaosaks:

- Klient-server võrk

- Lahenduse turvalisus

- Personaalandmete anonümiseerimine

- Tulemuste kirjeldamine.

Alljärgnevates alapeatükkides kirjeldatakse täpsemalt nende alaosade sisu.

3.2.1 Klient-server võrk

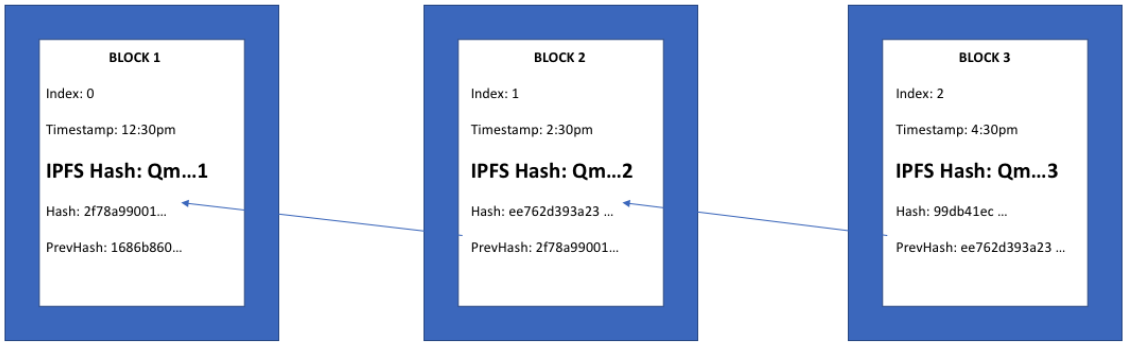
Rakenduse põhiosaks on hajutatud klient-server võrk (P2P), millega saavad teised kliendid liituda. Rakenduse kasutaja saadab päringud serverile, misjärel edastatakse päring igale liitunud kliendile. Kliendid aktsepteerivad päringu pöördumist või lükkavad selle tagasi vastavalt punktis 3.1 kirjeldatud tingimustele. Kui päring on autentne, pöördutakse päringu tegemiseks kliendi infosüsteemi. Andmed saadetakse serverile tagasi, mis omakorda veendub nende korrektsuses.

3.2.2 Lahenduse turvalisus

Autoripoolse lahenduse turvalisuse saab jagada kaheks: kasutaja- ja serveripoolseks.

Kasutajapoolse turvalisuse korral on oluline veenduda, et kasutaja, kes rakenduse kaudu päringut teeb, omab selleks õigusi. Kasutaja tuvastamiseks sobib kahepoolne autentimine (*two-way authentication*). Selline lähenemine võimaldab kontrollida, milliseid õigusi ja rolle konkreetne kasutaja omab. Edasi, kui kasutajal on olemas luba, siis genereeritakse vastavalt privaat- ja avalikuvõtme paarid. Nende võtmetega on võimalik teostada signeerimist ja ajatembeldust. Ajatembeldus tagab selle, et kasutajast jääks maha selge märk, et just tema sooritas konkreetsel ajahetkel päringu ning signeerimine tagab selle, et kasutajal on õigus tulemusandmeid kasutada.

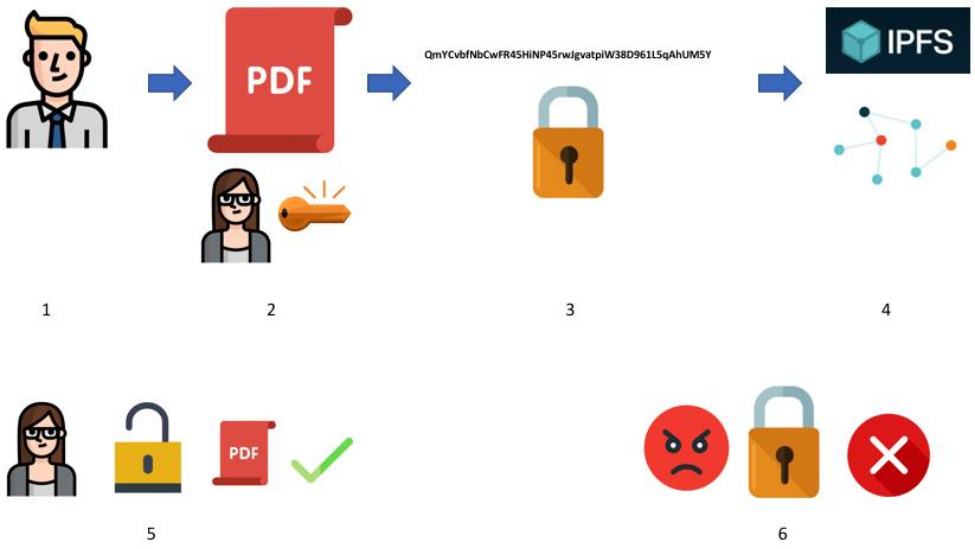
Teine turvalisuse aspekt on serveri ja asutuste vaheline andmevahetus. Selleks saab kasutada p2p-võrku - näiteks IPFS -, kus andmevahetus toimub plokiahela ja eelpool mainitud signeerimismeetodite abil. Võrku salvestatakse räsitud kujul andmete füüsiline aadress. Andmete kasutamiseks peab kasutaja teadma räsitud aadressi ning privaat-avaliku võtme kombinatsioone. Kui asutusest tulevad uute päringute tulemused, lisatakse need ühtsesse plokiahelasse. Joonisel 5 on toodud näide, kuidas andmed plokiahelasse on seotud.



