TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ergo Enn

TETRA RAADIOSIDE ARVUTI KASUTAJARAKENDUSE LOOMINE

bakalaureusetöö

Juhendaja: Toomas Lepikult
PhD
Kaasjuhendaja: Peeter Lump
MA

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: Ergo Enn

30.04.20

Annotatsioon

Käesoleva töö eesmärgiks on luua lahendus suhtlemiseks arvutiga üle TETRA raadiosidevõrgu. Hetkel turul saadaolevad lahendused on kulukad ja ei paku laiendusvõimalust, et ühildada raadioside muude infosüsteemidega.

Töös raames analüüsitakse TETRA raadioside keskjaama liidest ja leitakse puudused, mille lahendamiseks oleks vaja luua suhtlusvõimekus mõne välise liidesega, uuritakse erinevaid teeke ja raamistikke, millega antud probleemi oleks kõige mõistlikum lahendada ja luuakse nende analüüside põhjal kliendirakendused. Käesolevas töös ei käsitleta tarkvara loomist, millega kliendirakenduse välise teenuse liides peaks suhtlema.

Töö olulisemaks tulemuseks on kolme, turuhinnast märkimisväärselt soodsama, rakenduse arendamine, mis on ka kasutajasõbralikumad.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 20 leheküljel, 5 peatükki, 11 joonist.

Abstract

Development of computer user interface for Tetra radio communications system

The aim of current thesis is to find a solution for a software that communicates over TETRA radio network via computer. Currently available solutions are expensive and don't support the means of extensibility for connecting radiocommunication with other information systems.

This thesis analyzes the application programming interface of TETRA radiocommunication exchange and tries to find shortcomings that could use some external information sources to accomplish. Also, it analyzes different programming libraries and frameworks to find ones that would be most useful to fulfill given tasks. Finally, based on those analyzes, a client-side application was developed that allows user to communicate over TETRA radio network and has additional convenience features that would not be possible without interfacing some external information source. The development of said external information source is not in the scope of current thesis.

Most important outcome of this thesis is the successful development of three client-side applications that are significantly less expensive and more convenient for users than currently available software packages.

The thesis is in Estonian and contains 20 pages of text, 5 chapters, 11 figures,

Lühendite ja mõistete sõnastik

AMQP	(ingl k *Advanced Message Queuing Protocol*) avatud standard rakenduskihi suhtluse vahevarale
ETSI	Euroopa Telekommunikatsiooni Standardite Instituut
GSM	(ingl k *Global System for Mobile Communicatio*ns) mobiilside standard
IPC	(ingl k *Interprocess communication)* erinevate protsesside vaheline suhtlus
JSON	(ingl k *JavaScript Object Notation*) avatud standard inim- ja masinloetavalt andmete vahetuseks
REST	(ingl k *representational state trasfer*) tarkvara arhitektuuri stiil veebiteenustega automaatseks infovahetuseks
SSE	(ingl k *Server-Sent Events*) standard defineerimaks teenusepakkuja poolt saadetavate teadete voogu
TETRA	(ingl k. *Terrestrial Trunked Radio*) raadioside standard
WCAG	(ingl k *Web Content Accessibility Guidelines*) veebi sisu ligipääsetavuse juhend
WPF	(ingl k *Windows Presentation Framework*) kasutajaliideste loomise raamistik

Sisukord

Jooniste loetelu

1 Sissejuhatus

Enamikus maailma riikides on olemas mingil kujul hädaabi kõnedele reageerimise võimekus. Selle jaoks, et hädasolija abipalve jõuaks kõnekeskuse päästekorraldajalt reaalse päästjani, kes saaks inimest aidata, on talle vaja mingit moodi vastav info edastada. Euroopa riikides kasutatakse selleks TETRA (*Terrestrial Trunked Radio*) raadioside standardile vastavaid seadmeid. Sellised seadmed on olemuselt ja välimuselt väga sarnased tavalise mobiiltelefoniga, kuid nendele on loodud hulganisti lisaväärtusi, mis muudavad nende kasutamise rasketes ja kriitilistes tingimustes palju mugavamaks ja töökindlamaks kui tavalise mobiiltelefoniga. Näiteks võimaldab selline käsiraadiojaam suhelda ümbruskonna raadiojaamadega otse ilma tugijaamata, teha ühe nupuvajutusega grupikõnesid ja kuna need töötavad võrreldes hariliku (GSM) mobiilsidega väga madalal sagedusel, on nende leviala märkimisväärselt suurem [1]. Viimane on väga oluline väärtus, kuna paljud õnnetused juhtuvad kohtades, kuhu harilik mobiilside ei levi.

Euroopa liidus pole sisejulgeoleku valdkondade jaoks ühtset tarkvaralahendust, sest iga riigi infosüsteemid on erinevalt üles ehitatud vastavalt selle riigi seadusandlusele, majanduslikule võimekusele ja tehnoloogilise arengu tasemele. Küll aga on Euroopas lepitud kokku ühises raadioside standardis. Täna olemasolevad rakendused TETRA raadiosidega suhtlemiseks oskavadki ainult seda ja ühegi riigi oma infosüsteemiga need suhelda ei oska. Sellist võimalust pole arvatavasti üritatud luua kuna ühtne standard operatiivsündmuste juhtimise tarkvarale puudub ja paljud Euroopa liidu riigid pole oma infosüsteemidega veel sellises arengujärgus, et niisugust funktsionaalsust täna veel soovida.

Käesoleva diplomitöö eesmärgiks on luua arvutis kasutatav rakendus TETRA raadiosüsteemi võrgus suhtlemiseks ning operatiivsündmuste suhtluse haldamise mugandamiseks uurides raadioside keskjaama liidestusvõimekusi ja luua juurde liidesed, mis suhtleks väliste teenustega ja muudaks seeläbi rakenduse kasutamist lihtsamaks ja intuitiivsemaks.

Teema valikul otsustas autor rakenduse nullist loomise kasuks, kuna saadaolevad kommertslahendused ei võimalda muude teenustega liidestust ega võimalda seda ka juurde arendada lisamoodulina. Lisaks on need rakendused kallid ja nõuavad arvutile spetsiifilise riistvara paigaldust.

