TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Informaatika õppekava

Fred Hansen

Modernse monitooringu kasutuselevõtt EUIF infosüsteemide näitel

Bakalaureusetöö (9 EAP)

Juhendaja(d): Piret Luik, PhD

Tartu 2021

Modernse monitooringu kasutuselevõtt EUIF infosüsteemide näitel

Lühikokkuvõte:

Bakalaurusetöö eesmärgiks oli implementeerida paindlik monitoorimislahendus suurte tarkvarasüsteemide serverite jälgimiseks. Lahendus teostati Nortali EUIF projekti näitel. See koosnes monitoorimistööriistadest Prometheus, Micrometer ja Grafana ning on lihtsasti laiendatav teistele erinevate põlvkondade rakendustele. Monitoorimislahendus oli vajalik, kuna rakenduste jõudlusmonitooringu (APM) vahendid ei tee süsteemi piisavalt läbipaistvaks või ei ole muudmoodi sobivad.

Kirjeldatakse lahenduse implementeerimist kolme EUIF rakenduse põhjal, mille tulemuseks on süsteemihaldajatele olulise meetrika reaalajaline jälgimisvõimalus, et probleeme kiiremini tuvastada ja parandada.

Võtmesõnad: Prometheus, Grafana, Micrometer, süsteemid, monitooring

CERCS: P175 Informaatika, süsteemiteooria

Modern systems monitoring in Estonian Unemployment Insurance Fund

Abstract:

The purpose of this bachelor's thesis was to implement a flexible systems and application monitoring solution to monitor the servers of large-scale software systems. The monitoring solution was implemented based on the EUIF project of Nortal. It was composed of monitoring tools Prometheus, Micrometer and Grafana and is easily horisontally scaleable to new applications.

The thesis describes the implementation of the monitoring solution for three EUIF applications. The result is a tool that developers can use to monitor different system and application metrics in real time to identify and fix problems more efficiently.

Keywords: Prometheus, Grafana, Micrometer, systems, monitoring

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Tänapäeval on levinud, et ühe monoliitse rakenduse asemel võetakse kasutusele mitmed erinevad mikroteenused. Romana Gnatyk [1] kirjeldab, kuidas mikroteenuste iseseisvus teeb küll infosüsteemid paremini arusaadavaks ja hallatavaks, kuid tekitab ka erinevaid probleeme. Kuna mikroteenused on üksteisest eraldi seisvad rakendused, mis tihti ka jooksevad erinevates masinates, siis on keeruline neid jätkusuutlikult jälgida. Sellele probleemile pakutakse töös ühte võimalikku lahendust.

Mida suurem on infosüsteem, seda olulisemaks muutub usaldusväärse ja paindliku monitoorimislahenduse kasutusele võtmine. Ilma monitoorimissüsteemita on oht, et kui serveris või rakenduses midagi valesti läheb, siis arendajad satuvad segadusse ja probleemi parandamiseks kulub kauem aega. Hea monitoorimislahendus aitab arendajatel probleemidele kiiremini jälile saada ning teavitab ohukohtadest, mistõttu need saab lahendada enne kui need rakenduste tööd saavad hakata häirima. Kuna infosüsteemi tervist saab reaalajas jälgida, siis on tagatud ka nii arendajate, testijate kui ka kliendi meelerahu [2].

Töö eesmärk on implementeerida monitoorimislahendus mitmerakenduseliste infosüsteemide veebiserverite jälgimiseks Nortali EUIF projekti infosüsteemi jaoks, et süsteemi haldajad saaksid efektiivsemalt probleeme leida ja parandada. Lahendus peab toetama mitmeid raamistikke, et monitoorida erinevaid süsteemi- ja rakendusemuutujaid võttes kasutusele monitoorimistööriistad Micrometer, Prometheus ja Grafana.

Töö autor liitus Nortali EUIF ehk Eesti Töötukassa projektiga 2019 aasta septembris arendajana ja on sellest ajast teinud teistega koostööd, et töös mainitud Töötukassa rakendusi arendada.

Töö kontektsi paremaks mõistmiseks kirjeldatakse järgmistes peatükkides Töötukassat ja Nortali EUIF projekti üldiselt ja universaalse monitoorimislahenduse loomist selleks, et erinevate põlvkondade rakendusi monitoorida.

1 Tarkvarasüsteemide jälgimine

Töös kasutatakse mõistet monitooring tähendamaks veebirakenduste ja nende serverite jälgimist ehk seiret [3, 4]. On üldlevinud teadmine, et tootmises olevate rakenduste monitoorimata jätmine on halb praksis [5]. Kui monitoorimata jäetud rakenduses midagi ootamatut juhtub, võib lõputult aega kuluda diagnoosimisele, mis valesti läks.

Üks näide teema olulisuselt ilmnes kui autori varasema tööpraktika kogemuse käigus teises IT firmas tekkis olukord, kus kogemuse puudumise tõttu tegi autor vea ja see muutus rakendus automaatselt LIVE-keskkonnas. Seetõttu rakenduses tekkis viga, mille põhjust otsisid arendajad ligi pool tööpäeva. Teist näidet kirjeldab Sander Pärn [6] oma Bakalaureusetöös eHealthis, mis on samuti Nortali projekt, esinenud mälulekkest ja sellest pikast protsessist, kuidas see avastati. Kui mõlema näite puhul oleks olnud kasutusel hea monitooring, siis oleks tõenäoliselt probleem leitud ja lahendatud juba varem.

1.1 Suurte süsteemide haldamine

Nagu eelpool toodud näidetes ilmnes, siis mida suurem on infosüsteem, seda keerulisem on leida probleemide allikaid ja raskem on olla kindel, et süsteem on hea tervise juures. Monitoorimata süsteemis hakkavad arendajad probleeme lahendama sel hetkel, kui neile nendest teavitatakse ehk siis kui süsteemi rakendustes on ilmnenud viga ja kasutaja töö või tegevus on häiritud. Probleemide lahendamisaeg kasvab süsteemi keerukusega. Hea monitoorimislahendusega aga saavad arendajad probleemidest teada varem, väheneb probleemide lahendamiseks kuluv aeg ning seega suureneb süsteemi töökindlus [7].