Joonis 5. IPFS plokiahela näide. [25]

Toome siinkohal näite kasutajaloost, kuidas andmeid turvaliselt jagada. Joonisel 6 on protsess toodud ka piltlikul kujul.

1. Bob soovib jagada ainult Alice'ga personaalseid meditsiiniandmeid.
2. Bob saadab oma andmed IPFS serveri töökataloogi ja krüpteerib selle Alice'i avaliku võtmega.
3. IPFS server genereerib krüpteeritud faili jaoks spetsiaalse räsikoodi, mille abil on võimalik andmetele ligi saada.
4. Bob'i andmed asuvad nüüd turvaliselt võrgus.
5. Alice saab Bob'i saadetud andmeid dekrüpteerida ja kasutada, kuna talle kuulub faili avamiseks privaatvõti.
6. Kolmandad osapooled ei saa faili kasutada kahel põhjusel: neil puudub Bob'i andmete jaoks koostatud räsikood, ning isegi kui neil oleks kood olemas, ei saa nad andmeid lahti, sest neil puudub Alice'i privaatvõti.



Joonis 6. Turvaline andmevahetus protsess. [26]

3.2.3 Personaalandmete anonümiseerimine

Lõpplahenduse eesmärk on sekundaarsete meditsiiniandmete analüüs, kusjuures personaalsed andmed ei oma tähtsust, sest need andmeväljad ei anna analüütilises mõttes midagi juurde. Selleks, et andmeid peita, on oluline mõista missugusel kujul andmeid esitletakse. Lisas 1 on välja toodud *FHIR* päringu vastuse näide- tulemuseks on JSON-formaadis andmestik. Formaadi andmestikust selgub, milliseid parameetreid on vaja peita ja kus personaalsed väärtused asuvad.

Vaatluse alla tuleb kaks võimalust, kuidas personaalsed andmed anonüümseks muuta:

a) Andmeväljad peidetakse, kasutades näiteks räsimist või väljade asendamist.
b) Andmeväljad kustutatakse.

Antud töös on realiseeritud mõlemad lahendused. Tabelis 3 on toodud näide, kuidas personaalseid väärtuseid peita. Tabelis 4 on näide, kuidas andmete kustutamine toimub.

Nimi	Vanus	Sugu	Haigus kood
John Smith	40	M	R73.09
Mary Watson	59	N	E11.65
David Zaul	62	M	I25.9

Nimi	Vanus	Sugu	Haigus kood
*	35 < Vanus < 65	M	R73.09
*	35 < Vanus < 65	N	E11.65
*	35 < Vanus < 65	M	I25.9

Tabel 3. Tulemuste peitmine.

Nimi	Vanus	Sugu	Haigus kood
John Smith	40	M	R73.09
Mary Watson	59	N	E11.65
David Zaul	62	M	I25.9

Vanus	Sugu	Haigus kood
35 < Vanus < 65	M	R73.09
35 < Vanus < 65	N	E11.65
35 < Vanus < 65	M	I25.9

Tabel 4. Andmete kustutamine. Andmed enne (vasakul) ja pärast (paremal).

3.2.4 Tulemuste agregeerimine

Lahendus keskendub peamiselt kvantitatiivsele analüüsile, kus kasutatavaid objekte iseloomustavad suurused on arvulised. Andmeid analüüsitakse matemaatiliste meetoditega ja ka järeldused on enamasti arvuliselt kirjeldatavad. Nagu iga JSON-väljundi puhul, on ka siin teada milliseid väärtuseid on vaja omavahel kokku panna enne, kui lõpp-kasutajale tagasi saadetakse.