1.1 Ülesande püstitus

Luua töölaua rakendus, mis võimaldab üle TETRA raadioside:

- saata ja vastu võtta sõnumeid;
- helistada ja kuulata individuaalkõnesid;
- rääkida ja kuulata kõnegruppe
- järele kuulata raadiosides räägitut

Luua töölaua rakendus, mis võimaldab

- rääkida ühte kindlasse operatiivsündmuse alarmeerimise kõnegruppi
- hallata ühe kindla operatiivsündmuse alarmeerimise kõnegrupi liikmeid
- järele kuulata operatiivsündmuse alarmeerimise kõnegrupis räägitut
- vaadata operatiivsündmuse infot

2 Analüüs

2.1 TETRA raadioside standard

TETRA on ETSI (Euroopa Telekommunikatsiooni Standardite Instituut) standard üleeuroopalise professionaalse raadioside süsteemi jaoks [2]. See on peamiselt mõeldud kasutamiseks riiklike sise- ja välisjulgeoleku organisatsioonide sisemiseks suhtluseks, kuid osades riikides on see kasutusel ka erinevates industriaal- ja kommertsettevõtetes [3]. Üle maailma kasutab TETRA standardil süsteeme erinevatel otstarvetel üle 121 riigi [4]. TETRA raadioside sagedusvahemik on võrreldes hariliku mobiilsidega üle kahe korra madalam, jäädes Eestis 380-399,9 MHz juurde [5]. Hariliku mobiilside kõige madalam (GSM) sagedus Eestis on 876 MHz [5]. See tähendab, et raadioside ei ole häiritud hariliku mobiilside koormuse poolt ja madalam sagedus levib naturaalselt kaugemale, kattes sedasi soovitud pindala vähiksema pingutuse ja kuludega.

2.2 Lähteülesanne

TETRA raadioside keskjaam pakub liidest, mille kaudu saavad erinevad programmid anda käske keskjaamale. Liidese kaudu on võimalik liituda kõnegruppidega, saata sõnumeid, rääkida kõnegruppidesse jne. Liides ei toeta olulisi funktsioone: keskne autentimine, telefoniraamat, piiratud administreerimisõigused, raadiosidekõnede mugav järelkuulamine ja veel mõningaid väiksemaid mugavusfunktsioone. Nende funktsioonide võimaldamine nõuab uue tarkvara arendamist, mis suudab ühenduda TETRA raadioside keskjaamaga ning ka mõne eraldiseisva haldussüsteemiga, kust vastav info pärida.

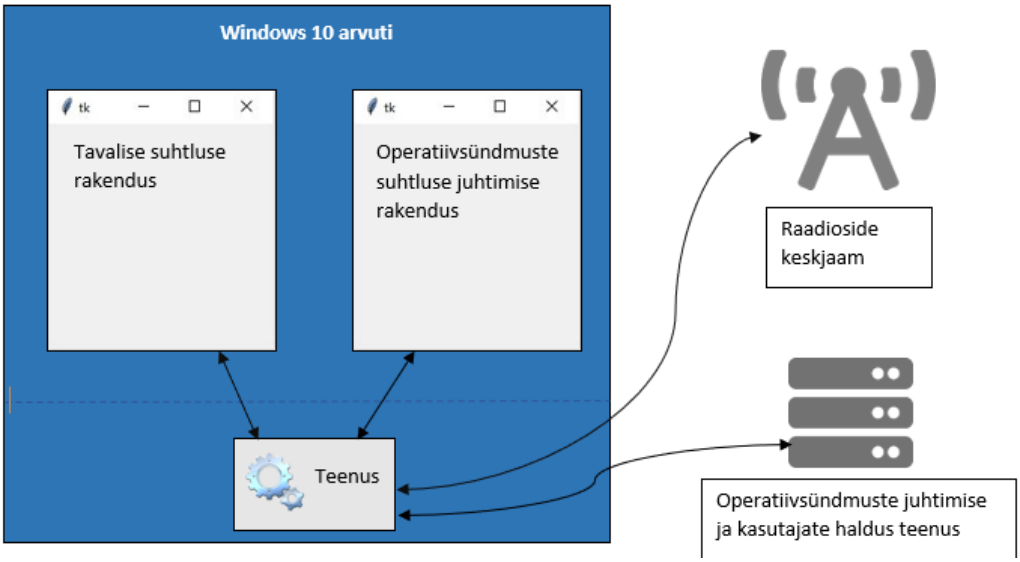
2.3 Süsteemi platvormi valik

Arenduse alguses tuli otsustada, mis operatsioonsüsteemi peal oleks rakenduse põhiline kasutajaskond. Kuna Euroopas on lauaarvutite operatsioonsüsteemidest Windowsi osakaal ~77%, siis võib eeldada, et rakendus leiaks kõige rohkem kasutust just sellel platvormil [6].

Ühe võimalusena oleks olnud rakendus arendada lokaalse veebiserverina ja kasutajaliidese kuvamiseks kasutada veebilehitsejat. See tähendab, et kogu rakendus on sõltuv kasutaja veebilehitsejast, mille tootjad võivad igal ajahetkel teha muudatusi, mis rikuvad rakenduse toimivuse. Varasemalt on selline olukord olnud Google Chrome veebilehitsejaga, millel otsustati blokeerida helide automaatne esitamine, mis omakorda tekitas tõrkeid erinevates veebirakendustes [7]. Sellest tulenevalt sai otsustatud, et rakendus tuleb arendada eraldiseisva kasutajaliidesena, et vältida liigsete riskide võtmist kontrollimatute sõltuvuste näol. Windowsi töölauarakenduste arenduseks on kõige rohkem arenenud ja kasutatud raamistik *Windows Presentation Framework* (WPF) [8], mis on kirjutatud C# programmeerimiskeeles. Kuna autoril on sellega ka varasem kogemus, siis otsustati rakendus just selle raamistiku baasil arendada.