Bakalaurusetöö autori hinnangul on Töötukassal üks Eesti keerulisemaid infosüsteeme, mis aastatega järjest kasvab. Varasemalt Nortali poolt kasutatud monitoorimissüsteemid on osutunud ebasobivaks süsteemiga toime tulema või lihtsalt liiga kulukaks. Süsteemi kasvu ja varasemate lahenduste ebasobivusega on kaasnenud vajadus leida uus paindlik monitoorimissüsteem, et süsteemi paremini hallata.

1.2 Infosüsteemide korrasolu indikaatorid

On mitmeid eri indikaatoreid, mis näitavad rakenduse või süsteemi korrasolu. See peatükk kirjeldab mõningaid selliseid indikaatoreid, mida on oluline süsteemi haldajatel jälgida, et kinnitada selle korrektset toimimist.

1.2.1 Rakenduste staatus

Kõige lihtsam muutuja, mida jälgida on rakenduse staatus. Rakendusele saadetakse *ping*, mille vastus võib olla 1 - rakendus on kättesaadav ehk rakendus käib, või 0 - rakendus ei vastanud ehk on seiskunud [8].

1.2.2 Avatud failide kogus, mälu kasutus ja GC pausid

Avatud failide arv ja mälu kasutus on olulised näitajad Sander Pärna [6] poolt kirjeldatud mälulekete avastamisel. Nortali EUIF testkeskkondade igal *worker* füüsilisel masinal on eraldi maksimaalne lahti olevate failide kogus, milleks on 1048576 faili. Kui see arv on mõnel sõlmel oluliselt suurem kui tavaliselt, siis võib tegu olla mälulekkega.

Suurem osa rakendustes käsitletavatest failidest on PDF formaadis, millest tulenevalt on oht, et *worker*'i mälu saab täis enne kui failide arv maksimumini jõuab. Seetõttu jälgitakse ka mälukasutust [9].

Prügi kogumine (ingl. k. *Garbage Collection* ehk GC) on automaatne mäluhaldus. Java programmeerimiskeeles toimib see automaatselt, kuid paus GC protsessis tähendab, et mälu piirkond on täis ja JVM peab ootama kuni sinna ruumi tekib. Kui paus kestab rohkem kui sekundi või ühes sekundis on mitu pausi, mille kogupikkus on suur osa ühest sekundist, siis võib tegu olla mälulekkega [9, 10].

1.2.3 Protsessori kasutus

Protsessori kasutus on üks kõige olulisematest meetrikatest, mida süsteemis tuleks jälgida. Kui protsessori kasutus erineb tavalisest tasemest, siis võib see olla mitme erineva probleemi tõttu. Kui see on kõrgem normist, võib olla süsteem nakatunud kahjurvaraga. Kui protsessori kasutuses on näha, et kõrge tase vaheldub madalama tasemega, mida saadab rakenduse aeglustumine, siis võib tegu olla protsessori ülekuumenemisega. Sel juhul tuleb füüsiliste masinate jahutust kontrollida või ressursse juurde anda. Monitooritud süsteemi on turvalisem uut tarkvara jooksma panna, kuna selle mõju protsessori kasutusele on koheselt näha [11].

1.2.4 Vigade sagedus ning HTTP veakoodide esinemine

Ideaalne vigade arv rakenduses on null, kuid reaalses olukorras pole see tihti võimalik. Siiski on vajalik üritada seda saavutada, kuna vigade esinemine on ilmselge näitaja rakenduse või süsteemi korrasolust [12].

1.2.5 Päringute reaktsiooniaeg

Päringute reaktsiooniaeg (ingl k. *Peak Response Time* ehk PRT) on üks kõige olulisematest muutujatest, mida jälgida. PRT aitab leida, millised päringud on probleemsed ja potentsiaalselt ka nende probleemide põhjuseid. Kuvatakse päringud, neile kulunud aeg ja HTTP otspunktid (ingl. k. - *endpoint*), millele päringud tehti. Enamus rakendusesiseseid päringuid ja päringuid teistele EUIF rakendustele peaks jääma alla 250ms. Kui tuleb ette, et üks päring võtab korduvalt palju aega või mitmed eri päringud võtavad järjekindlalt kauem aega, siis tuleb arendajatel leida sellele põhjus või päringuid optimeerida. Keskmine reaktsiooniaeg (ingl. k. - *Average Response Time* ehk ART) näitab süsteemi

üldist jõudlust ja selle järgi saab hinnata rakenduste kasutusmugavust. Kui ART ja PRT on mõlemad kõrged võib olla tegemist serveriprobleemiga [13].

Siinkohal tuleb arvestada, et päringule kuluvat aega mõjutab ka see, kas tegu on Nortali või Töötukassa testkeskkonnaga või LIVE keskkonnaga. EUIF rakenduste puhul, nagu paljudki teised avaliku sektori rakendused, teevad ka regulaarselt X-tee päringuid [14] ja X-tee masspäringuid, mis potentsiaalselt pärivad kümneid tuhandeid andmeid korraga, mistõttu need päringud võivad võtta isegi LIVE keskkonnas $\sim$5 min aega.