Lahendustulemuste hulka kuuluvad järgmised esmase analüüsi tulemused:

- Keskmine
- Miinimum
- Maksimum
- Mediaan

3.3 Lahendus

Järgnevalt on iga eelpool kirjeldatud osa kohta realiseeritud lahendus. Tarkvaraline lahendus on realiseeritud kasutades *Pythoni* (versioon 3.7) programmeerimiskeelt. Kasutusel on ka IPFS hajutatud võrk, mille eesmärk on pakkuda andmevahetust. Turvalisuse tagab avaliku võtme krüptograafia. Erinevate standardite käsitlemiseks on kasutusel eelpool mainitud *openEHRServer* ja *HAPI FHIR* lahendused, mille poole päringute tegemiseks pöördutakse, et saada tulemusi, mis sarnanevad meditsiiniasutuste omadele.

Antud lahenduse näitlikustamiseks kasutatakse autori enda arvutit, milles luuakse mitu klienti. Töö loomiseks kasutatud masina parameetrid toodud tabelis 5.

Operatsioonisüsteem	macOS High Sierra v10.13.16
Protsessor	2,2 GHz Intel Core i7
Mälu	16 GB 1600 MHz DDR3

Tabel 5. Arvuti parameetrid töö lahendamiseks.

Järgnevate peatükkide näidetes on kasutusel kohtvõrgu aadressid erinevate andmesideühendus lõpp-punktidega ehk portidega. Andmeid ja päringuid vahetatakse edastusohje protokolli ehk TCP abil (*Transmissioon Control Protocol*).

Rakenduse kood on kättesaadav Tallinna Tehnikaülikooli koodihoidlast [27].

3.3.1 Klient-server võrk

Loodud on P2P-võrk, millega on võimalik välistel masinatel liituda. Jooniselt 7 on näha, et masinaga 6001 on liitunud kaks välist masinat, mis asuvad portidel 5003 ja 7001.

```
127.0.0.1 wrote:
/nodes
Sending to node localhost:5003 following nodes {'nodes': ['localhost:6001', 'localhost:5003', 'localhost:7001']}
Sending to node localhost:7001 following nodes {'nodes': ['localhost:6001', 'localhost:5003', 'localhost:7001']}
6001: Sending response: {'message': 'Nodes list', 'total_nodes': ['localhost:6001', 'localhost:5003', 'localhost:7001']}
```

Joonis 7. Lokaalne P2P võrgu näide.

Seejärel edastab masin pordiga 6001 päringu teistele liitunud tippudele (*node*). Iga päringu saanud tipp hakkab töötlema päringut.