2.4 Rakenduse arhitektuur

TETRA raadiosidet kasutatakse erinevatel otstarvetel ja erinevates olukordades. Kuna kõik kasutusalad ei vaja kõiki funktsionaalsuseid, saab rakenduse tükeldada loogilisteks osadeks. Rakenduse põhiosa saab suuresti jaotada kaheks nõrgalt sidustatud osaks: tavaline suhtlus ja operatiivsündmusele reageerimine. Tavaline suhtlus hõlmaks endast raadioside kanalitesse rääkimist, nende reaalajas- ja järelekuulamist, sõnumite saatmist ja vahetuid kõnesid teistele seadmetele. Operatiivsündmusele reageerimine hõlmaks endast ühe kindla kõnegrupi liikmete haldamist, sinna rääkimist ja sündmuse info kuvamist mugavamaks ning kiiremaks info edastamiseks. Nende osade omavaheline sidusus oleks minimaalne, sest ainus info, mida mõlemad peavad omama, on raadiojaama identifikaator, mille alt suheldakse. Sellest tulenevalt otsustati luua kaks eraldiseisvat rakendust, mis suhtlevad ühise Windows'i teenusega, mis omakorda suhtleb väliste liidestega (joonis 1).



Joonis 1 Rakenduse arhitektuur.

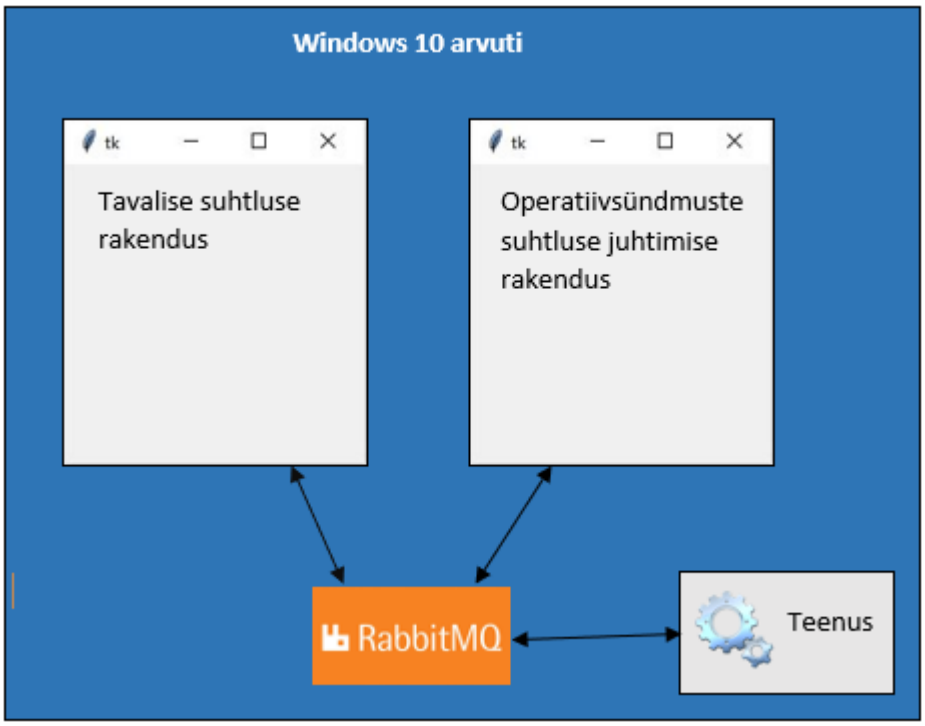
3 Teostus

3.1 NET raamistiku valik

C# programmeerimiskeeles on valida kahe baasraamistiku vahel. NET Framework on vanem ja töötab ainult Windows operatsioonsüsteemi peal. NET Core on uuem raamistik, mille baasil loodud rakendusi saab erinevates operatsioonsüsteemides jooksutada, kuid seda teatud mööndustega. Üheks selliseks mööndusek on töölaua rakendused, kuna nende kood on liiga platvormispetsiifiline. See tähendab, et NET Core mitme platvormi toe eelist ei saa täna veel ära kasutada. Lisaks sellele, et NET Core raamistik on uuem, võib tekkida olukord, kus mõni arendusprotsessi lihtsustav teek veel seda ei kasuta, mis välistab selle teegi kasutamise antud projektis. Neid aspekte arvesse võttes otsustas autor arendada tarkvara NET Framework raamistiku baasil.

3.2 Protsessidevaheline suhtlus

Rakenduse arhitektuuri järgi on vaja luua suhtlusvõimekus kolme eraldiseisva, kuid samal füüsilisel seadmel jooksva protsessi vahel (joonis 2). Kuna sellise probleemiga polnud autor varem kokku puutunud, pakkus diplomitöö juhendaja välja kaks lahendust. Esimene neist oli Windowsi IPC (ingl k *Interprocess communication*), mis toimib hästi riistvaralähedaselt ja võimaldab sama füüsilise masina peal töötavatel protsessidel omavahel suhelda. Teine pakkumine oli RabbitMQ, mis on sõnumite vahendusteenus ning, üks enamlevinud suhtluslahendus veebi(mikro)teenuste vaheliseks suhtluseks [9]. Kuigi mõlemad lahendused vastasid projekti nõudmistele, otsustas autor RabbitMQ kasuks, kuna tal oli sellega varasem kokkupuude ja kuna see teenus on paljude rakenduste poolt aktiivses kasutuses ja arenduses, siis on sellele rohkem abistavaid teeke ja väiksem tõenäosus tõrgeteks.



Joonis 2. Rakenduste vaheline suhtlus RabbitMQ'ga.

