TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Oliver Vanamets 186015IAAB

Mikroteenuste ja konteinerarhitektuuri juurutamine riigiasutuses TEHIKu näitel

Bakalaureusetöö

Juhendaja: Siim Vene
MSc,
Marko Valing
Rakenduslik
kõrgharidus

Tallinn 2021

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: Oliver Vanamets

29.03.2021

Annotatsioon

Käesoleva lõputöö eesmärk on analüüsida uudse tarkvaraarenduse paradigma ehk mikroteenuste arhitektuuri otstarbekust avaliku sektori asutuse kontekstis. Aluseks on võetud kogemused mujalt maailmast ning seni loodud mikroteenuste arhitektuurile rajatud teenuste käekäik.

Töö tulemusel leiti, et kõnealune uus tarkvaraarenduse meetod on sõltumata mõningatest kitsaskohtadest sobilik ning isegi eelistatud lahendus tulevaste riiklike infosüsteemide loomise juures. Uudse lähenemisega saadav kasu hilisema halduse ja edasiarenduste lihtsuse arvelt teeb mikroteenuste arhitektuurist avalikus sektoris efektiivse töövahendi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 5 peatükki, 2 joonist.

Abstract

Implementing Microservices and Container Architecture in a Government Institution on the Example of TEHIK

The aim of this thesis is to analyse a relatively new software development method that divides large monolithic services into smaller, loosely coupled components called microservices and to evaluate it's practicality in the context of a government institution.

First part of the thesis describes the problem that many organisations within the ICT sector are experiencing and that is the choice between building software on top of traditional monolithic architecture or choosing a modern microservices approach. In the second chapter the author moves on to describe both of those options in greater detail.

The third part of this thesis describes a typical microservices adoption process based on the findings from other research papers on the same topic. Common problems and secondary effects that might occur within such a process are also discussed.

In the fourth chapter, the author describes the current state of microservices adoption in TEHIK and their experience so far. This chapter also includes the practical part of setting up a new service based on the microservices architecture. The last part of this thesis summarises the findings and describes the future of microservices within the Estonian public sector.

The thesis is in Estonian and contains 28 pages of text, 5 chapters, 2 figures.

Lühendite ja mõistete sõnastik

DevOps	Töövõtete kogum toetamaks tarkvaraliste lahenduste kiiret progresseerumist arendusjärgust toodanguni
GitLab CI/CD tarneahel	GitLab *Continuous Integration/Continuous Delivery* tarneahel ehk GitLab versioonihaldusprogrammi poolt käitatav skriptide kogum teostamaks programmikoodi konteineritesse pakendamist automatiseeritult
konteiner	Rakenduse (või mikroteenuse) standardne pakend, mis sisaldab programmikoodi, sõltuvusi, teeke või muid abivahendeid, mis rakenduse käivitamiseks vajalik
koodibaas	Tarkvara programmikoodi kogum
koormusjaotur	Tarkvaraline või riistvaraline lahendus jaotamaks kasutuskoormust mitme sarnase funktsionaalsusega süsteemi või selle osa vahel
Kubernetese klaster	Serverite kogum, milles käitatakse konteinereid
Riigipilv	Majandus- ja Kommunikatsiooniministeeriumi poolt üles seatud avaliku sektori asutustele kasutamiseks mõeldud pilvelahendus
Silt	*Tag* ehk lühike sümbolite jada juhtimaks automatiseeritud protsesside käivitamist
Skript	Automatiseeritult teostatavaid toiminguid kirjeldav käsujada
TEHIK	Tervise ja Heaolu Infosüsteemide Keskus

Sisukord

Jooniste loetelu

Sissejuhatus

Infotehnoloogilisi teenuseid võib jagada nende arhitektuurilise ülesehituse järgi kaheks: monoliitsel ning mikroteenuste arhitektuuril põhinevad teenused. Mikroteenuste arhitektuuri kasutamine tänapäevaste IT-teenuste rajamisel on ülemaailmne tõusev trend ning uute teenuste arhitektuurilise kavandamise algusjärgus tuleb vastu võtta põhimõtteline otsus kummagi meetodi kasuks [1].

Käesolevas töös analüüsitakse mikroteenuste ja sellega käsikäes käiva konteinerarhitektuuri põhimõtete otstarbekust avaliku sektori seisukohast. Riigiasutustelt eeldatakse sügavamat planeeritust ning pikemaajalise perspektiivi arvesse võtmist kui erasektorilt seega nõuab sedavõrd fundamentaalsete muutuste sisse viimine eelanalüüsi ning põhjalikku argumenteerimist. Töös käsitletav probleem on eelmainitud uue paradigma laialdase leviku tõttu varem või hiljem päevakorras igas IT-teenuseid arendavas või haldavas riigiasutuses.

Töö teoreetilises osas käsitletakse nii monoliitse kui ka mikroteenuste arhitektuuri tugevusi ja nõrkusi, hinnatakse uue lähenemisviisi otstarbekust avaliku sektori IT-teenuste kontekstis lähtudes seni loodud teenuste toimimisest ja muu maailma kogemusest ning püütakse leida lahendusi mikroteenuste arhitektuuri juurutamisel tekkida võivatele murekohtadele.

Töö praktilises osas käsitletakse e-kiirabi teenuse rajamist mikroteenuste ja konteinerarhitektuuri põhimõtetele TEHIKu vaatest. Töö autor osaleb nimetatud teenuse rajamisel süsteemiadministraatorina.

1 Probleemi kirjeldus

Seoses uute tehnoloogiliste lahenduste kiireloomulise levikuga on tarvis infosüsteemide loomisel arvestada paratamatusega, et täna kasutatavad nii tarkvaralised kui ka riistvaralised lahendused ei ole püsivad ning loodavat infosüsteemi peab saama minimaalsete kulutustega sobitada kas uute tehnoloogiate või tulevaste teiste infosüsteemidega. Erinevate infosüsteemide lõimumisele on üles ehitatud ka meie e-riigi alustala, X-tee.