3.3.2 Turvalisus

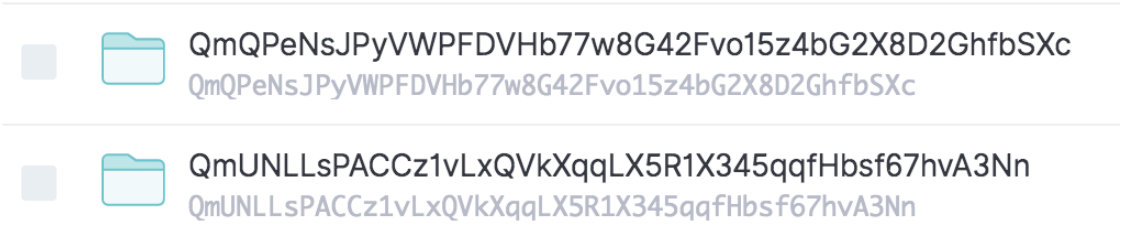
Käesolevas töös võeti kasutusele IPFS-süsteem [28], mis oma olemuselt on detsentraliseeritud võrk, mille abil on võimalik jagada näiteks faile või JSON-objekte. Andmed, mida jagatakse on räsitud ning paigutatud serverisse, kust neid on võimalik edasi saata. Tagamaks täit turvalisust kahe osapoole vahel, on lahenduses kasutusel ka signeerimine ning selleks kasutas töö autor GPG teeki. GPG lahendus genereerib avaliku- ja privaatvõtme. Joonisel 8 on toodud näide genereeritud avalikust võtmest.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF61CjoBCADH9rtmKlGK/pGLOzpNxeWwepOBu4xvCDK5ShsNTvzKGY0LWWGg
d9TZFlRedZxNoJPQOSZ4G8NLiqsZqLf+j1+gAvgiaK7vh17LVkzDCmy0mm65JD/F
XnziijjVGAgEg21mdsF+83IG4gmTpopITo7Dl9EZQ1FgZX7xP7AZHe8gcZYw400Y
aPWsO10MthPeHiDNRfksxTlL7sdXz1Y1W8EsnBZNa1eaBl0VI9vBgxl/O1mNE9M+
81mnDeU5vjuTrkgh+ZIe1iAzIjD4UyqNs1nV08nz0Z6h9cOpPwQbnUuhTxjKuOm7
5//MKXWGvvCyfxjsOFrXJpPFZSi/1o9/iW4JABEBAAG0FmpvaG4gZG9lIDxqb2hu
QGRvZS5lZT6JAVQEEwEIAD4WIQQZK2yL1lDLXhB9Ame0g9s9GMpqxwUCXrUKOgIb
AwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRC0g9s9GMpqxy6GB/oC
H/CMVLl6XgSdrVf3WNObfHmVqvSQrd+GZtbbmqmt+4LbuszAX/OCFzRUYXdQYjUc
Z9B0bUPpXK8iPGRq1t4rzZFqt6sVHQe6q1q0sBCnbx0AvDSYFzGr6oLeLx5Mg6Ic
thdn2wTJXUaTvbuEZt1CYuCFeegeBE/rG8Ez2LEP+Ht3uk/5K1krGsuomOdHSjTj
clAbDvNtQCoQF97MLmoWC5AkccMrMb5j+oaX73xPRf/B9viQcV05ytHrhooTbYat
n6PAmWehg+tyPGt1LS8+zOwptEjEnIBpf5093EtLxY9rJiDUbHyRdm0dHIWgqgMh
GclbH+kWv5+j7KbES+BBuQENBF61CjoBCADS+fy08W23PEry+Y6JeNLEd/MwTp+Y
KP/1fBFzZR/gx60m5wSg+SuhFjKWZZ9pGkquc2AWufk7NoWK1C/A+ve72M39fRXH
```

Joonis 8. Osa genereeritud avalikust võtmest.

Joonisel 9 on toodud näide sellest, millisel kujul jõuavad andmed IPFS serverisse. Sellisel kujul pole võimalik tuvastada, mis on andmete sisuks ja kui mitmest asutusest andmeplokk koosneb.



Joonis 9. IPFS serveris asuvad krüpteeritud andmed.

3.3.3 Andmete hajutamine

Järgnevalt on toodud kaks joonist: esimene, Joonis 10, sisaldab originaalandmeid, mis on *openEHR* standardis ning Joonis 11 on selle andmestiku tulemus pärast töötlemist. Tegemist on andmete kustutamise meetodiga, kus personaalsed väljad, nagu näiteks kasutaja id, nimi ja sünniaeg, on originaalist eemaldatud.

```
{
  "resourceType": "Patient",
  "id": "1234567890",
  "gender": "male",
  "birthDate": "2000-09-05",
  "active": true,
  "deceasedBoolean": false,
  "name": [
          {
                  "use": "official",
                  "family": "Smith",
                  "given": "John"
          }
  ],
          "address": [
            {
                  "use": "home",
                  "text": "1 Farnham Road, Guildford, Surrey",
                  "line": [
                          "1 Farnham Road"
                  ],
                  "city": "Guildford",
                  "postalCode": "GU24RG",
                  "country": "UK"
                  }
          ]
  }
```

Joonis 10. Originaal andmestik.

```
1. {
2.   "resourceType": "Patient",
3.   "gender": "male",
4.   "active": true,
5.   "deceasedBoolean": false,
6.   "name": [],
7.   "address": [
8.      {
9.           "use": "home",
10.                "city": "London",
11.                "country": "UK"
12.     }
13.  ]
14.  }
```

Joonis 11. Andmed pärast töötlust.

3.3.4 Tulemuste agregeerimine

Tulemuste koondamisel kasutas autor Pythoni teeki *pandas* ja *numpy*, mis on eelkõige mõeldud andmete ettevalmistamiseks ja töötlemiseks.

Antud töös on kasutatud meetodit, kus võetakse kõige väiksem ja suurem arvuline väärtus, seejärel väiksem väärtus ümardatakse alla esimese täisarvulise väärtuseni ning suurim ümardatakse üles.

Näiteks, olgu meil järgmine andmestik vanuselised väärtused:

```
{"name":  "Peep",    "age":  31,  "city":  "Tartu",  "HR/bpm":  58},
{"name": "Toomas",    "age": 23,   "city": "Elva",    "HR/bpm": 110},
{"name":  "Maarja",   "age":  36,  "city":  "Paide",   "HR/bpm": 77}
```

Pärast andmete agregeerimist saame Joonise 12 toodud tulemused.