RabbitMQ on avatud lähtekoodiga sõnumite vahendusteenus, mis ehitati algselt vastavalt Advanced Message Queuing Protocol (AMQP) standardile [10]. Selle standardi kohaselt saab luua sõnumite vahetuse kanaleid (ingl k *exchange*) ja järjekordi, kus iga kanaliga saab seostuda null kuni mitu järjekorda ja iga järjekorraga null kuni mitu järjekorra sõnumite kuulajat, kusjuures iga järjekorra ja kanali vahelisel seosel peab olema defineeritud marsruutimise võti. Vastavalt deklareeritud kanali tüübile ja järjekordade marsruutimise võtmetele, otsustab RabbitMQ, millisesse järjekorda mingi sõnum suunata. Kanali tüübid on:

- otsene (ingl k *direct*), mis nõuab, et päringu marsruutimise võti oleks täpselt võrdne järjekorra võtmega, et sinna sõnumit saata;
- teemapõhine (ingl k *topic*), mis lubab päringu marsruutimise võtmes kasutada metamärke(ingl k. *wildcard*), et saada hajus vaste järjekorra võtmega;
- laialipaiskav (ingl k. *fanout*), mis saadab kõikidesse järjekordadesse kõik sõnumid olenemata marsruutimise võtmetest;
- päisepõhine (ingl k. *header*), mis lubab määrata sõnumitele ja järjekordadele marsruutimise võtme asemel null kuni mitu võti-väärtus paari ja sõnumid

saadetakse järjekordadesse, kus kõik sõnumis olevad võti-väärtuse paarid vastavad järjekorra võti-väärtus paaridele.

Kuna planeeritud rakenduse arhitektuuris on teenus, millega peab kahepoolselt suhtlema kaks töölauarakendust, siis otsustas autor kasutada teenusest kasutajarakenduste suunal teema põhist kanalit ja vastupidisel suunal otsest kanalit. Selline seadistus võimaldab teenusel suunata iga päringu vastuse tagasi ainult selle päringu sooritajale, kuid ilma kindla päringuta teavitused paisata kõigile. Otsene kanal teenuse suunal võeti kasutusele, kuna see võimaldab tulevikus lisada uusi teenuseid uute järjekordadega ja olemasolevad rakendused jäävad ilma uuendusi nõudmata tööle.

RabbitMQ ei piira, mis tüüpi sõnumi sisu on, seega oleks triviaalseim sõnumite sisus kasutada binaarkujul C# andmeobjekte. Selle asemel otsustas autor JavaScript Object Notation (JSON) kasuks, sest see võimaldab tulevikus lihtsamalt lisada uusi töölauarakendusi, mis võivad olla kirjutatud teistes programmeerimiskeeltes ja need suudaks ikka suhelda olemasoleva teenusega [11].

Projekti arendamise ja testimise käigus tekkisid RabbitMQ'ga mõned juhtumid, kus testmasinatesse paigaldatud RabbitMQ teenus ei olnud leitav projekti raames arendatud tarkvarade poolt, mis tähendas, et rakendused olid kasutuskõlbmatud. Lisaks puudub RabbitMQ'l sisseehitatud tugi kontrahtide jaoks, et suhtluskanalite sisule tugevad piirangud anda. Selle probleemi lahenduseks analüüsiti erinevaid lisa teeke nagu MassTransit, Rebus ja RawRabbit [12] [13] [14]. Need teegid võimaldasid küll suhtluskanaleid tugevalt kirjeldada, kuid nende kõigi eesmärk oli lahendada mikroteenustes tuntud lõpliku kooskõla printsiipi (ingl k eventual consistency) ja võimaldada süsteemide skaleerimist suurema läbilaskevõime nimel. Nendest võimalustest kumbki, aga pole antud projekti jaoks oluline faktor, vaid vaja oli pigem kaugprotseduurikutse võimekust ja sõnumite valikulist laialipaiskamist kõigile kuulajatele. Neid aspekte arvesse võttes otsustas autor luua uue teegi, mida ta kasutas oma projektides ja mis võimaldas projektis soovitavat suhtlust üle RabbitMQ suhtlusteenuse.

Projekti arendamise ajal, kuid mitte antud projekti raames, avastas ja tutvus autor Google poolt arendatava gRPC raamistiku, mis toetab tugevate suhtlusprotokollide defineerimist (joonis 3) ja kaugprotseduuride kutseid [15]. Lisaks, kuna tegu on otseselt

kaugprotseduurikutse raamistikuga, võib eeldada, et see on paremini optimeeritud antud tegevuseks kui üldotstarbeline sõnumivahetusteenus. Seega kui autor seisaks täna sarnase valiku ees, otsustaks ta tõenäoliselt gRPC kasuks.

```
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}
message HelloRequest {
  string name = 1;
}
message HelloReply {
  string message = 1;
}
```

Joonis 3 Näide, kuidas gRPC puhul liidesed kirjeldatakse [16].