Tehnoloogilise arengu kiirenemine ei ole iseenesest mitte midagi uut, ent nüüdseks näib olevat ületatud teatud lävepunkt, mille tõttu on tekkinud surve lühendada tarkvaraliste lahenduste elutsüklit allapoole senise tarkvaraarenduse paradigma taluvuspiiri. See tähendab, et seniste mahukate monoliitsete infosüsteemide loomine võib olla aegunud nähe ning taustal toimuvad tehnoloogilised arengud sunnivad selliseid infosüsteeme muutuma kiiremini kui nende aluseks olev arhitektuur seda mõistlikult võimaldab.

Arvestades, et tehnoloogilist arengut juhivad need, kes sellesse suurimaid investeeringuid teevad ehk tehnoloogia gigandid nagu näiteks Amazon, Netflix, Uber, ja teised, on selge, et eelmainitute poolt seatud arengusuund jõuab varem või hiljem ka ülejäänud infotehnoloogia sektoritesse [2]. Nähes viimasel aastakümnel sellises kaliibris ettevõtete liikumist agiilsemat arendust võimaldavale arhitektuurile (mikroteenuste arhitektuurile) võib eeldada, et tõuseb ka üleüldine uute tehnoloogiliste lahenduste kasutusele võtu kiirus, mis mõjutab pikemas perspektiivis ka väiksemaid infotehnoloogia valdkonna organisatsioone.

Ehkki TEHIK on juba loomas uusi infosüsteeme mikroteenuste arhitektuurile on mikroteenuste kasutamise mõistlikkust avaliku sektori kontekstis vähe uuritud. Viimasest lähtuvalt tekibki vajadus analüüsida uudset lähenemisviisi ning sobivuse korral kaardistada potentsiaalseid murekohti lihtsustamaks sarnaseid ülemineku protsesse teistes, eelkõige avalikes asutustes.

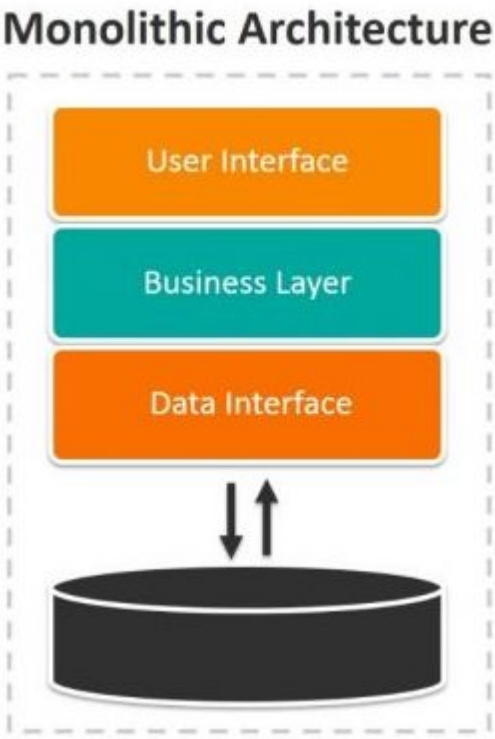
2 Komponendipõhine lähenemine

Tulemaks toime infosüsteemide haldamisega aina kiiremini muutuvas tehnoloogilises keskkonnas on mitmed tehnoloogiaettevõtted otsustanud asendada senise monoliitse tarkvaraarenduse meetodi komponendipõhise ehk mikroteenuste arhitektuuri vastu.

Järgnevalt kõrvutatakse klassikalist monoliitset ja moodsamat mikroteenuste arhitektuuri selgitamaks viimase olemust võrreldes nende positiivseid ja negatiivseid aspekte nii arenduse kui ka hilisema halduse perspektiivis.

2.1 Monoliitne arhitektuur

Monoliitset arhitektuuri peetakse traditsiooniliseks tarkvaraarenduse arhitektuuriks. Sellisele arhitektuurile ehitatud programm või teenus ei oma teineteisest selgelt eristatavaid ja eraldi käideldavaid komponente. Monoliitse teenuse võib enamikel juhtudel jagada mõttelisteks osadeks ent omaette ükski neist osadest määravat praktilist väärtust ei oma [3]. Ehkki on võimalik luua mõttelisi eraldusi teenuse eri komponentide vahel ei ole need praktikas eraldiseisvalt kasutatavad nende omavahelise tiheda integratsiooni tõttu. Monoliitsele arhitektuurile rajatud näitliku teenuse ülesehitust kirjeldab järgnev joonis (Joonis 1).



Joonis 1. Monoliidi näitlik ülesehitus [4].

Monoliitse arhitektuuri esmane arendus on sirgjoonelisem ning lihtsamini hoomatav. Tihe integratsioon eri osade vahel tingib olukorra, kus ei ole tarvis ehitada igale komponendile standardset liidest teiste komponentidega suhtlemiseks ning see säästab esmases arendusjärgus aega ning seostub seetõttu väiksemate kuludega. Samuti on ühes tükis toodetud teenust esmase arenduse järel hõlpsam testida ning lihtsustatud on ka monitooringu teostamine, sest eelmainitud tegevusi ei ole tarvis korrata iga üksiku komponendi jaoks, vaid seda saab teha terve teenuse lõikes [3].

Ühe tervikliku teenuse arendamise eeliseks on ka lihtsustatud turvalisuse tagamine, sest teenuse komponente käitatakse ühtse tervikuna samas serveris. Ühes keskkonnas koos töötamine võimaldab koondada teenuse eri osade omavahelist suhtlust ühe serveri piiridesse, mis omakorda vähendab kontaktpinda mille kaudu oleks võimalik toime panna potentsiaalseid rünnakuid ja halvata teenuse tööd [3].