```
    age  HR/bpm
0    20      60
1    40     110
```

Joonis 12. Agregeeritud vastuste tulemused.

Keerulisemate ja sügavamate JSON-tulemsute läbimiseks on kasutusel rekursiivne meetod, mis otsib kokku kõik arvulised väärtused koos nende nimedega. Sellisel viisil on teada, missuguste väärtuste ja tulemustega arvutamised toimuvad.

4 Töö analüüs

Järgnevates peatükkides analüüsib autor tehtud töö tugevaid külgi, toob välja puudujäägid, kirjeldab alternatiivseid lahendusvõimalusi, mida oleks saanud töö realiseerimiseks kasutada. Lõpetuseks toob autor välja tööga seotud tulevikuplaanid ja edasised eesmärgid.

4.1 Töö tugevad küljed

Töö lõpptulemuse tugevuseks on asjaolu, et lahendus on platvormidest sõltumatu. Rakendus ei eelda ühtegi kindlat andmebaasi- sobivad nii relatsioonilised kui ka mitterelatsioonilised andmebaasid. Samuti ei oma tähtsust, missuguseid operatsioonisüsteeme erinevad asutused kasutavad. Kõige tähtsam aspekt on see, et peab olema toetatud REST-tüüpi päringute tegemise võimalus, mille abil on võimalik koguda andmeid olenemata andmebaasidest või operatsioonisüsteemidest.

Töö tulemusena pakkus autor välja omapoolse lahenduse ja praktilise rakenduse aktuaalsele probleemile. Loodud rakendus saab olla vundamendiks edasistele uuringutele ja viisidele, kuidas andmeid analüüsida ja turvaliselt transportida mitmete asutuste vahel. Alapeatükis 4.4 on välja toodud võimalikke teemaga seotud edasiarendusi ning tulevikusuundi.

4.2 Töö puudujäägid

Väljapakutud lahendusel puudub hetkel kasutajaliides. Päringute edastamine toimib läbi *cURL* päringute. Joonisel 13 on toodud cURL päringu näide. See tähendab seda, et kasutaja peab täpselt teadma, missugune on päringu keha ning mis parameetrid tuleb kaasa anda. Kasutajaliidese olemasolu võimaldaks kasutajal etteantud parameetrite põhjal päring kokku modifitseerida, mistõttu oleks eksimisruum väiksem. Samuti annaks kasutajaliidese kiht tarkvaralises mõttes juurde võimaluse serveri pool kasutaja sisendit töödelda.

```
curl -X POST http://www.yourwebsite.com/login/ -d
'username=yourusername&password=yourpassword'
```

Joonis 13. cURL päringu näide.

Teiseks puudub antud lahendusel paindlikkus. Kuna käesolev töö keskendub peamiselt kahele enamlevinud standardile, jäävad vanemad versioonid või kohandatud lahendused kõrvale.

Viimase puudujäägina võib välja tuua rakenduse kiiruse. Tehnilise poole pealt ei ole rakendust testitud suurte andmemahtude peal. Antud töös ei ole püütud leida parimaid võimalikke algoritme, vaid rõhk on pandud terviklahenduse loomisele. OPEN-projekti esialgses kirjelduses on kirjas ka mittefunktsionaalne nõue, mis lubab rakendusel ühte päringut töödelda mitu tundi, mis igapäevases tarkvaraarenduses on pikk aeg.

4.3 Lahenduse muud võimalused

Järgnevalt on autor toonud välja alternatiivseid võimalusi, kuidas oleks saanud püstitatud probleemi veel lahendada ning analüüsitakse nende erinevate meetodite kitsaskohti.

Antud magistritöös sai kasutatud REST-päringuid, millel on siiski ka olulised piirangud meditsiiniandmete pärimisel. Näiteks toob E. Allwell-Brown välja oma artiklis [29], et REST-päringud ei tule toime keerukamate, mitmekihiliste arhetüüpsete päringutega, mis teeb nad keeruliseks ja raskesti loetavaks ning sobilikum oleks kasutada AQL-päringuid. AQL päringud meenutavad oma olemuselt relatsioonilise andmebaasi süntaksit, mistõttu on keeleliselt kergem ühendada mitut tabelit ja mitmekihilisi päringuid. Päringu teostamisel peab kasutaja täpselt teadma, milliseid muutujaid või väärtusi otsitakse. Joonisel 14 on näide sellest, milline näeb välja AQL-päring. Juhul kui tegemist ainult *openEHR* standardiga, siis on soovituslikum kasutada AQL-tüüpi päringuid. Uuemad FHIR standardid aga ei toeta AQL keelelisi päringuid, mistõttu käesolevas töös sai kasutamiseks valitud REST-tüüpi lähenemine.