1.2.6 Andmebaasi ühendused

Andmebaasi ühendused, ehk Java koodi puhul JDBC ühendused (ingl. k. - *Java Database Connectivity*) näitavad kui palju teevad rakendused andmebaasipäringuid. Igal rakendusel on maksimum arv ühendusi, mida saab korraga teha. Kui rakenduse JDBC päringud jõuavad tihti maksimumini, siis tuleb päringuid optimeerida, näiteks *batch*'imisega [15]. Juhul kui järjekindlalt on JDBC ühendused maksimumi peal või ei lähe nulli kui rakenduse kasutus väheneb, siis on tõenäoliselt tegu ühenduslekkega (ingl. k. - *connection leak*) ehk päringut või transaktsiooni ei suleta korralikult [16].

2 Nõuete analüüs

Et paremini mõista konteksti, milles monitoorimisrakendust implementeritakse, kirjaldatakse selles peatükis Nortali EUIF rakendusi lähemalt, erinevaid tööriistu monitoorimise jaoks ja kuidas seda EUIF projektis varem tehti.

2.1 Nortali EUIF rakendused

Nortali poolsed Töötukassa rakendused on eelkõige mõeldud Töötukassa töötajate kasutuseks. Nende peamine funktsioon on automatiseerida töötajate tööd - eeltäita vorme, saata teavitusi ja meeldetuletusi ja suhelda teiste Eesti e-teenustega üle X-tee [17].

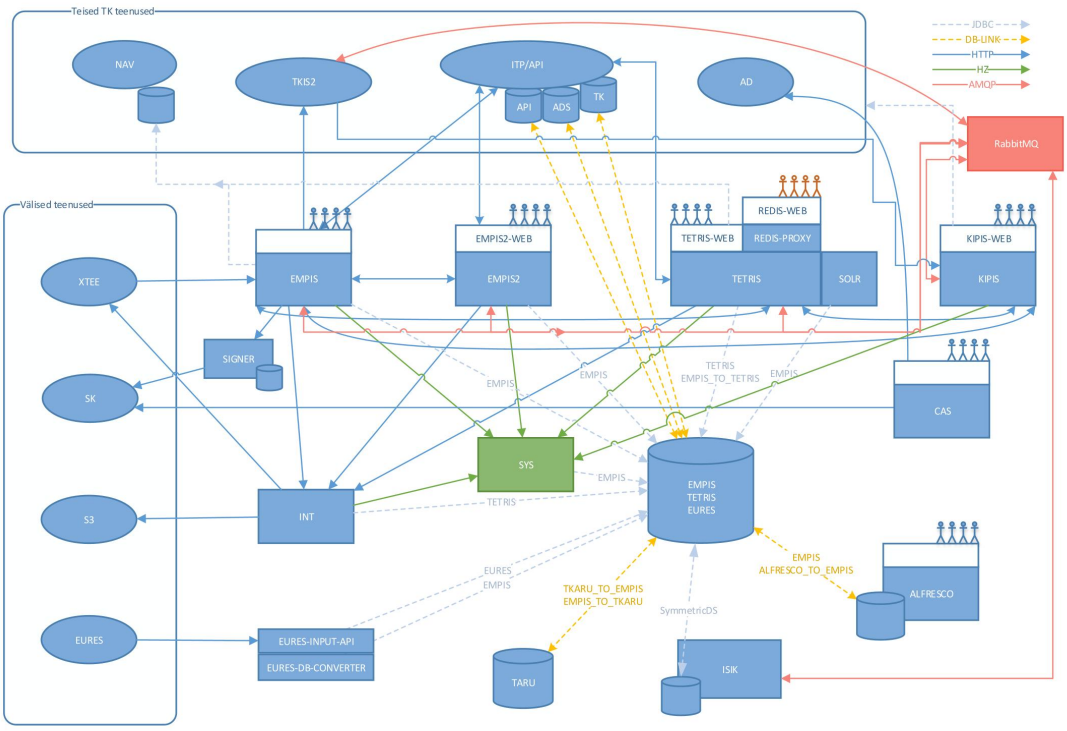
Kõik Nortali EUIF rakendused töötavad Kubernetes tehnoloogial põhinevas klastris. Nortali testkeskkondade klastril on neli sõlme (ingl. k. *node*) - üks *master* ja kolm *worker* sõlme. Rakendused käivitatakse Docker konteinerites ehk *pod*'ides. Kubernetes hindab *worker* masinate ressursse ning võib käivitada rakenduse konteineri vastavalt vajadusele mistahes *worker* sõlmes.

Analoogselt toimivad klastrid ka klienditesti (TK-test) ja tootmiskeskkonna (LIVE) jaoks, millele edaspidi viitan lühenditega. Kõik Töötukassa infosüsteemi rakendused ja nende seosed on välja toodud joonisel 1.

Kõik monitooritavad rakendused on kirjutatud programmeerimiskeeles Java ja kasutatud raamistikku Spring Boot 2.0 või hilisemat kõigil peale kahe rakenduse. Rakenduses EMPIS2-s on kasutatud Spring Boot 1.5.9 ja EMPIS-es on kasutusel Spring Framework 4.1.9 ja Aranea Framework 1.2.2. Aranea [19, 20] on Nortali enda poolt välja töötatud Java põhine teek.

Töös implementeeritakse monitoorimislahendus kolmele rakendusele - Isikuandmete Moodul (**IAM**), Kinnipidamiste Süsteem (**KIPIS**) ja Tööhõive infosüsteem 2 (**EMPIS2**, ingl. k. - *Employment Informatics System 2*). IAM ja KIPIS on kõige uuemad Töötukassa rakendused. Seega nendes on kõige lihtsam lahendust implementeerida ja nende kaudu on võimalik montitoorimisplatvormil Prometheuse süsteemi meetrika kätte saada. EMPIS2 on üks vanematest rakendustest, kuid mitte kõige vanem - lahenduse implementeerimine on selles raskem kui eelmainitud uuemates rakendustes, kuna Micrometer toetab lisateegiga Spring Booti versioone 1.5.0 või hilisemad. Micrometerit kirjeldatakse lähemalt hiljem selles peatükis ja lahendusele olulisi rakenduste iseärasusi on kirjeldatud kolmandas peatükis.