Monoliitse IT-teenuse negatiivsed aspektid ilmnevad eelkõige hilisema halduse ja edasiste arendustööde käigus. Iga muudatuse tegemine või uue funktsionaalsuse lisamine kätkeb endas uue programmikoodi sobitamist kogu eelneva koodibaasiga ning testimist

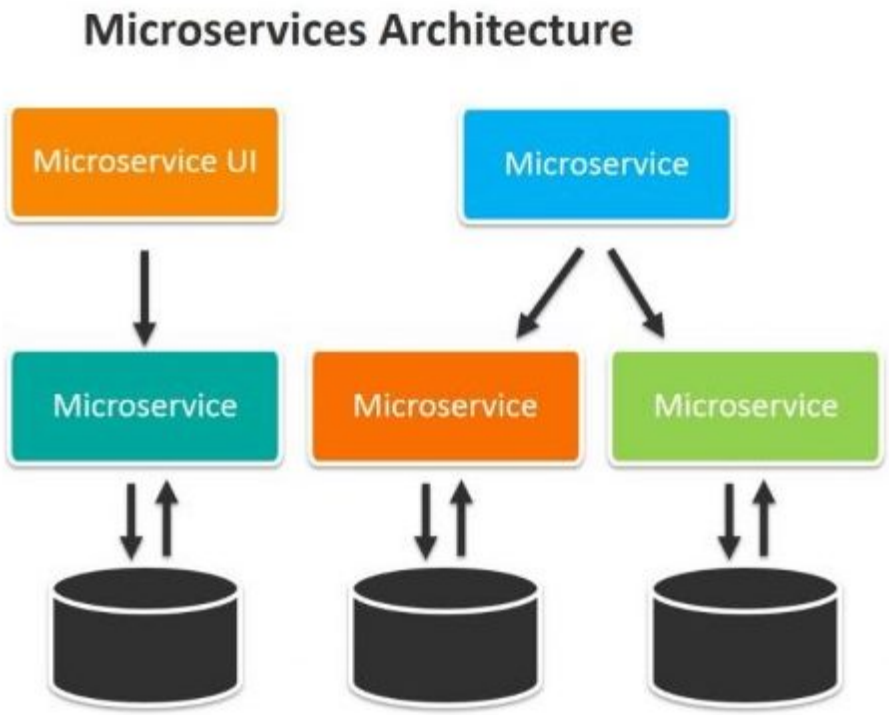
terve teenuse lõikes. Aegamisi suurenev koodibaas on aga raskesti hoomatav ja hallatav, tõrgete tuvastamine töömahukam ning hiljem on uusi arendajaid keerulisem projekti kaasata, sest enne kaasa löömist tuleb neil end kurssi viia kogu teenuse siseeluga [3]. Keerukas koodibaas võib tekitada ka olukorra, kus pärast esmast - mikroteenuste arhitektuuriga võrreldes pisut soodsamat - arendust on keeruline leida koostööpartnerit, kes oleks nõus teostama hilisemaid arendustöid mõistliku hinna eest. Soodsama esialgse arendustöö tõttu võib tellija sundida end hiljem suuremaid kulutusi tegema.

Esialgse planeerimise ja arenduse käigus valitud programmeerimiskeel ja raamistik seavad piire edasiste töövahendite valikul ning seda just monoliidi terviklikkuse tõttu. See omakorda raskendab olemasoleva teenuse sobitamist uute tehnoloogiatega ning teiste teenustega. Koormuse kasvades on monoliitse teenuse skaleerimine võrdlemisi kulukas, sest puudub võimalus skaleerida üksikuid suurima koormuse all olevaid komponente. Selle asemel tuleb dubleerida kogu rakendus mõnesse teise serverisse ning jagada koormust eri serverite vahel koormusjaoturi abil [3].

Monoliitse lähenemise puhul võib kannatada ka teenuse käideldavus, sest tõrge mistahes komponendis mõjutab kogu teenuse tööd ning üksikuid osi ei ole võimalik eraldi muuta, lisada ega taaskäivitada. Seetõttu võib monoliitne arhitektuur osutuda kõrget käideldavust nõudvates valdkondades ebatõhusaks.

2.2 Mikroteenuste arhitektuur

Kui monoliitne teenus toimib ühe jagamatu tervikuna, siis mikroteenuste arhitektuuri põhimõtteid järgides jagatakse terviklik teenus pisemateks komponentideks ehk mikroteenusteks. Igal komponendil on oma tööloogika, vajadusel oma andmebaas ning eelnevalt paika pandud suhtluskanal teiste komponentidega liidestumiseks kellelt andmeid saadakse või kellele neid hiljem edastatakse. Üks mikroteenus täidab üht spetsiifilist rolli ning on ka praktikas eraldiseisva üksusena käsitletav. Järgnev joonis (Joonis 2) kirjeldab mikroteenuste arhitektuurile rajatud näitliku teenuse ülesehitust [3], [5].



Joonis 2. Mikroteenustele rajatud teenuse näitlik ülesehitus [4].

Mikroteenustel põhineva IT-teenuse esialgne arendus on enamasti kulukam, sest lisaks kõigi komponentide tööloogikale tuleb paika panna ka nende omavaheline liidestus, üles seada rohkem andmebaase ning testimine ja monitooring peab hõlmama igat üksikut komponenti. Standardne liidestus komponentide vahel võimaldab soovi korral kasutada erinevate mikroteenuste arendamisel erinevaid programmeerimiskeeli ja raamistikke, ent see omakorda lisab keerukust ning nõuab arendustiimilt laiemat teadmiste ja oskuste pagasit [3], [5].

Mikroteenuste arhitektuurile ehitatud teenuse eelised monoliitse teenuse ees avalduvad eelkõige selle hilisema halduse ja edasiarenduste käigus. Modulaarne ülesehitus võimaldab uuendada teenuse üksikuid komponente ilma ülejäänud teenuse tööd mõjutamata. Niisamuti on lihtsustatud ka uue funktsionaalsuse hilisem lisamine või pärast esmast arendust teiste teenustega liidestamine, sest olemasoleva teenuse komponendid ei ole omavahel nii tihedalt seotud kui monoliitsel teenusel ning liidestuse jaoks on võimalik luua olemasolevasse süsteemi uus vastav mikroteenus. Mikroteenustel põhineva teenuse käideldavust monoliitse ees parandab asjaolu, et potentsiaalsete tõrgete

korral on häiritud ainult selle komponendi töö milles antud tõrge ilmnes ning see ei pruugi mõjutada ülejäänud teenuse tööd [3], [5].

IT-teenuse iseseisvateks mikroteenusteks jaotamine toob endaga paratamatult kaasa suurema võrgukoormuse, sest teenuse komponentide omavaheline suhtlus käib üle interneti võrgu [6]. See omakorda suurendab kontaktpinda potentsiaalsete rünnete jaoks ning seetõttu tuleb mikroteenustel põhineva IT-teenuse turvamisel arvestada rohkemate ründevektoritega.