```
SELECT v/uid/value as composition_id,
 obs/data/origin/value as measurement_time,
 obs/data/events/data/items[at0004]/value/magnitude as systolic,
 obs/data/events/data/items[at0005]/value/magnitude as diastolic,
 c/context/setting/value as composition_setting,
 c/name/value as composition_title,
 c/context/start_time/value as composition_start_time
FROM Ehr [uid=$current_ehr_uid]
CONTAINS VERSION v
CONTAINS COMPOSITION c[openEHR-EHR-COMPOSITION.encounter.v1]
CONTAINS OBSERVATION obs[openEHR-EHR-
OBSERVATION.blood_pressure.v1]
WHERE obs/data/events/data/items[at0004]/value/magnitude > 185
ORDER BY c/context/start_time/value
```

Joonis 14. AQL päringu näide.

Sekundaarsete terviseandmete päringu probleemi püüdsid lahendada ka S.Wu ja J.Du oma artiklis „Electronic Medical Record Security Sharing Sodel Based on Blockchain“. [30] Nemad kasutasid oma lahenduses plokiahela meetodit, kuid artiklis ei olnud välja toodud täpsem tarkvaraline lahendus. Aprillis 2020 tuli välja Eesti ettevõte Guardtime OÜ omapoolse API-põhise platvormiga [31], mis kasutab samuti plokiahelat tervisekaartide vahendamiseks. Nende lahenduse puudujäägiks antud töö kontekstis on see, et liidese kaudu on võimalik teha ainult üht päringut ühe tervisekaardi kohta.

Antud töös leiti ja tagastati tulemuseks kõige lihtsamad statistilised väärtused – miinimum, maksimum, mediaan, keskmine. Nende väärtuste eesmärk oli näidata, et tagastustulemuste peal on võimalik teostada esmajärgus lihtsamaid analüütilisi arvutusi. See aga ei tähenda seda, et tulemuste abil ei oleks võimalik leida keerulisemaid ja ehk otstarbekamaid arvutusi nagu näiteks standardhälve või t-testi väärtuste leidmine. Teiste analüütiliste väärtuste leidmine ei olnud esmajärgus oluline antud töö kontekstis. Täpsemate väärtuste leidmine on hea võimalus töö edasi arendamiseks, kuid sel juhul võiks olla konkreetsete nõudmistega ülesande püstitus ning lähteandmete kogu.

Antud töös kasutati tulemuste analüüsimiseks Pythoni teeke *numpy* ja *pandas*. On olemas ka teised analoogsed ja teatud juhtudel ka paremad võimalused - näiteks *TensorFlow* [32] ja *PyTorch* [33] lahendused. Mõlemad võimalused on samuti vabavaralised ning mõeldud kõigile kasutamiseks. Teatud katse-eksitusmeetodid näitavad, et *numpy* ja *pandas* võivad suurte andmemahtude korral kasutada palju arvuti ressurssi. [34]

Täpsemalt puudub *pandas'e* teegil paralleelarvutuslik tugi ning suurte andmestike peal võivad tulemuste arvutused võtta kaua aega. Kuna aga käesoleva töö eesmärk oli näidata ja tõestada, et võimalik on üldisi arvutusi teostada ning autoril on suuremad kogemused just valitud lahenduste kasutamisel, siis langes otsus just selliste valikute kasuks.

4.4 Töö edasised plaanid

Käesolev magistritöö pakkus välja esmase baaslahenduse sekundaarsete meditsiiniandmete pärimisele erinevatest asutustest. Töö kirjutamise käigus selgusid mitmed kitsaskohad, mis vajaks edasist arendamist, sobides ka järgnevateks uurimisobjektideks.

Näiteks võiks edaspidi luua rakendusele kasutajaliidese, otsida võimalusi rakenduse kiiruse parandamiseks erinevate algoritmide näol ning luua terviklahendus erinevate standardite ühtseks käsitlemiseks.

Kuna töös on kasutatud ka väliseid teeke ja lahendusi, siis võimalikuks edasiarenduseks oleks realiseerida nendest sõltumatu platvorm. Näiteks saaks IPFS süsteemi ja avaliku võtme krüptograafia ühendada ühtseks lahenduseks, kusjuures selle täpsem eesmärk tuleks kirjeldada selgete ja piiritletud nõudmistega.

2020 aasta mais kutsus *Innovative Medicine Initiative* (IMI) kõiki teadusasutusi üles kandideerima tuleviku meditsiiniliste lahenduste arendamisesse. Mitmed kutses välja toodud teemad on väiksemal või suuremal määral seotud meditsiiniandmete teisese kasutamisega ning turvaliste andmevahetusvõimalustega. [35] Käesolev töö on väike samm selles suunas, et vastata IMI nõudmistele ning panustada e-tervise infrastruktuuri edendamisse.

Viimase sammuna sooviks autor leida koostöövõimalusi mõne era- või avalikusektori asutusega, kelle eesmärk oleks kirjeldatud probleemile lahendusvõimaluste arendamine.

Kokkuvõte

Käesoleva töö eesmärgiks oli välja pakkuda ja realiseerida üks võimalik lahendus sekundaarsete meditsiiniandmete pärimiseks ja analüüsimiseks.

Töö esimeses pooles analüüsiti ja pandi paika teoreetilised alused autoripoolse lahenduse jaoks ning töö teises pooles esitles autor enda lahendust, realiseerides ühe võimaliku viisi OPEN-projektis esitatud probleemile.

Praktiline osa realiseeriti *Python*i programmeerimiskeeles. Lisaks kasutati puhtale keelele lisaks veel *numpy, pandas, gnupg* ja *ipfs* teeke.

Töös ei realiseeritud tarkvaraarenduses iseloomulikke ühik- ega integratsiooniteste, sest need jäid antud töö skoobist välja ning lõpp-tulemusele ei lisa suurt väärtust.

Töö lahenduskäigus said täidetud varasemalt välja toodud viis nõuet :

1) Loodi süsteem, P2P-võrk, millega kliendid saavad liituda
2) Andmepäringute tegemine liitunud klientide vahel.
3) Tagastustulemuste hajutamine;
4) Meditsiiniasutustes kasutusel olevate standardite – openEHR ja FHIR – sidumine.
5) Turvaline andmeedastus osapoolte vahel, kasutades IPFS lahendust.