Rakenduse EMPIS monitoorimine on samuti väga oluline, kuid seal ei ole Spring Booti kasutusele võetud. EMPISes on lahenduse kasutusele võtmine seetõttu keerulisem kas Micrometeri praegusesse EMPISesse sobitamise või EMPISe Spring Bootile uuendamise tõttu ning töös sellele rakendusele lahendust ei tehta.



Joonis 1. Töötukassa rakendused ja nendevahelised seosed. Uuem versioon riigihanke dokumendis [18] olevast joonisest.

2.2 Varasemad lahendused

Varasemalt on EUIF projektis kasutatud ainult APM ehk rakenduste jõudlusmonitooring (ingl. k. *Application Performance Monitoring*) platvorme, mis kahjuks on osutunud Töötukassa nõuetele ebasobivaks.

2.2.1 New Relic

Esimene versioon EMPIS rakendusest võeti kasutusele aastal 2009, kuid alles 2013 võeti EMPIS rakenduse monitoorimiseks kasutusele New Relic, mis on APM platvorm [21]. New Relicu tasuline plaan osutus lõpuks Töötukassale liiga kulukaks.

2.2.2 Plumbr

Kuna New Relic muutus Töötukassale liiga kulukaks, vahetati see 2016. aastal Nortalis alguse saanud Plumbri vastu. 2020. aastal ostis Plumbri ära Splunk [22] mistõttu läks ka

selle lahenduse kasutamine Töötukassale oluliselt kallimaks.

2.2.3 Prometheus ja Skywalking

Seega otsustas Nortal otsida vabavara, mis sama rolli paremini täidaks ning valiti süsteemide ja rakenduste monitoorimisplatvorm Prometheus ja APM platvorm Skywalking. Nimetatutest viimast töös ei käsitleta.

2.3 Alternatiivsed lahendused

Kuna Prometheus ei ole ainus võimalik monitoorimislahendus, uuris töö autor ka teisi võimalusi süsteemide monitoorimiseks. Selles alampeatükis on välja toodud peamised erinevused Prometheuse ja võrreldava tööriista vahel. Kuna peamine põhjus uue lahenduse otsimiseks oli monitoorimisteenuste hind, siis võrreldatavad monitoorimistööriistad on peamiselt vabavaralised.

2.3.1 Graphite

Graphite on tööriist, mis salvestab monitooringu andmeid ja genereerib neist ajagraafikuid, kuid sellel ei ole andmete kogumise võimalust. Lisaks on Prometheuse andmemudel painlikum ja võimaldab andmete kogumist teostada suvalise intervalli tagant, aga Graphite kirjutab vanad andmed üle uutega pärast kindlat ajavahemikku ning andmete kogumisintervall on muutmatu [23, 24].

2.3.2 InfluxDB

Vabavaraline InfluxDB versioon on väga sarnane Prometheusele. Mõlema andmetüüp toetab mitmedimensionaalset meetrikat. Neil on sarnased andmete tihendusalgoritmid ja mõlemad on mõeldud laiendatavatele süsteemidele [23].

InfluxDB on parem kui selle peamiseks funktsiooniks süsteemis on logimine. Tasulise versiooniga kaasneb ka andmete klastriteks ühendamine (ingl. k. *clustering*), mis on ideaalne pikaajaliseks andmete salvestamiseks [23].

Prometheus on parem kui selle peamiseks funktsiooniks süsteemis on meetrika jälgimine. Seda on lihtsam kasutada ja horisontaalselt skaleerida kui süsteem laieneb. Lisaks on Prometheus täiesti vabavaraline [23].

2.3.3 OpenTSDB

OpenTSDB on baas monitooringu andmete ajateljes salvestamiseks. Seega see on väga sarnane Graphite'ile ja samad võrdlused kehtivad siin. Sellel on vaja lisaks implementeerida Hadloop ja HBase ning seega on komplekssem [23].

2.3.4 Nagios

Nagios on pigem sobilik lihtsamate ja väiksemate süsteemide jälgimiseks, kus piisab musta kasti meetodil monitoorimisest. Kuna tõenäoliselt Töötukassa infosüsteemid järjest suurenevad, tuleb eelistada Prometheusi, mis on parem suuremõõtmelistele süsteemidele [23].

2.3.5 Sensu

Sensu on sobilikum kui vaja on ühildada mitut monitoorimistööriista või kui on vaja jälgida nii sündmusi (ingl. k - *event*) kui ka meetrikat. Prometheusel on parem Kubernetese integratsiooni võimalus, mis EUIF rakendustel on oluline, kuna kõik rakendused on Kubernetese klastris [23, 25].

2.4 Prometheus ja APM-id

Oluline on vahet teha töös pakutaval monitoorimislahendusel ja rohkem levinud APM-tel platvormidel nagu New Relic ja Skywalking. APM tööriistad on kasulikud selleks, et probleeme kooditasandil avastada ja isoleerida. Kuid seal nende võimekus piirdub-ki - suurem osa probleeme ei ole rakenduse-spetsiifilised vaid rakenduse serveri taristu tasemel [2]. Sellist informatsiooni saab koguda meetrika kogumiseks mõeldud monitoorimisplatvormiga nagu Prometheus [26]. APM-ide, nagu Skywalking, piiranguks on see, et neil on ainult Java agent, mis käib monitooritava rakenduse külge, kuid Prometheusi ja Grafanat koos kasutades saab tervet süsteemi jälgida.