2.3 Konteinerid

Mikroteenuseid käitatakse konteinerites. Konteiner on standardne pakend, mis sisaldab kõike vajalikku ühe mikroteenuse käivitamiseks - näiteks programmikoodi, teeke, sõltuvusi, ja muid abivahendeid, millele vastav mikroteenus toetub. Standardsete konteinerite kasutamine võimaldab automatiseerida mikroteenuste käivitamist ja jooksutamist sõltumata aluseks oleva serveri operatsioonisüsteemist. Niisamuti on tänu konteineritele võimalik nõudluse kasvades mikroteenuste skaleerimist automatiseerida [5].

Kui IT-teenuse üks komponent satub oodatust suurema koormuse alla, saab konteinereid haldav programm käivitada paralleelselt olemasolevaga teise identse mikroteenuse vastava konteineri abil ning jaotada koormust nende vahel. Kuna koormuse kasvades dubleeritakse ainult kõrgemat koormustaluvust vajavat komponenti, mitte kogu teenust, on skaleerimine monoliitse teenusega võrreldes kuluefektiivsem [5]

3 Mikroteenuste arhitektuuri juurutamine

Uudse paradigma juurutamine toob paratamatult endaga kaasa teatavaid murekohti ning sedavõrd põhimõttelise muutuse läbiviimine võib takerduda liigse bürokraatia, konservatiivselt meelestatud töötajate, või teatud juhtudel ka aegunud seadusandluse taha [7].

3.1 Kogemusi mujalt maailmast

Ehkki mikroteenuste arhitektuur kätkeb endas selgeid ja märkimisväärseid eeliseid võib selle kasutusele võtmine põhjustada ootamatuid probleeme või nõuda kaasnevaid muutusi, et uudsest lähenemisest suurimat kasu lõigata. Järgnevalt lahatakse tüüpilist ülemineku protsessi algsetest ajenditest kuni juurutamise protsessi lõpuni.

3.1.1 Motivatsioon

Motiveeritus liikumaks monoliitselt arhitektuurilt mikroteenustele toetub erinevates organisatsioonides alatihti just järgmistele aspektidele: monoliidi suur ja raskesti hoomatav koodibaas põhjustab ootamatuid katkestusi teenuse töös ning uute versioonide tarnimine on aeganõudev ja riskantne just teenuse käideldavuse vaatest, monoliitse teenuse tihe integratsioon sunnib arendajaid ja haldajaid kasutama iganenud tehnoloogilisi lahendusi ning integratsioon teiste tulevaste või olemasolevate infosüsteemidega on raskendatud [7].

Nimetatud kitsaskohad muutuvad aja möödudes aina teravamaks, sest käesolev tehnoloogiline arengujärk liigub üha kaugemale olemasolevast monoliitsest infosüsteemist. Drastiline vanusevahe IT-teenuse ja seda ümbritseva tehnoloogilise atmosfääri vahel tekitab üha enam tõrkeid teenuse töös ning ületades teatud lävendi võib muutuda kõnealune IT-teenus kasutamatuks [7]. Levinud on muster, kus just sellise olukorra eel või ajal hakataksegi otsima alternatiive ning jõutakse mikroteenuste paradigmani.

3.1.2 Takistused

Üleminek mikroteenuste arhitektuurile võib avaliku sektori organisatsioonides takerduda teatud juhtudel ka iganenud seadusandluse taha. Markantsem näide aegunud teemakohasest seadusandlusest pärineb Brasiiliast, kus mikroteenuste arhitektuuri kasutusele võtmist avaliku sektori asutustes takistasid riiklikele infosüsteemidele seatud arhitektuurilised nõuded. Jäik reeglistik määras ära lubatud tarkvaralised lahendused nende versiooni täpsusega ehk keelatud oli ka muidu lubatud raamistike ja abiteenuste uuemate versioonide kasutamine [7].

Avalikes asutustes võib uuenduste läbi viimise takistuseks saada ka töötajaskonna vähene motiveeritus uuendustega kaasa minna. Riigisektori töökohad on valdavalt püsivamad ning erasektoriga võrreldes väiksem töökeskkonnast lähtuv surve muutustega kaasa minna võib tekitada olukorra, kus avaliku sektori töötaja ei ole sedavõrd aldis uute tehnoloogiatega kohanema [7].

3.1.3 Kaasnevad muutused

Mikroteenuste arhitektuur toetab oma olemuselt kiireid ja tiheda graafiku alusel tehtavaid tarneid ehk jooksvad muudatused teenuse tööloogikas on evolutsioonilised ning neid viiakse läbi tihemini kui monoliitse teenuse puhul. Sedasorti agiilne arendusmeetod koostöös teenuse modulaarse ülesehitusega võimaldab luua kõrgendatud autonoomsusega arendus- ja haldustiime, kellest igaüks vastutab teatud komponendi või komponentide käekäigu eest. Standardiseeritud liidestus eri komponentide vahel võimaldab eraldiseisvatel arendustiimidel töötada erineva kiirusega ning ka töövahendite (programmeerimiskeel, raamistik, abiteenused) valikul on igal meeskonnal vabamad käed, viimast just mikroteenuste vähese omavahelise seotuse tõttu [7].

Arendusmeeskondade jagunemine väiksemateks rühmadeks, suurenenud autonoomsus tehnoloogiate valikul ning selge vastutus- ja omanikutunne ühe või mitme komponendi suhtes hägustab piire teenuse arendajate ja teenuse haldajate vahel. See loob võimaluse juurutada paralleelselt mikroteenuste arhitektuuriga ka DevOps mentaliteeti, mis joondab teenuse arendajaid ja administraatoreid ühtseks meeskonnaks [7].

On täheldatud, et mikroteenuste arhitektuuri kasutusele võtust tingitud struktuurimuutused muudavad ka asutuse sisest töökultuuri. Suurenenud

autonoomsusega meeskonnad põhjustavad olukorra, kus on tarvis toetuda asutusesisesele professionaalsele usaldusele. See tähendab, et mitmete teineteisest vähe sõltuvate meeskondade tehtava töö samaaegne jälgimine on keeruline, sest erinevaid komponente ning nende eest vastutavaid meeskondi on mitmeid ja nende töö ei pruugi toimuda (ja ei peakski toimuma) sünkroonselt. Rangema struktuuriga asutustes võib selline muutus osutuda problemaatiliseks [8].