Töö käigus leiti mitmeid võimalusi töö jätkamiseks ja edasiarendamiseks, mis ei mahtunud käesoleva töö skoopi. Lõputöö täitis alguses püstitatud eesmärgid, pakkudes välja ja realiseerides omapoolse nõuetele vastava lahenduse.

Kasutatud kirjandus

[1] S. M. Meystre, C. Lovis, T. Bürkle, G. Tognola, A. Budrionis ja C. U. Lehmann, „Clinical Data Reuse or Secondary Use: Current Status and Potential Future Progress,“ *Yearbook of Medical Informatics,* pp. 38-52, 8 2017.

[2] E. A. W. O. Hackle, „SPIRIT: Systematic Planning of Intellifent Reuse of Integrated Clinical Routine Data,“ *Methods of Information in Medicine,* pp. 114-124, 2 2016.

[3] M. Šavel, „Open Pseudonymised Health Data and Distributed Computing e-Infrastructure for Medical Science and Clinical Practice (OPEN),“ pp. 6-25, 29 3 2017.

[4] R. Copeland, „Google’s ‘Project Nightingale’ Gathers Personal Health Data on Millions of Americans,“ The Wall Street Journal, 11 11 2019. [Võrgumaterjal]. Available: https://www.wsj.com/articles/google-s-secret-project-nightingale-gathers-personal-health-data-on-millions-of-americans-11573496790.

[5] „eDelivery: Exchange data and documents securely and reliably,“ CEF Digital, [Võrgumaterjal]. Available: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eDelivery.

[6] „SAT for eID,“ European Union, 2017. [Võrgumaterjal]. Available: https://joinup.ec.europa.eu/release/eid-sat/v101.

[7] „eSignature,“ [Võrgumaterjal]. Available: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eSignature.

[8] „openEHR Architecture Overview,“ openEHR, [Võrgumaterjal]. Available: https://specifications.openehr.org/releases/BASE/Release-1.0.3/architecture_overview.html. [Kasutatud 2020].

[9] T. Beale, „Archetypes: Constraint-based Domain Modelsfor Future-proof Information Systems,“ %1 *OOPSLA 2002,* 2002.

[10] „openEHR: Archetype Technology Overview,“ openEHR, [Võrgumaterjal]. Available: https://specifications.openehr.org/releases/AM/latest/Overview.html.

[11] „Wikimedia,“ [Võrgumaterjal]. Available: https://upload.wikimedia.org/wikipedia/commons/7/73/Openehr_block_diagram.png?1589184374157.

[12] „FHIR Overview,“ HL7, [Võrgumaterjal]. Available: https://www.hl7.org/fhir/overview.html.

[13] „FHIR Guide to Designing Resources,“ [Võrgumaterjal]. Available: https://wiki.hl7.org/index.php?title=FHIR_Guide_to_Designing_Resources.

[14] K. Truu, „Mobiilse iseteenindusplatvormi loomine,“ pp. 22-23, 2018.

[15] T. Cerny, M. J. Donahoo ja M. Trnka, „Contextual Understanding of Microservice Architecture: Current and Furute Directions,“ *Appliced Computing Review,* pp. 31-32, 12 2018.

[16] P. Pazos, „github,“ [Võrgumaterjal]. Available: https://github.com/ppazos/cabolabs-ehrserver. [Kasutatud 2020].

[17] „HAPI FHIR,“ [Võrgumaterjal]. Available: http://hapifhir.io. [Kasutatud 5 4 2020].

[18] P. Pazos, „Github,“ [Võrgumaterjal]. Available: https://github.com/ppazos/cabolabs-ehrserver/blob/master/README.md.

[19] „The pros and cons of Cloud vs. in-house Servers,“ [Võrgumaterjal]. Available: https://sysgen.ca/cloud-vs-in-house-servers/. [Kasutatud 2020].

[20] G. H. Ganser ja P. Hewett, „An Accurate Substitution Method for Analyzing Censored Data,“ *Journal of Occupational and Environmental Hygiene,* pp. 233-244, 17 2 2010.

[21] K. Muralidhar ja R. Sarathy, „Data Shuffling: A New Masking Approach for Numerical Data,“ *Management Science,* pp. 658-670, 5 2006.

[22] A. J. Menezes, P. C. van Oorschot ja S. A. Vanstone, „Digital Signatures,“ %1 *Handbook of Applied Cryptography*, 1996, p. Ch 11.

[23] P. Gutmann, „Secure Deletion of Data from Magnetic and Solid-State Memory,“ *USENIX Security Symposium Proceedings,* 22 6 1996.

[24] „Google DeepMind NHS app test broke UK privacy law,“ BBC, 3 7 2017. [Võrgumaterjal]. Available: https://www.bbc.com/news/technology-40483202.

[25] „Medium,“ [Võrgumaterjal]. Available: https://miro.medium.com/max/1400/1*rHznJLgIXt63JWvo5Re4fQ.png.

[26] „Medium,“ [Võrgumaterjal]. Available: https://miro.medium.com/max/1400/1*yzYjtRViCDeWyhGnVzsUYw.png.

[27] T. Joosep, „GitLab,“ [Võrgumaterjal]. Available: https://gitlab.cs.ttu.ee/tajoos/open.

[28] J. Benet, „IPFS - Content Addressed, Versioned, P2P File System“.

[29] E. Allwell-Brown, „A Comparative Analysis of HL7 FHIR and openEHR for Electronic Aggregation, Exchange and Reuse of Patient Data in Acute Care,“ 2016.

[30] S. Wu ja J. Du, „Electronic medical record security sharing model based on blockchain,“ %1 *International Conference on Cryptography, Security and Privacy*, 2019.

[31] „Guardtime Health HSX: An API platform for distributed secure health appliations,“ Guardtime Health, 2020.

[32] „TensorFlow,“ Google, [Võrgumaterjal]. Available: https://www.tensorflow.org/overview.

[33] „PyTorch,“ [Võrgumaterjal]. Available: https://pytorch.org/.

[34] R. Orac, „Are you still using pandas for big data?,“ [Võrgumaterjal]. Available: https://towardsdatascience.com/are-you-still-using-pandas-for-big-data-12788018ba1a.

[35] „IMI Future Topics,“ Innovative Medicine Initiative, [Võrgumaterjal]. Available: https://www.imi.europa.eu/apply-funding/future-topics.

[36] T. Wade, „Refining Gold from Existing Data,“ *Current Opinion in Allergy and Clinical Immunology,* pp. 181-185, June 2014.

Lisa 1 – HL7 FHIR päringu näide