2.5 Monitoorimislahenduse komponendid

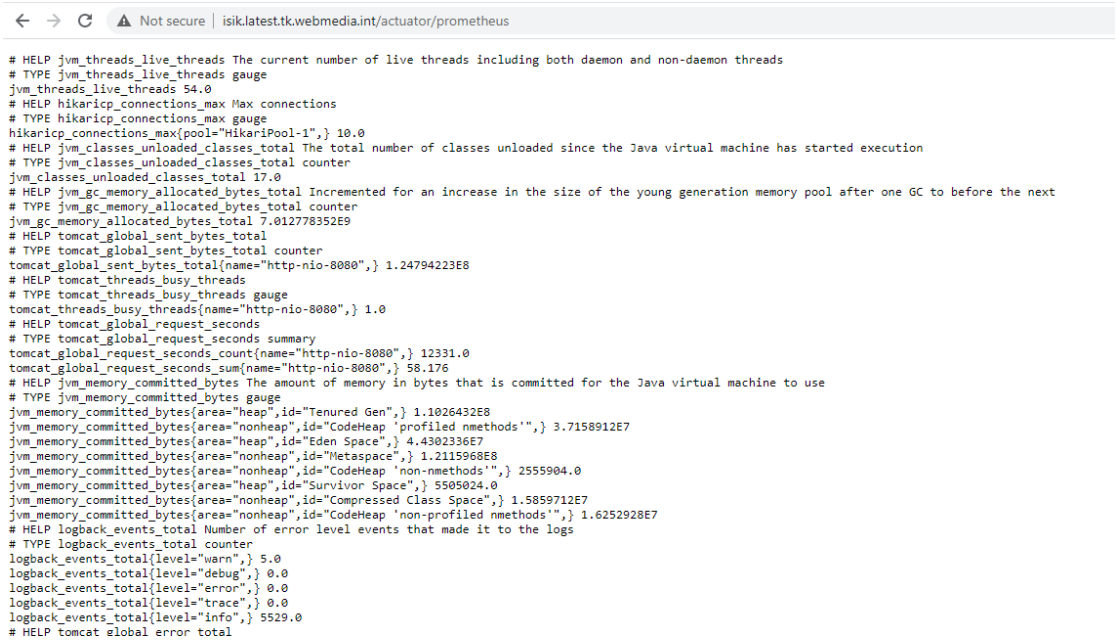
Töös implementeeritav lahendus koosneb kolmest osast. Selles peatükis tutvustatakse neid järjest.

2.5.1 Prometheus

Andmete kogumiseks kasutati monitoorimisplatvormi Prometheus. Prometheus kogub rakendustelt süsteemi kohta andmeid ja on peamine komponent kasutusele võetavast lahendusest. See teeb seda kraapides (ingl. k. *scrape*) HTTP otspunkte. Kraapimine on protsess, millega Prometheuse server pärib seadistatud otspunktidelt monitooringuandmeid. Prometheuse kraape tulemus on näha joonisel 2. Prometheust saab kasutada rakenduse-, andmebaasi- kui ka süsteemimuutujate jälgimiseks kasutades erinevaid lisandplokke ehk *exporter*'eid [26].

Prometheus jookseb EUIF rakendustest eraldi serverina. See kraabib andmeid iga 5 sekundi tagant ning salvestab nad ajajadana. Prometheuselt andmete pärimiseks on oma

päringukeel PromQL (*Prometheus Query Language*), mida saab testida Prometheuse serveri enda kasutajaliideselt [23, 26]. Prometheusel on ka väga hea Dockeri ja Kubernetese tugi, mis on eriti oluline EUIF rakenduste puhul [25].



Joonis 2. Toores Prometheuse kraapimise väljundandmestik. Joonisel on esitatud ainult osa väljundist.

2.5.2 Micrometer

Selleks, et andmeid Prometheusele nähtavaks teha kasutati Micrometerit. Micrometer on JVM'i põhistele rakendustele mõeldud teek. Micrometer paneb kogutud andmed Prometheuse poolt loetavale otspunktile [27].

2.5.3 Grafana

Andmete visualiseerimiseks kasutati Grafanat. Sellega on kogutud andmed tõlgitud inimloetavale kujule ning tehtud graafikud, millega saab rakendusi ja keskkondi ajas jälgida. Grafanas saab ka defineerida teavitusi, et keskkondade haldajad saaks probleemidele reageerida juba enne kui nad ilmnevad [28]. Grafana kasutajaliides ja näidisgraafikud on näha joonisel 3.

Joonis 3. Grafana koondpaneel (ingl. k - *dashboard*) näidisgraafikutega

2.6 Prometheuse eelised

Omadused, mis kallutavad valiku Prometheuse kasuks on Kubernetese laialdane tugi, suurte süsteemide ning horisontaalse laienemise võimalused ja *exporter*'ite kasutusvõimalus, millega on võimalik monitoorimislahendust muuta vastavalt projekti vajadustele.

Alternatiivseid monitoorimistööriistu ja Prometheust on võrreldud tabelis 1. Kuna monitoorimislahenduse maksumus on EUIF-ile oluline, siis võrreldakse monitoorimislahenduste tasuta versioone. APM-ina on tabelis võrreldud peamiselt Skywalkingut, teistel APM-idel võib olla mõningaid mittedrastilisi erinevusi.

Tabel 1. Prometheuse ja alternatiivsete monitoorimistööriistade võrdlus.

	Prometheus	**Graphite**	**InfluxDB**	**OpenTSDB**	**Nagios**	**Sensu**	**APM**
Kubernetese tugi	+	+	+	+	+	-	+
Paindlik andmemudel	+	-	+/-	+	-	+	-
Süsteemi meetrika	+	+	+	+	+	+	-
Lihtne implementeerida	+	+	-	-	+	-	+
Pikaajaline andmete salvestus	-	+	+/-	+	-	+	-
Suurtele süsteemidele	+	+	+	+	-	+	+
Exporter'id laiendusteks	+	-	+	-	-	-	-

3 Lahenduse teostus

Selles peatükis kirjeldatakse lähemalt monitoorimislahenduse üles seadmist.