3.2 Mikroteenuste arhitektuuri juurutamine arendusettevõtte vaatest

Arendusettevõtte seisukohast vaadatuna on käesoleval arutelul pisut teistsugune struktuur. Avaliku sektori asutustele mitmeid arendustöid teinud AS Nortal on hanketingimuste tõttu olnud sunnitud rajama uusi süsteeme mikroteenuste arhitektuurile. Nende sisemine analüüs näitas vahetult mikroteenustele ülemineku järgsel ajal 18%-list arendustempo langust. Saanud aega uudse lähenemisega harjumiseks ning vastava kompetentsi kogumiseks on suudetud arendustempot taas tõsta, ent siiski mitte ülemineku eelsele tasemele jäädes viimasele kümnendiku jagu alla [9].

Just arendustempo languse ja töö keerukuse kasvu tõttu soovib arendusettevõte hanke koostajatelt sügavamat analüüsi otsustamaks kas üht või teist süsteemi siiski tasub luua mikroteenuste arhitektuurile [9]. Komponent, mis Nortali analüüsist kahjuks puudub on, hilisema halduse, edasiarenduste ja tulevaste süsteemidega liidestuse küsimus. Suutlikkus luua sarnane funktsionaalsus alternatiivseid meetodeid kasutades lühema aja ning seeläbi ka väiksemate kuludega on arendustöö tellija (näiteks TEHIKu) seisukohast vaadatuna ainult osa tervikpildist. Tellija peab arvestama ka tellitava süsteemi hilisemate edasiarenduste ja halduskuludega, mis monoliitse teenuse korral on tõenäoliselt suuremad. Käesoleva töö autor leiab, et mikroteenuste otstarbekuse Nortali poolsel analüüsil on praktiline väärtus vaid Nortali siseste hinnangute andmisel ning sealsetest andmetest lähtuvalt ei tohiks teha laiemaid järeldusi mikroteenuste kasumlikkuse kohta. Arendusettevõte esindab selles arutelus vaid üht osapoolt, kes loodava teenuse edaspidise käekäigu eest otseselt ei vastuta

4 Mikroteenused TEHIKus

Alates 2020. aasta aprillist on avalikul kasutusel Tööelu Infosüsteemi (edaspidi: TEIS) esimesed komponendid. Tegemist on TEHIKu esimese mikroteenuste arhitektuurile rajatud infosüsteemiga ning selle eri komponentide arendus kestab tänaseni [10]. Luues uut infosüsteemi algusest lõpuni mikroteenuste arhitektuuri järgides, on võimalik mugavalt ja vähese lisatööga avada osa komponente avalikuks kasutamiseks varem ning jõudumööda sinna uut funktsionaalsust juurde lisada. Selline lähenemine võimaldab jaotada arenduskulusid pikema perioodi peale ning üleminek vanalt monoliitselt süsteemilt uuele viiakse läbi sammhaaval.

4.1 Mikroteenuste käitamine TEHIKus

IT-teenus - näiteks TEIS - koosneb mikroteenustest, mis on pakendatud konteineritesse. TEHIKus kasutatakse Docker-tüüpi konteinereid ning neid jooksutatakse Kubernetese klastrites. Kubernetese klaster on konteinereid jooksutavate serverite (TEHIKu puhul virtuaalserverite) kogum, millede vahel jaotatakse konteinerite käitamise ülesandeid. Koormuse kasvades on võimalik suurendada Kuberenetese klastri käsutuses olevate virtuaalserverite võimekust või neid juurde lisada, lubades seeläbi rohkemate konteinerite käimas hoidmist.

Haldamaks mitmeid Kubernetese klastreid, on kasutusel klastrite orkestreerimise töövahend Rancher. Rancher pakub mugavat ja tsentraliseeritud keskkonda Kubernetese klastrite halduseks ning sisaldab endas integreeritud seirefunktsioone hõlbustamaks mikroteenustele omast keerukamat monitooringu protsessi.

4.2 Senine kogemus

Olles tegelenud TEISi halduse ja paralleelselt ka edasiarendusega nüüdseks juba pisut üle aasta, võib TEHIK senisele kogemusele toetudes teha esimesi järeldusi mikroteenuste arhitektuuri otstarbekuse ja efektiivsuse kohta. Märkimisväärsematest tähelepanekutest

loetletakse jooksvate tarnete (olgu nendeks siis parandused või edasiarendused) läbi viimise lihtsust ja kiirust, kõrgendatud käideldavust ning vähenenud sõltuvust teenust käitavast riistvarast.

Teenuse uute versioonide käivitamine toimub tänu modulaarsele ülesehitusele ning konteinerite abile ilma käideldavust kahjustamata. See tähendab, et uuendusi võib läbi viia kasvõi tööpäeva sees ning vana versiooni töö ei lõppe täielikult enne uue versiooni edukat käivitamist. Kui uue versiooni käivitamisel peaks esinema tõrkeid, on võimalus peatada tarne, suunata uus versioon tagasi testimisse ning jätkata vana versiooni kasutamist kuniks uues versioonis on tõrked kõrvaldatud. Tööpäeva sees tarne läbi viimine oli seniste monoliitsete teenuste puhul mõeldamatu ning neid teostati nädalavahetustel just käideldavuse kahjustumise ohu tõttu.

Ühe tarne raames ei ole vajadust lülitada ümber tervet infosüsteemi, sest on võimalik uuendada üht komponenti korraga. Sellest tulenevalt on ühe tarne maht väiksem ning ka potentsiaali tõrgete tekkimiseks vähem. Vähenenud tarne maht teeb uuenduste läbiviimise kiiremaks.

Käesoleva alapeatüki alguses nimetatud vähenenud sõltuvus teenust käitavast riistvarast andis suurima efekti ajal, mil riigipilve haldav Majandus- ja Kommunikatsiooni-ministeerium soovis riigipilve aluseks olevates serverites läbi viia operatsioonisüsteemi uuendust. TEISi mikroteenustele rajatus võimaldas seda ühe komponendi haaval ajutiselt ümber lülitada TEHIKu enda serveritesse riigipilve alusmasinate uuendamise ajaks. Riikliku mastaabiga infosüsteemi sedasorti ajutine ümberpaigutus ilma käideldavust kahjustamata on omamoodi erakordne ning see sai toimuda suuresti tänu mikroteenuste arhitektuurile.