3.3 Kasutajaliidese disain

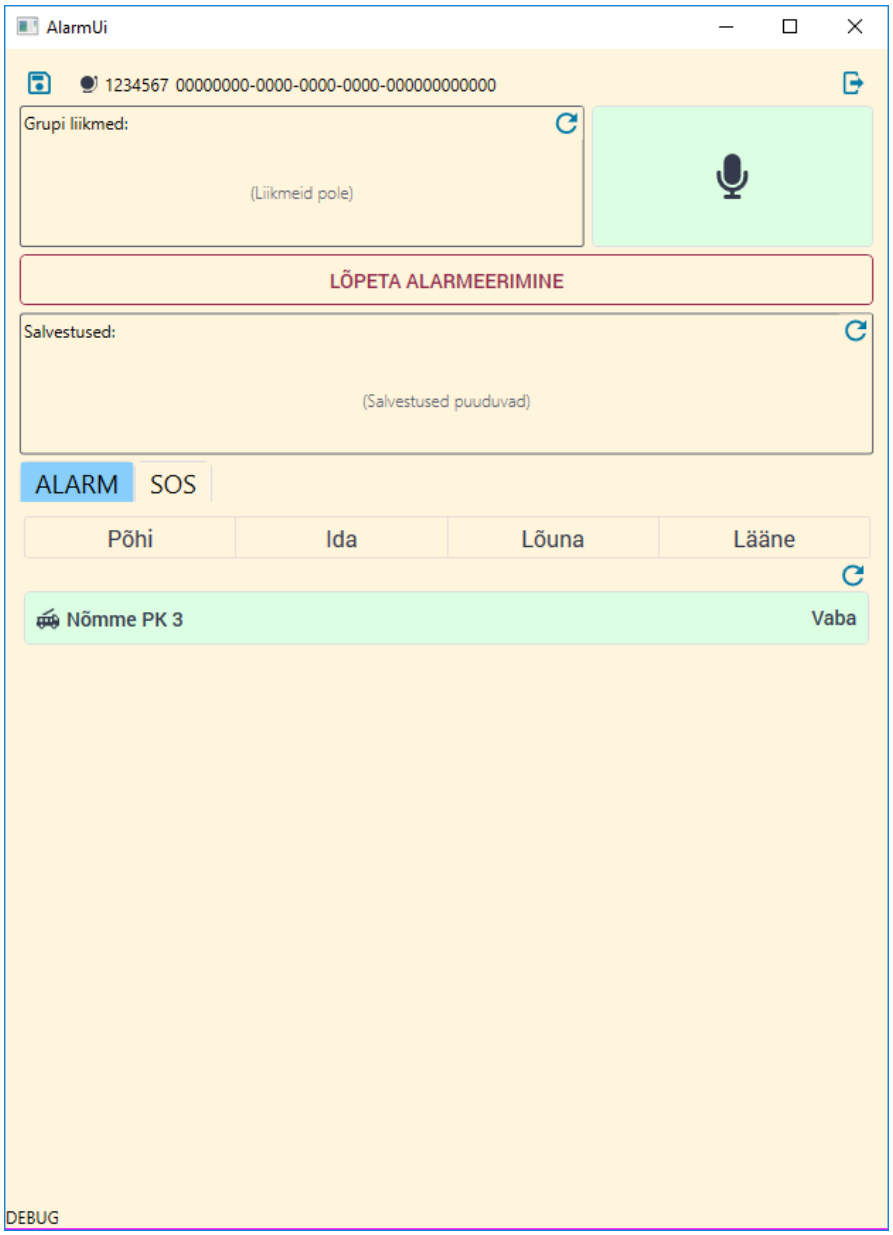
TETRA süsteemide põhikasutajad on erinevate riikide sisejulgeoleku töötajad. See tähendab, et rakenduse põhikasutajaskond soovib tõenäoliselt rakendust kasutada ööpäevaringselt. Selle mugavdamiseks otsustas autor mitte kasutada standardset valgel taustal must tekst värviskeemi, vaid valis midagi silmasõbralikumat. Järgides WCAG 2.0 nõudeid teksti kontrastile, tuli leida taustavärvi ja teksti värvi kombinatsioon, mille kontrasti suhe ei oleks alla 4.5:1 [17]. Kuna inimestel on mugavam lugeda heledal taustal tumedat teksti [18], kuid samas pimedas võib puhas valge liiga ere lugemiseks olla, otsustas autor kasutada beeži (#fff5dc) tausta ja tumehalli (#34394c) teksti. Erinevate väljade esiletõstmiseks valiti tasutavärvi topelt komplementaarvärve (joonis 4), sest need on sama küllastumuse ja helendusega, kuid erineva värvitooniga, jättes sedasi värvid omavahel harmooniasse.

Joonis 4 Alustoon, põhitoon ja alustoonist tuletatud topelt komplementaarvärvid.

Kasutajaliideseid, mida selle töö raames luuakse, on kaks: tavalise raadioside juhtimise rakenduse kasutajaliides (joonis 4) ja operatiivraadioside haldamise kasutajaliides (joonis 5). Nende mõlema lihtsamaks stiliseerimiseks lõi kasutaja uue stiili teegi, mis on üles laetud pakettide halduskeskkonda ja mida mõlemad kasutajaliidesed ning ka

tulevikus loodavad kasutajaliidesed saavad kasutada, et järgida rakenduse ühtset kujundust. Teek sisaldas endast kõiki kasutajaliidese komponentide stiile ning ka värvipaletti. Ikoonide jaoks kasutas autor MahApps.Metro.IconPacks teeki koos Google Material Design ja Font Awesome ikoonidega [19] [20] [21].

Joonis 5 Raadioside juhtimise kasutajaliidese näide.



Joonis 6 Operatiivraadioside juhtimise kasutajaliidese näide.

3.4 Reaktiivne kasutajaliides

WPF raamistik on arendatud staatilisi või harva muutuvaid elemente silmas pidades. Iga soov muuta vaate sisu nõuab käsitsi raamistiku teavitamist, et see teaks vaadet värskendada (joonis 8). Kui tegu on ühe suure staatilise andmeobjekti kuvamisega, siis on selline käitumisviis mõistlik, kuid kuna käesolevas projektis on tegemist väga suure hulga kiiresti muutuvate andmeobjektidega, tuleb probleemile läheneda teistmoodi. Selle jaoks on olemas spetsiaalsed teegid ReactiveUI ja DynamicData, mis teevad vaikimisi igast muutujast jälgitava muutuja, et kasutajaliidese teavitamist ei peaks käsitsi programmeerima (joonis 7). Samuti annavad need erinevaid abimeetodeid

andmete reaalajas filtreerimiseks, grupeerimiseks ja partii muudatuste kohta ühe korra teavitamiseks [22] [23]. See lihtsustab märkimisväärselt arendaja tööd ja vähendab tarkvara võimalike ootamatute käitumiste tekkimist arendaja rutiinist tekkivate vigade tõttu. Lisaks teevad teegid rakenduse mugavamalt testitavaks, sest WPF raamistiku sisseehitatud teavitamine käib sõnede kaudu, kuid teekide kaudu teavitus käib tugevalt tüübitud muutuja viidete kaudu.

```
public TalkGroupMembersListVm(ITalkGroupMembersService talkGroupMembersServic
e, IStationsService stationsService)
{
    talkGroupMembersService.Members
                           .Connect()
                           .Transform(member => new TalkGroupMemberVm(member
,stationsService))
                           .ObserveOn(Dispatcher.CurrentDispatcher)
                           .Bind(out _members)
                           .DisposeMany()
                           .Subscribe();
}

private ReadOnlyObservableCollection<TalkGroupMemberVm> _members;
public ReadOnlyObservableCollection<TalkGroupMemberVm> Members => _members;
```