Prometheus ja Grafana serverid pannakse käima Nortali klastris monitooringu nimeruumi. Seal töötades saab Prometheus kätte nii Nortali testkeskkondade meetrika kui ka TK-test ja LIVE klastri meetrika. Igale monitooritavale rakendusele tuleb seadistada Micrometer, kuid Prometheuse ja Grafana seadistust tuleb teha üks kord ja hiljem on lihtne horisontaalselt laiendada.

3.1 Monitoorimislahenduse teostamine

Lahenduse praktiline osa jaguneb järgnevatesse osadesse:

3.1.1 Micrometer

1. **Micrometer'i seadistamine**

   (a) **IAM** ja **KIPIS** - Mõlemad arendatud Spring Boot 2.1.3-ga. Micrometer on mõeldud kasutuseks rakendustes, kus on Spring Boot 2.0 või hilisem. Micrometeri seadistamine oli lihtne - tuli vaid lisada kaks sõltuvust rakenduste Gradle'i build faili: üks selleks, et Micrometer hakkaks andmeid koguma ja teine selleks, et need andmed otspunktile väljastada.

   ```
   implementation 'io.micrometer:micrometer-registry-
       prometheus'
   implementation 'org.springframework.boot:spring-boot-
       starter-actuator'
   ```

   (b) **EMPIS2** - Arendatud Spring Boot 1.5.11-ga. Lisaks eelnevate sõltuvuste lisamisele tuleb lisada ka Micrometeri lisateek vanemate Spring Boot versioonide jaoks. Sellega laieneb Micrometeri tugi versioonini 1.5.0 või hilisem.

   ```
   implementation group: 'io.micrometer', name: 'micrometer-
       spring-legacy', version: '1.3.17'
   ```

   EMPIS2-s oli vaja ka uuendada Spring Security teegid versioonile 4.1.1 kuna `spring-boot-starter-actuator` ei ühildunud versiooniga 3.2.5.

2. `spring-boot-starter-actuator` väljastab kogutud meetrika otspunktile [29], kuid hea turvalisusega rakendustes ei pääse ilma turvaindentifikaatorita (ingl. k. - *security token*) kuhugile ligi. Kuna erinevalt suuremale osale Töötukassa andmetest, ei ole tegu tundlikku teavet sisaldavate andmetega, siis saab eemaldada sellelt otspunktidilt selle turvanõude. Lisaks tuleb otspunkti nähtavaks tegemiseks rakenduse konfiguratsioonifaili lisada seaded:

```
management:
  health:
    ldap:
      enabled: false
  endpoints:
    web:
      exposure:
        include: health, prometheus, heapdump
```

Sellega väljastatakse andmed otspunktile `http://[rakenduse-url]/actuator/prometheus`. Väljund on näha jooniselt 2.

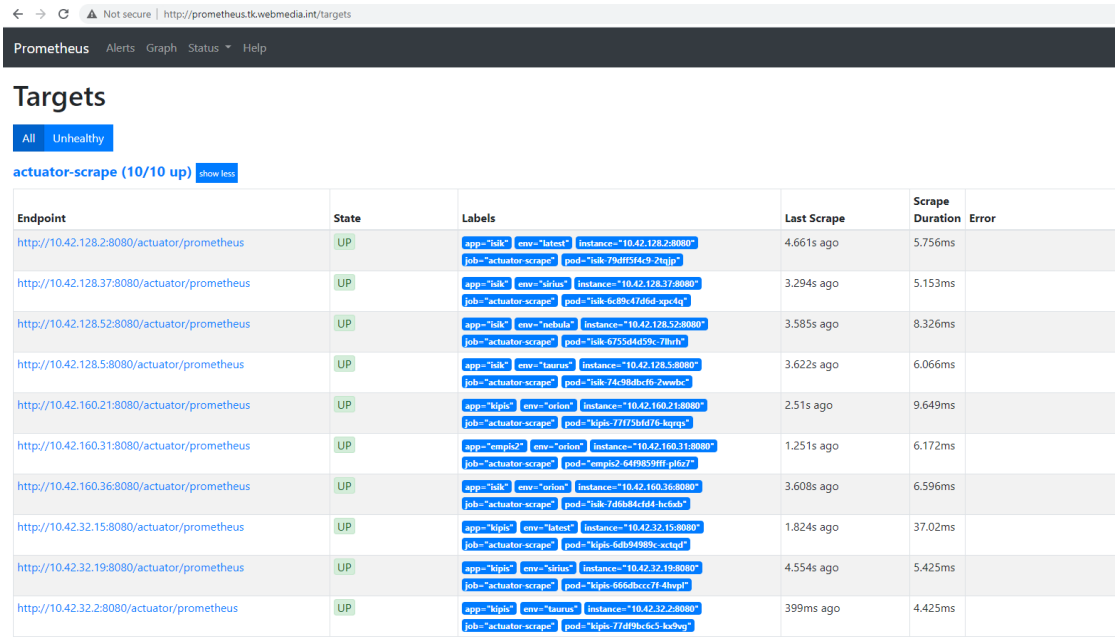
3.1.2 Prometheus

1. **Meetrika kogumine** (Prometheusi seadistamine) - Prometheuse rakendus jookseb teistest rakendustest eraldi monitoring nimeruumis. Suurem osa selle muutmisest käib läbi konfiguratsioonifaili:

```
data:
  prometheus.yml: |-
    global:
      scrape_interval: 5s
      scrape_timeout: 5s
    scrape_configs:
      [<-tk-test conf->] [<-live conf->]
      - job_name: 'actuator-scrape'
        metrics_path: '/actuator/prometheus'
        kubernetes_sd_configs:
          - role: endpoints
            namespaces:
              names:
                - latest
                - sirius
                - taurus
                - orion
                - nebula
        relabel_configs:
          - source_labels: [
              __meta_kubernetes_service_label_monitoring]
            separator: ;
            regex: actuator
            replacement: $1
            action: keep
          - source_labels: [__meta_kubernetes_endpoint_port_name
              ]
            separator: ;
            regex: http
```