4.3 Struktuurimuutused

Lõikamaks suurimat kasu mikroteenuste arhitektuurist kui uuest tarkvaraarenduse paradigmast on soovitatav - nagu näitab ka teiste kogemus mujalt maailmast - peegeldada neid muutusi ka asutusesiseses struktuuris. Nimelt soosib mikroteenuste arhitektuuri kasutusele võtmine paralleelselt ka DevOps mõtteviisi integreerimist. Sõltuvalt

käsitletava organisatsiooni eesmärkidest võidakse jõuda mikroteenuste paradigmani just DevOps mentaliteedi rakendamise kaudu - või ka vastupidi.

TEHIKus on käimas arhitektuuriliste muutustega paralleelselt ka DevOps mentaliteedi osaline üle võtmine. Ehkki ei rutata arendajaid ja administraatoreid ühte meeskonda paigutama, on teatud mõjutused siiski märgata. DevOps innustab looma ühtlast töövoogu tihedate ja kiirete tarnete näol ning selle eesmärgi täitmiseks suunatakse administraatoriteid looma automatiseeritud tarneahelat, mis võtaks koodibaasis toimunud muudatuse ning viiks selle ellu soovitud keskkonnas - test- või toodangu- keskkonnas. Seeläbi suureneb administraatorite kokkupuude arenduses toimuvaga ning see omakorda aitab neil paremini tajuda tervet teenuse elutsüklit kontseptsioonist arenduse ja toodangukeskkonnani välja.

TEHIKus on asutuse struktuuri tõttu DevOps mentaliteedi täielik üle võtmine raskendatud, sest mahukate projektide raames tehtav arendustöö ostetakse valdavalt sisse välistelt arendusettevõtetelt ning arendajad ja administraatorid ei ole osa samast organisatsioonist. Sõltuvalt arendusettevõtte töökorraldusest võivad ühe projekti raames arendajad vahetuda ning sedavõrd dünaamilises situatsioonis oleks arendajate ja administraatorite ühtseks meeskonnaks liitmine ebamõistlikult keeruline ettevõtmine.

4.4 E-kiirabi

TEISi projekti senine edukus ning ülemaailmne trend mikroteenuste ja konteinerarhitektuuri kasutamise suunas loovad soodsa pinnase kasutamaks mikroteenuste arhitektuuri ja konteinereid ka tulevaste infosüsteemide arendamisel. Üheks selliseks infosüsteemiks on e-kiirabi, mille arendus on parasjagu algusjärgus. Järgnevas protseduuride kirjeldustes on lähtutud süsteemiadministraatori seisukohast.

Sarnaselt TEISile algab ka e-kiirabi projekti praktiline osa süsteemiadministraatori vaates GitLabi vastavate projektide loomisega. GitLab on versioonihalduskeskkond, kuhu laetakse üles arendajate poolt koostatud programmikood ning mis võimaldab jälgida erinevate arendajate poolt üles laetavat koodi ja selle muudatusi. Samuti on võimalik vajadusel ennistada tehtud muudatusi taastamaks eelnenud seisu - näiteks mittetöötava programmikoodi üles laadimise korral. Versioonihalduskeskkonna kasutamine

arendustegevuses on levinud ka monoliitsete teenuste arendusprotsessis ehk see etapp võib olla osa nii monoliitse kui ka mikroteenustele rajatud teenuse arendusest.

Nagu eelnevalt kirjeldatud käitatakse TEHIKus mikroteenuseid Rancheriga hallatavates Kubernetese klastrites. Niisamuti peab e-kiirabi projekti raames kirjutatud programmikood saama pakendatud Dockeri konteineritesse ning konteinerid on tarvis anda edasi Rancherile käivitamiseks.

Kogu koodibaasi (GitLabi projekti) ja konteinereid käitavate klastrite halduse (Rancheri) vahele jäävat tarneahelat automatiseeritakse kasutades GitLabi sisseehitatud CI/CD tarneahelat. Viimane kujutab endast skriptide kogumikku, mis käivitub antud juhul iga GitLabis oleva koodibaasi täiustuse korral, kui üles laadimise juurde lisatakse vastav silt. GitLab CI/CD tarneahel alustab tegevust, nähes eelnevalt kokku lepitud silti programmikoodi üles laadimise juures.

Kuna teenust soovitakse käitada (sarnaselt TEISile) Dockeri konteineritena, mida orkestreerib Rancher, on tarvis programmikood pakendada automatiseeritud moel samuti Dockeri konteineriteks. Selle tarvis luuakse JFrog Artifactoy's repositoorium, kus hoiustatakse üldist Dockeri konteineri malli ning vajalikke sõltuvusi ja teeke, mida üks käesoleva projekti konteiner vajada võiks. GitLab CI/CD tarneahela ülesandeks on eelmainitud repositooriumist võetud konteineri malli, sõltuvuste ja teekide põhjal luua GitLab koodibaasis kirjeldatud funktsionaalsusega täisväärtuslik Dockeri konteiner ning panna see Rancheri kaudu Kubernetese klastrisse tööle. Lubamaks GitLabi automatiseeritud tarneahelal käivitada Rancheris konteinereid, on tarvis luua Rancherisse tarnete läbiviimise konto ehk kasutajakonto, millel on Rancheris sobivad ligipääsud ning mille sisselogimisandmed on antud GitLabile kasutamiseks.

Arendustöö toimub kirjeldatud seades kuni e-kiirabi esimese toodanguversiooni valmimiseni. Seejärel paigutatakse valminud toodanguversioon Kubernetese toodanguklastrisse (klastrisse, milles käitatakse toodangus olevaid ehk avalikuks kasutamiseks valmis teenuseid).

TEISi rajamine mikroteenuste arhitektuurile toob tulu ka väljaspool TEISi projekti. Nimelt on võimalik mõningaid TEISi komponente või TEISi arendades loodud

abivahendeid tarvitada e-kiirabi teenuse arendusel. Osaliselt on kasutatavad ka automatiseeritud tarneahela skriptid, mis said juba TEISi arendades valmis kirjutatud ning mida pidi kohati vähesel määral muutma, et neid kasutada ka e-kiirabi teenuse arendusel.