```
{
 "resourceType": "Bundle",
 "id": "bundle-response",
 "meta": {
  "lastUpdated": "2014-08-18T01:43:33Z"
 },
 "type": "transaction-response",
 "entry": [
  {
   "fullUrl": "http://example.org/fhir/Patient/12423",
   "resource": {
    "resourceType": "Patient",
    "id": "12423",
    "meta": {
     "versionId": "1",
     "lastUpdated": "2014-08-18T01:43:31Z"
    },
    "text": {
     "status": "generated",
     "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Some narrative</div>"
    },
    "active": true,
    "name": [
     {
      "use": "official",
      "family": "Chalmers",
      "given": [
       "Peter",
       "James"
      ]
     }
    ],
    "gender": "male",
    "birthDate": "1974-12-25"
   },
   "response": {
```

```
      "status": "201 Created",
      "location": "Patient/12423/_history/1",
      "etag": "W/\"1\"",
      "lastModified": "2014-08-18T01:43:33Z",
      "outcome": {
        "resourceType": "OperationOutcome",
        "issue": [
          {
            "severity": "warning",
            "code": "K35",
            "details": {
              "text": "Needs an operation"
            },
            "expression": [
              "Patient.managingOrganization"
            ]
          }
        ]
      }
    }
  ]
}
```