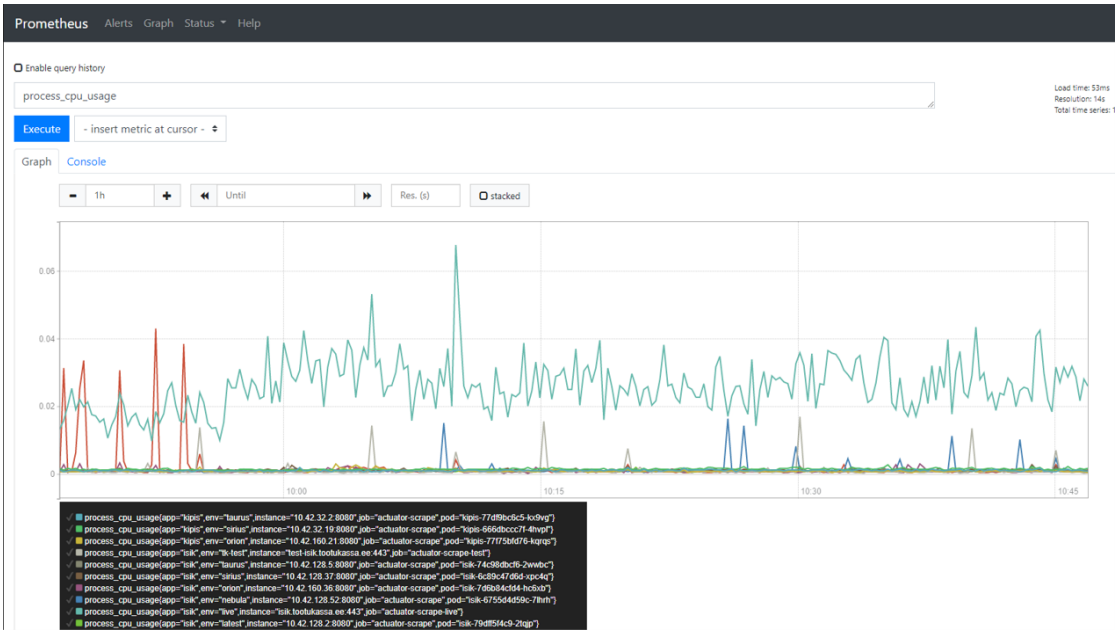
```
    replacement: $1
    action: keep
  - source_labels: [__meta_kubernetes_namespace]
    separator: ;
    regex: (.*)
    target_label: env
    replacement: $1
    action: replace
[<-relabel_config edasi->]
```

Siit on võimalik muuta kraapimiste intervalli (`scrape_interval`). Nortali testkeskkondade monitooritavate rakendusete otspunktid leiab Prometheus üles automaatselt nimeruumide järgi. On võimalik ka seadistada teabesilte (`relabel_configs`), mille järgi saab hiljem Grafanas defineerida muutujaid, et andmeid sorteerida. Kui kõik on seadistatud, siis Prometheuse graafilises liideses kuvatakse sihtotspunktid koos teabesiltidega (joonisel 4).



Joonis 4. Prometheuse leitud sihtotspunktid.

Prometheuse kasutajaliidesest saab ka teha kogutud andmete põhjal päringuid. Joonisel 5 on näha Prometheuse genereeritud graafik rakenduse protsessori kasutusest ehk `process_cpu_usage` rakenduse IAM põhjal. Graafiku all on selle näitaja andmestiku legend koos kogutud muutujatega ja teabesiltidega.



Joonis 5. Prometheuse liideses kogutud andmete kuvamine. Kuvatud on protsessori kasutus.

3.1.3 Grafana

1. Prometheusi **päringute koostamine** ja graafikute seadistamine - kui Prometheus on Grafanas andmeallikaks lisatud, siis saab Prometheuse päringutest paneele teha. Iga paneeli kohta tuleb teha PromQL-keeleline päring. Vajadusel saab paneeli kohta olla ka mitu päringut - näiteks protsessori kasutuse puhul on oluline nii rakendusepõhine protsessori kasutus kui ka terve süsteemi tarbimine. Päringud saab koostada kasutades Prometheuses defineeritud teabesilte. Autor realiseeris kõigi 1. peatükis loetletud oluliste meetrikate jälgimise.

2. **Grafana koondpaneelide koostamine** - olenevalt rollist on Nortali EUIF töötajatel vaja jälgida erinevaid meetrikaid. Näiteks testijatele on olulisem LIVE keskkond ja arendajatele Nortali testkeskkonnad. Selleks on Grafanas võimalik koostada erinevaid koondpaneele, et neil oleks lihtsam jälgida ainult vajalikku infot.

   (a) Rakenduste koondpaneelid - kuvatakse peamiselt oluline rakenduste info. Sorteeritav rakenduste, keskkondade aja ja muude muutujate järgi.

   (b) üldine staatuste koondpaneel - ainult rakenduste ning keskkondade staatuste ja põhiinfo kuvamiseks.