E-kiirabi projekti tähtajad on arendaja poolsetel põhjustel edasi liikunud ning seetõttu ei ole käesoleval hetkel võimalik kirjeldada projekti järgmisi arengujärke. Praegune e-kiirabi projekti situatsioon sobib suurepäraselt selgitamaks mõningaid takistusi, mis võivad mikroteenuste arhitektuuri kasutusele võtmisel realiseeruda. Nimelt on e-kiirabi teenus seda arendava arendusettevõtte jaoks esimene mikroteenuste arhitektuurile rajatav teenus ning uue arhitektuurilise mentaliteediga harjumine võtab paratamatult aega. Seepärast on ka e-kiirabi arendusega seonduv ajagraafik nihkunud, lubamaks arendajatel kohaneda ning arendusettevõttel kompetentsi koguda.

Hoolimata esialgse arenduse keerukusest leiab käesoleva töö autor, et hilisema halduskoormuse ja muude käesolevas töös eelnevalt mainitud positiivsete aspektide valguses on siiski kasulik vajaminev õppeperiood üle elada. Seni on mikroteenuste arhitektuuri nõue olnud osa vaid mõningatest riigihangetest ehk arendusettevõttes vastava kogemuse puudumine on mõistetav. Aja möödudes kasvab ka arendusettevõtete mikroteenuste arhitektuuri alane kompetents ja kogemuste pagas, mistõttu võib eeldada, et tulevikus on sedasorti probleeme vähem või puuduvad need sootuks.

4.5 Riiklike infosüsteemide käitamine tulevikus

Kuna mikroteenused võimaldavad oma olemuselt automatiseeritud skaleerimist vastavuses koormuse kasvu või langusega, on avalike asutuste infosüsteeme käitavad serveripargid ehitatud teatud võimekuse varuga. Ootamatu koormuse kasv mõnele infosüsteemi osale põhjustab selle sama osa automaatse skaleerimise seni ootel oleva riistvara arvelt, et tulla toime kasvanud koormuse all. Sellise võimsuse varu hoidmine on ühelt poolt lisakulu ent teistpidi hädavajalik tagamaks teenuse käideldavust ka kõrgendatud koormuse ajal.

Tõenäosus, et mitu või kõik avalikud infosüsteemid satuvad samaaegselt kõrgendatud koormuse alla, on väike, mistõttu oleks otstarbekas käitada riiklikke infosüsteeme

tsentraliseeritud serveriparkides (mitmes erinevas asukohas kaitsmaks teenuseid füüsiliste ohtude eest), kus varuks olev riistvaraline võimsus oleks jagatud mitme infosüsteemi valdusesse. Samale mentaliteedile toetub ka Majandus- ja Kommunikatsiooniministeeriumi loodud riigipilv, milles käitatakse mõningaid TEHIKu hallatavaid teenuseid.

2021. aasta aprillikuus Majandus- ja Kommunikatsiooniministeeriumi ning Microsofti vahel sõlmitud ühiste kavatsuste leppe üheks kandvaks teemaks on avalikud pilveteenused ning nende turvaline kasutusele võtmine [11]. Sellest lähtuvalt võib eeldada, et pilveteenused on tulevikus avaliku sektori asutuste igapäevaste töövahendite hulgas ning olemasolev riistvaraline võimekus koondatakse asutuste üleselt ühtsesse riiklikku pilve ja see hakkab baseeruma Microsofti poolt välja töötatud lahendustel.

Pilveteenuste kasutuse laienemine avalikus sektoris loob veelgi soodsamad tingimused mikroteenuste arhitektuuri rakendamiseks. Eelnevalt mainitud automatiseeritud skaleerimise võimekuse monoliitsele infosüsteemile lisamine on viimase ülesehituse tõttu raskendatud, mistõttu annaks asutuste ülene pilveteenus koostöös mikroteenuste arhitektuurile rajatud infosüsteemidega märkimisväärse efektiivsuse eelise seni toimivate vanamoodsate süsteemide ees.

4.6 Järeldused

Mõningatest murekohtadest hoolimata on senine mikroteenuste arhitektuuri kasutusele võtmise protsess TEHIKus olnud edukas. Ehkki soov minna üle moodsamale arhitektuurile on põhjustanud teatavaid põrkumisi arendust teostavate ettevõtetega leiab käesoleva töö autor, et TEHIK on liikumas õiges suunas ning sellise protsessi läbi viimine lihtsustab tulevikus teiste avaliku sektori asutuste sarnaseid protsesse. Viimast seetõttu, et meie võrdlemisi väikeses siseriiklikus majandusruumis on juba mitmed suuremad arendusettevõtted saanud TEHIKu või mõne muu tellija eestvedamisel esimesed kogemused mikroteenuste arhitektuuri arenduses ning hiljem on teistele asutustele sarnase töö teostamine seetõttu lihtsam.

Hilisemate projektide loomist mikroteenuste arhitektuurile lihtsustab ka asjaolu, et teatud komponendid võivad olla eelnevatest projektidest väheste muudatustega üle võetavad, nagu ka TEISi ja e-kiirabi puhul. Autor arvab, et tulevikus võiks suurendada ka erinevate

riigiasutuste omavahelist koostööd just loodava programmikoodi taaskasutamise eesmärgil, et vältida juba tehtud töö uuesti tegemist - mikroteenuste arhitektuuri modulaarne iseloom toetab igati juba kirjutatud koodi taaskasutamist.

Ehkki käesolevas peatükis kirjeldatud mikroteenuste arhitektuuri kasutusele võtmine on toimunud avaliku sektori asutuses, ei ole senimaani realiseerunud mõningad avaliku sektoriga seonduvad probleemid, mida on täheldatud sarnaste protsesside juures mujal maailmas. Olukord, kus riigiasutus veab erasektori partnerite arengut moodsamate töömeetodite suunas, on infotehnoloogia valdkonnas tavapäratu. Käesolevas lõputöös eelnevalt kirjeldatud näide Brasiilia jäigast infosüsteemide arhitektuurilisi omadusi sätestavast seadusandlusest on Eesti kontekstis vaadelduna arusaamatu ning näib autorile tarbetult piirav. Arvestades mõttelaadi erinevusi Eesti riigiasutuste ja muu maailma sarnaste organisatsioonide vahel, arvab autor, et Eestis saab mikroteenuste arhitektuurile üleminek olema mõnevõrra valutum.