Joonis 7 Kõnegrupi liikmete kuvamine DynamicData teegiga.

```
public TalkGroupMembersListVm(ITalkGroupMembersService talkGroupMembersServic
e, IStationsService stationsService)
{
    Members = new ObservableCollection<TalkGroupMemberVm>(talkGroupMembersSer
vice.Members);

    talkGroupMembersService.OnTalkGroupMembersUpdated += (sender, updatedMemb
ers) => {
        Application.Current.Dispatcher.Invoke(()=>{
            foreach (var updatedMember in updatedMembers)
            {
                var member = Members.FirstOrDefault(member=>member.Id==update
dMember.Id);
                if(member != null){
                    Members.Remove(member);
                }
                Members.Add(new TalkGroupMemberVm(updatedMember,stationsServi
ce));
            }
        });
    };
    talkGroupMembersService.OnTalkGroupMembersRemoved += (sender, removedMemb
ers) => {
        Application.Current.Dispatcher.Invoke(()=>{
            Members.RemoveAll(member=>removedMembers.Any(removedMember=>membe
r.Id==removedMember.Id));
        });
    };
    talkGroupMembersService.OnTalkGroupMembersAdded += (sender, newMembers) =
> {
        Application.Current.Dispatcher.Invoke(()=>{
            foreach (var newMember in newMembers)
            {
                Members.Add(new TalkGroupMemberVm(newMember,stationsService))
;
            }
        });
    };
}

public ObservableCollection<TalkGroupMemberVm> Members { get;set; }
```

Joonis 8 Kõnegrupi liikmete kuvamine reaktiivsete teekideta.

3.5 Teenus

Nagu algselt mainitud, koosneb kogu arendatav komplekt kolmest põhikomponendist. Üheks nendest komponentidest on Windows'i teenus. Teenus on arendatud Topshelf raamistiku baasil, sest see võimaldab rakendust mugavalt arenduskeskkonnas või konsoolirakendusena jooksutada, kuid samas toetab ka teenusena installeerimise

võimekust [24]. Tavaline System.ServiceProcess.ServiceBase raamistikul arendatud rakendus võimaldaks ainult teenuse installeerimist, mis teeb tõrgete otsimise märkimisväärselt ebamugavamaks.

3.6 Suhtlust teenusevälise liidesega

Teenus suhtleb väliste liidestega üle REST (ingl k *representational state trasfer*) päringute. Kuna rakenduse loomise hetkel ei suudetud leida ühtegi REST päringute teeki, mis võimaldaks muutuvate sessioonivõtmete kasutamist, siis tuli see alguses lahendada suure hulga duplikaatkoodiga iga liidese lõpp-punkti kohta. Sellist lahendust on raske testida ja lõhub ainsa vastutuse printsiipi, tegeledes korraga veebiaadressi koostamisega, päringu mudeli tekstikujule muutmisega ja vastuse tekstikuju andmeobjektiks muutmisega (joonis 9). Hilisemate otsimiste käigus leidis autor teegi RESTease, mis dokumentatsiooni järgi võimaldab antud probleemi lahendada palju abstraktsemalt ja testitavalt võimaldades luua ühe liidese klassi, mis defineerib ära kõik veebiteenuse lõpp-punktid ja abstraheerib arendaja jaoks kõik vahepealsed ebaolulised toimingud. Selle käigus muudetakse nähtavaks ainult üks meetod, millele lähevad päringu parameetrid sisse ja tulevad andme mudelid välja (joonis 10) [25].

```
public Task<SearchResult> SearchAsync(SearchParams param){
    string searchParametersJson = JsonConvert.SerializeObject(param);
    string url = this.baseUrl + "search?params=" + HttpUtility.UrlEncode(sear
chParametersJson);
    string response = await this.httpClient.DownloadStringTaskAsync(url);
    return JsonConvert.DeserializeObject<SearchResult>(response);
}
```

Joonis 9 Praegune lahendus päringute tegemiseks.

```
public class SearchParams
{
    public string Term { get; set; }
    public string Mode { get; set; }
}

public interface ISomeApi
{
    [Get("search")]
    Task<SearchResult>
SearchAsync([Query(QuerySerializationMethod.Serialized)] SearchParams param);
}

ISomeApi api = RestClient.For<ISomeApi>("http://api.example.com");
// Requests http://api.example.com/search?params={"Term": "foo", "Mode":
"basic"}
await api.SearchAsync(new SearchParams() { Term = "foo", Mode = "basic" });
```

Joonis 10 RESTease teegiga sama päringu koostamine.

Kuna tegu on reaalajasüsteemiga ja REST päringud toetavad ainult ühepoolset (klient küsib serverilt) pärimist, otsustas autor kasutusele võtta Server Sent Events (SSE), et võtta vastu välise liidese teateid kui see peab vajalikuks kasutajaliidest mingist sündmusest teavitada. SSE kasuks otsustas autor, kuna süsteemid, millega rakendus peab pärast suutma ühilduda, võivad olla arendatud raamistikel, mis ei toeta uuemaid suhtluskanali meetodeid ja SSE standard on eksisteerinud juba aastast 2006 [26].

3.7 TETRA keskjaama liidesega suhtlemine

TETRA standardile vastav keskjaam koos tarkvaraga võib olla toodetud erinevate firmade poolt. Eestis kasutusel olev Airbus TETRA süsteem kasutab suhtluseks *Microsoft Component Object Model* liidest [27]. Kogu suhtlus keskjaamaga toimub asünkroonselt, mis tähendab, et päring saadetakse ühe liidese kanali kaudu, kuid vastus ei tule samast kanalist, vaid määramata aja pärast hoopis teise kanali kaudu. Selline käitumine on kasutajaliidese jaoks kohmakas, mistõttu loodi lahendus (joonis 11), mis teeb päringu ja ootab teisest kanalist vastust ühe sama asünkroonse meetodi sees, kasutades selle jaoks C# System.Threading.SemapthoreSlim klassi.

```
public async Task<TResponse> Execute<TRequestParams, TSuccessfulRequest>(
TRequestParams request, Action<EventHandler<TSuccessfulRequest>>
successfulRequestHandlerSubscription,Action<EventHandler<TSuccessfulRequest>>
successfulRequestHandlerUnsubscription, Task<short?> mainMethod){
    var signal = new SemaphoreSlim(0);
    short? cookie = 0;
    GetRadioSubscribersConfirmationEventArgs result = null;
    int methodStatus = (int)TcsMethodStatusValues.OK;
    void SuccessHandler(object sender, TSuccessfulRequest args){
        if (args.Cookie == cookie.Value){
            result = args;
            signal.Release(1);
        }
    }
    void FailedHandler(object sender, GeneralConfirmationEventArgs args){
        if (args.Cookie == cookie.Value){
            methodStatus = args.MethodStatus;
            signal.Release(1);
        }
    }
    successfulRequestHandlerSubscription(SuccessHandler);
    this.proxy.GeneralConfirmation += FailedHandler;
    try{
        cookie = await mainMethod;

        var isSignaled = await signal.WaitAsync(TimeSpan.FromSeconds(10));

        if (methodStatus != (int)TcsMethodStatusValues.OK){
            throw new Exception($"Request failed with status: {methodStatus.H
umanizeMethodStatus()}");
        }
    }
    catch{
        signal.Release(1);
        throw;
    }
    finally{
        successfulRequestHandlerUnsubscription(SuccessHandler);
        this.proxy.GeneralConfirmation -= FailedHandler;
    }
    return result;
}
```

Joonis 11 Mitut kanalit kasutava sõnumi asünkroonseks funktsiooniks tegemise meetod.

3.8 Logimine

Arvestades rakenduse hajusat arhitektuuri, on korralik logimine arenduse protsessis ja hiljem vigade lahendamisel äärmiselt oluline osa antud projektist. NET Frameworkile on loodud mitmeid erinevaid logimise teeke, millel saab seadistada erinevaid logimise tasemeid ja väljundi liike. Autor otsustas antud projektis kasutada NLog teeki, kuna see on jõudlustestide andmetel kõige kiirem ja võimaldab kahte põhilist logimise väljundi liiki, mida autor tundis kõige enam vajavat [28] [29]. Need on faili logimine ja reaalaja logivaatlusrakendusse logi saatmine. Esimene on oluline, et kasutajad saaks vigade korral edastada oma arvutis jooksutatud rakenduse logifaili ja arendaja saaks selle põhjal vea põhjust otsida ja viga parandada. Teine on oluline arenduse ajal süsteemide omavahelise suhtluse reaalajas analüüsimiseks, sest rakenduse arenduse käigus mõistis autor, et enamik vigadest tekib ebakorrektse rakenduste omavahelise suhtluse tagajärjel.

3.9 Turvalisus

TETRA raadioside on paljudes riikides kasutusel sise- ja välisjulgeoleku valdkondades, kus on tihti erinevad ranged turvanõuded seoses digitaalse andmeside suhtluse jaoks. Kuna erinevatel riikidel on erinevad nõuded ja lahendused infosüsteemide turvalisuse tagamiseks, ei saa antud töö käigus loodud rakendusele luua ühtegi spetsiifilist lisa turvalisust pakkuvat võimekust, kuna see võib piirata mõne teise riigi või valdkonna süsteemi peal antud rakenduse kasutamist. Seega on rakendust mõeldud kasutada virtuaalses privaatvõrgus, millel on ligipääs soovitud TETRA keskjaama liidesele. Vajadusel saab kerge vaevaga arendada olemasolevatele kasutajaliidestele uue teenuse, mis suhtleb kliendi nõuetele vastavaid protokolle kasutades väliste teenustega, kuid lokaalses masinas RabbitMQ kaudu.

4 Töö tulem

4.1 Töö reaalne tulem

Diplomitöö käigus arendas autor edukalt kolm rakendust, mis on omavahel nõrgalt hajusalt seotud. Raadioside juhtimise rakendus võimaldab: saata ja vastu võtta sõnumeid, teha ja vastata individuaalkõnedele, suhelda ja kuulata vabalt valitud kõnegruppe, otsida kontakte telefoniraamatust, järelkuulata gruppidesse räägitut. Operatiivraadioside haldusrakendus võimaldab: rääkida ühte kindlasse kõnegruppi, lisada ja eemaldada sellest kõnegrupist kindlaid liikmeid, järelekuulata antud gruppi räägitut, jälgida välisest liidesest tulevat operatiivsündmuse infot.

4.2 Töö majanduslik tulem

Diplomitöö kirjutamise hetkeks oli autor süsteemi arendusse panustanud üle 750 tunni. Eesti statistikaameti andmetel on Eestis tarkvaraarendaja mediaan brutopalk 2820 eurot kuus, mis tööandjale tähendab ~3800 eurot [30]. Eeldades, et kuu on võrdne 160 töötunniga, tuleb tööandjal selle eest tasuda ~18 000€. Eesti riigihange sarnase toote ja sellega koos vaja mineva spetsiifilise riistvara soetamisega läks maksma 3 000 000 eurot [31].

5 Kokkuvõte

Käesoleva diplomitöö eesmärgiks oli luua kasutajarakendus TETRA raadioside juhtimiseks ja operatiivsündmuste raadioside haldamiseks kasutades selleks olemasoleva raadioside keskjaama liideseid ja luues liidestusvõimekuse väliste teenustega, võimaldamaks rakenduse mugavamat kasutamist operatiivsündmuste lahendamisel.

Analüüsi osas võrdles autor erinevaid platvorme ja teeke, mille peal oleks võimalik antud ülesannet lahendada ja millistel operatsioonisüsteemidel peaks rakendus kasutatav olema. Püstitati esialgne süsteemi arhitektuur ja tutvustati sisejulgeoleku raadioside tagamaid.

Praktilises osas valis autor välja teegid, millega erinevaid probleeme lahendada. Kuna töö käigus avastati osadele tegevustele paremad teegid kui algselt valitud, siis autor tõi esile paremad teegid ja põhjendas nende kasulikke aspekte võrreldes käiku läinud lahendusega.