Eelnevalt kirjeldatud e-kiirabi projekti käekäik ilmestab selgelt uue arhitektuurilise lahenduse kasutusele võtmisel tekkida võivaid takistusi, ent monoliitse infosüsteemi loomine tänapäevases tehnoloogilises kontekstis kätkeb endas pikemas perspektiivis tõenäoliselt märkimisväärselt suuremaid kulutusi ning ei ole seetõttu avaliku sektori asutuses alternatiivina enam mõeldav.

Globaalne trend ning käesolevas töös läbi viidud analüüs lubavad arvata, et mahukate infosüsteemide üleminek mistahes vormis mikroteenuste arhitektuurile on varem või hiljem paratamatu ning selleks, et vähendada iganenud, monoliitse malli järgi tehtava töö hilisemat uuesti tegemist, tuleks autori arvates edaspidised riiklikud infosüsteemid luua just nimelt mikroteenuste arhitektuurile. Hiljutine kokkulepe Majandus- ja Kommunikatsiooniministeeriumi ning Microsofti vahel kiirendab autori arvates mikroteenuste arhitektuurile üleminekut veelgi.

5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli välja selgitada mikroteenuste arhitektuuri kui uue tarkvaraarenduse paradigma otstarbekus avaliku sektori IT-teenuste kontekstis ning sobivuse korral kaardistada potentsiaalsed murekohad selle juurutamisel.

Töö käigus analüüsiti mitut mikroteenuste arhitektuurile ülemineku protsessi mujalt maailmast, hinnati TEHIKu senist mikroteenuste rakendamise edukust ning osaleti ühe uue mikroteenuste arhitektuurile rajatava teenuse arenduse ettevalmistustöödes.

Mikroteenuste arhitektuuri juurutamise protsessi analüüsid mujalt maailmast ning senine kogemus TEHIKus annavad autori arvates kinnitust sellele, et mikroteenuste arhitektuur pakub selgeid ja märkimisväärseid eeliseid klassikalise, monoliitse arhitektuuri ees. Hoolimata modulaarsusest tingitud kõrgematest esmastest arenduskuludest ja mõnevõrra suurenenud ülesehituslikust keerukusest, on mikroteenuste arhitektuuri kasutusele võtmine avalikus sektoris autori arvates õige samm.

Töö praktilises osas, e-kiirabi arenduse ettevalmistustöödes, oli autori osaks Rancheris arenduskeskkonna üles seadmine, täiendavate virtuaalserverite loomine ja nende Kubernetese klastri käsutusse andmine (võimaldamaks e-kiirabi projektist tuleneva lisakoormusega toimetulekut), välistele arendajatele ligipääsuõiguste jagamine, Rancherisse automaatsete tarnete läbiviimise konto loomine ja GitLab-iga ühendamine ning Rancheri poolel võrgusuunamise sätestamine.

Käesoleva töö alguses püstitatud mikroteenuste arhitektuuri otstarbekuse küsimus sai töö käigus positiivse vastuse ning potentsiaalsed murekohad on kaardistatud. E-kiirabi teenuse edasine arendus jätkub arendusettevõtte jaoks uudse arhitektuurilise ülesehitusega harjumise tõttu teatud hilinemisega.

Kasutatud kirjandus

[1] “Cloud Microservices - Global Market Trajectory & Analytics,” Global Industry Analysts, Inc, 2021.

[2] K. Ismail, “7 Tech Giants Embracing Microservices,” *CMSWire*, 2018, [Võrgumaterjal]. Available: https://www.cmswire.com/information-management/7-tech-giants-embracing-microservices/. [Kasutatud: 20.04.2021].

[3] R. Gnatyk, “Microservices vs Monolith,” *N-iX*, 2018. [Võrgumaterjal]. Available: https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/. [Kasutatud: 22.04.2021].

[4] J. Johnson, L. Shiff, „What Is Microservice Architecture? Microservices Explained,“ *BMC blogs*, 2021. [Võrgumaterjal]. Available: https://www.bmc.com/blogs/microservices-architecture/.

[5] A. Amanse, “Why should you use microservices and containers?” *IBM Developer*, 2020. [Võrgumaterjal]. Available: https://developer.ibm.com/depmodels/microservices/articles/why-should-we-use-microservices-and-containers/. [Kasutatud: 18.04.2021].

[6] M. Fowler, “Microservice Trade-Offs,” 2015. [Võrgumaterjal]. Available: https://martinfowler.com/articles/microservice-trade-offs.html. [Kasutatud: 25.04.2021].

[7] W. Luz, E. Agilar, M. C. de Oliveira, C. E. R. de Melo, G. Pinto, and R. Bonifa´cio, “An Experience Report on the Adoption of Microservices in Three Brazilian Government Institutions,” *SBES2018*, 2018.

[8] S. Baškarada, V. Nguyen, A. Koronios, “Architecting Microservices: Practical Opportunities and Challenges,” *Journal of Computer Informaton Systems*, 2018.

[9] A. Torim, “Komponendipõhine tarkvaraarendusraamistik kui avaliku sektori tarkvaraplatvorm”, Tallinn, 2017.

[10] R. Soobik, “Uus infosüsteem aitab tööandjatel luua ohutut töökeskkonda,” *Tööelu*, 2020. [Võrgumaterjal]. Available: https://tooelu.ee/et/uudised/2810/uus-infosusteem-aitab-tooandjatel-luua-ohutut-tookeskkonda. [Kasutatud 15.04.2021].

[11] „Eesti valis pilvelahenduste partneriks Microsofti,“ *Postimees 28.04.2021.* [Võrgumaterjal]. Available: https://majandus24.postimees.ee/7236028/eesti-valis-pilvelahenduste-partneriks-microsofti?_ga=2.16948400.865151278.1619110193-512025969.1589317663&fbclid=IwAR20rrJzE03Ls4ggrJ9cU59Y8G_X4rI6VNQipQvZmINpOQI3LQSY6INFIq8. [Kasutatud 28.04.2021].

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

Mina, Oliver Vanamets

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Mikroteenuste ja konteinerarhitektuuri juurutamine riigiasutuses TEHIKu näitel“, mille juhendajateks on Siim Vene ja Marko Valing,
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

29.04.2021

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