Diplomitöö käigus valminud lahendus on autori hinnangul heaks baasiks erinevate riikide operatiivraadioside süsteemide kasutamise mugavdamiseks ja lisafunktsionaalsustega laiendamiseks. Lisaks saavutati tööga märkimisväärne rahaline kokkuhoid, sest Eesti riigihange sarnase süsteemi omandamiseks maksis 3 000 000 eurot, kuid autori loodud lahenduse hind ulatub täna 20 000 euro hinnaklassi.

6 Kasutatud kirjandus

[1] TCCA, [Võrgumaterjal]. Available: https://tcca.info/tetra/tetra-standard/technology-benefits/. [Kasutatud 28 03 2020].

[2] European Telecommunications Standards Institute, „ETSI EN 300 392-2 v3.2.1," [Võrgumaterjal]. Available: https://www.etsi.org/deliver/etsi_en/300300_300399/30039202/03.02.01_60/en_30039202v030201p.pdf. [Kasutatud 28 03 2020].

[3] „Entropia - What are the subscriptions," [Võrgumaterjal]. Available: https://www.entropia.eu/en/item_what-are-the-subscriptions_66.aspx. [Kasutatud 28 03 2020].

[4] „TETRA Industry Group," [Võrgumaterjal]. Available: https://web.archive.org/web/20120313035919/http://www.tetrahealth.info/worldCountries.htm. [Kasutatud 23 03 2020].

[5] E.-. j. infotehnoloogiaminister, „Eesti raadiosagedusplaan," 28 01 2019. [Võrgumaterjal]. Available: https://www.riigiteataja.ee/aktilisa/1250/1201/9006/MKM_22012019_m6lisa1.pdf. [Kasutatud 29 03 2020].

[6] StatCounter, „Desktop Operating System Market Share Europe," [Võrgumaterjal]. Available: https://gs.statcounter.com/os-market-share/desktop/europe. [Kasutatud 28 03 2020].

[7] S. Gibbs, „Google tried to block autoplay videos on Chrome. But it broke apps and games," The Guardian, 16 mai 2018. [Võrgumaterjal]. Available: https://www.theguardian.com/technology/2018/may/16/google-chrome-autoplay-blocking-apps-games-break-loud-videos. [Kasutatud 17 03 2020].

[8] „9 Must Decisions in Desktop Application Development for Windows," [Võrgumaterjal]. Available: https://michaelscodingspot.com/9-must-decisions-in-desktop-application-development-for-windows/. [Kasutatud 18 03 2020].

[9] Pivotal Software, [Võrgumaterjal]. Available: https://www.rabbitmq.com/. [Kasutatud 28 03 2020].

[10] Pivotal Software, „RabbitMQ Protocols," [Võrgumaterjal]. Available: https://www.rabbitmq.com/protocols.html. [Kasutatud 18 03 2020].

[11] ecma international, „The JSON Data Interchange Syntax," [Võrgumaterjal]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. [Kasutatud 24 03 2020].

[12] C. Patterson, „MassTransit," [Võrgumaterjal]. Available: https://masstransit-project.com/. [Kasutatud 12 04 2020].

[13] M. H. Grabe. [Võrgumaterjal]. Available: https://github.com/rebus-org/Rebus. [Kasutatud 26 03 2020].

[14] P. Dahlman. [Võrgumaterjal]. Available: https://github.com/pardahlman/RawRabbit. [Kasutatud 22 03 2020].

[15] gRPC, „gRPC Guide,“ [Võrgumaterjal]. Available: https://grpc.io/docs/guides/. [Kasutatud 26 03 2020].

[16] gRPC, „gRPC Quick Start,“ [Võrgumaterjal]. Available: https://grpc.io/docs/quickstart/csharp/. [Kasutatud 25 03 2020].

[17] „WCAG 2.0 Contrast,“ [Võrgumaterjal]. Available: https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html. [Kasutatud 20 03 2020].

[18] D. &. C. C. R. Bauer, „Improving the legibility of visual display units through contrast reversal,“ Taylor & Francis, London, 1980.

[19] MahApps, „MahApps.Metro.IconPacks,“ [Võrgumaterjal]. Available: https://github.com/MahApps/MahApps.Metro.IconPacks. [Kasutatud 25 03 2020].

[20] Google, „Material Design Icons,“ [Võrgumaterjal]. Available: https://github.com/google/material-design-icons. [Kasutatud 26 03 2020].

[21] Fonticons, Inc., „Font Awesome,“ [Võrgumaterjal]. Available: https://fontawesome.com/. [Kasutatud 26 03 2020].

[22] [Võrgumaterjal]. Available: https://reactiveui.net/. [Kasutatud 28 03 2020].

[23] R. Pheasant. [Võrgumaterjal]. Available: https://dynamic-data.org/. [Kasutatud 26 03 2020].

[24] Topself, „Topself,“ [Võrgumaterjal]. Available: http://topshelf-project.com/. [Kasutatud 25 03 2020].

[25] A. Male. [Võrgumaterjal]. Available: https://github.com/canton7/RestEase. [Kasutatud 20 03 2020].

[26] W3C, [Võrgumaterjal]. Available: https://www.w3.org/TR/eventsource/. [Kasutatud 28 03 2020].

[27] Airbus Defence and Space, „TCS API Description,“ 2015.

[28] J. Marsh, „Benchmarking 5 popular .NET logging libraries,“ [Võrgumaterjal]. Available: https://www.loggly.com/blog/benchmarking-5-popular-net-logging-libraries/. [Kasutatud 06 04 2020].

[29] NLog, „NLog,“ [Võrgumaterjal]. Available: https://nlog-project.org/. [Kasutatud 24 03 2020].

[30] Eesti Statistikaamet, „Ametite kuupalgad 2019,“ [Võrgumaterjal]. Available: https://andmestikud.stat.ee/ametipalk/. [Kasutatud 28 03 2020].

[31] Rahandusministeerium, [Võrgumaterjal]. Available: https://riigihanked.riik.ee/rhr-web/#/procurement/726435/documents?group=B. [Kasutatud 28 03 2020].