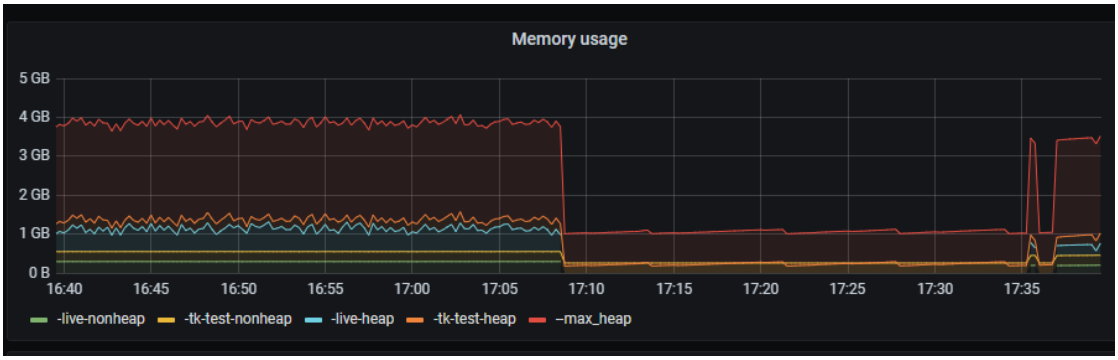
   (c) *playlist*'ide koostamine - kohati on oluline aga, et nii testkeskkondi kui

ka LIVE keskkonda oleks näha. Selleks on Grafanas võimalik seadistada *playlist*'e - funktsionaalsus, millega kindla ajaintervalli tagant vahetub kuval koondpaneel. See on kasulik selleks, et kontoris olevast info-ekraanist saaks läbi Raspberry Pi meetrikat näha.

3.2 Tulemus

Tulemuseks on kogutud andmete kuvamine Grafana abil, mis on jaotatud koondpaneelideks. Kuvatakse arendajatele, testijatele ja muudele süsteemihaldajatele vajalikku infot. Lõpptulemus sisaldab ka võimalust, kuidas kasutajad saavad pärast lahenduse teostamist võimalikult lihtsalt uusi mõõdikuid defineerida, mida jälgida.

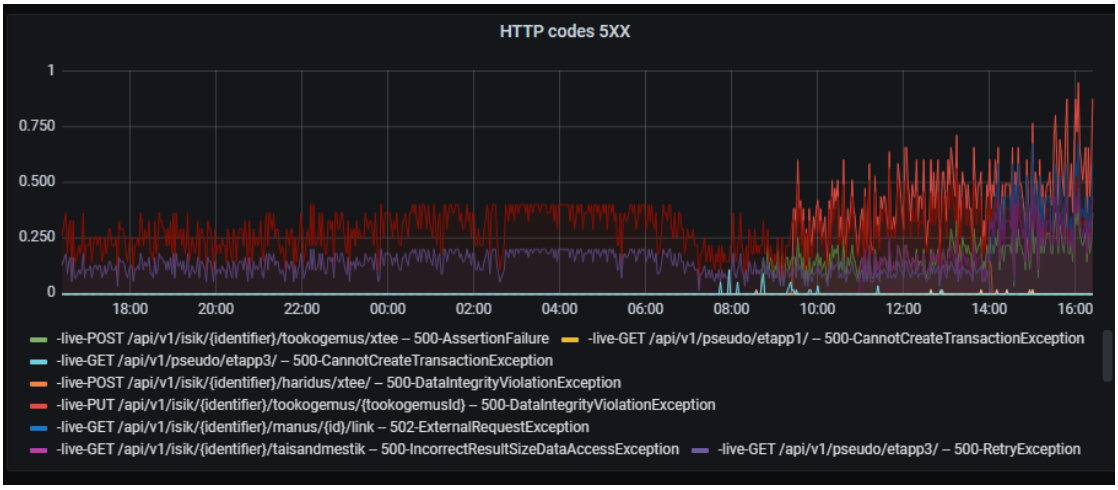
Monitoorimislahenduse edu on näha joonisel 6 on Grafana paneel muutmälu kasutusest IAM tarne ajal.



Joonis 6. Grafana paneel. Mälu kasutus Isikuandmete mooduli tarne ajal. Mälu kasutusest on näha hetki, mil IAM lülitatakse välja, pannakse käima ja seejärel tehakse taaskäivitus.

On võimalik eristada hetki, mil IAM lülitatakse välja, pannakse käima ja, kuna rakenduse käivitusel tekib viga, tehakse seejärel taaskäivitus. Joonisel 7 on näha, kuidas pärast hiljutist tarnet on LIVE keskkonnas veateadete arv suurenenud. Samuti on toodud veateate nimetus ja HTTP kood - sel juhul on suurenenud 503 koodiga vigade esinemine.

Grafana playlist on igal ajal süsteemi haldajatele nähtav, kuna koondpaneelid jooksevad *playlist*'ina kontoris oleval ekraanil, mis on toodud joonisel 8, või vastaval otspunktil kättesaadav.



Joonis 7. Grafana paneel HTTP koodide 5XX veateadete esinemissagedustega. Jooniselt on näha, kuidas LIVE keskkonnas on pärast tarnet veateadete sagedus suurenenud.

Joonis 8. Grafana playlist käimas Nortali Tartu kontoris.

3.3 Võimalikud edasiarendused

Edasiarendusena on plaanis monitoorimislahendust täiustada, et see oleks veelgi kasulikum igapäevases töös. Kõigepealt tuleb lahendus võtta kasutusele **kõigis Nortali EUIF rakendustes**. Kuna mõned neist rakendustest on küllalt vanad, siis selle raames uuritakse Micrometeri võimalusi vanemate Spring Boot ja Springframework jaoks. Seejärel tuleks laialdasemalt Prometheuse *exporter*'eid kasutada, et oleks veelgi rohkem meetrikaid, mida monitoorida:

Andmebaaside meetrika - Prometheusel on võimalus, et jälgida muuhulgas PostgreSQL ja Oracle andmebaase. Sellest meetrikast on plaanis ka teha eraldi Grafana koondpaneel.

Kubernetese meetrika - Oluline on jälgida Kubernetese klastrit ja Dockeri konteinerite infot.

Custom meetrika - Prometheuse *exporter*'eid saab ka ise kirjutada ja seega on võimalik, et rakenduse spetsiifilisi mõõdikuid Grafanas kuvada.

Kokkuvõte

Töö eesmärk on implementeerida monitoorimislahendus mitmerakenduseliste infosüsteemide veebiserverite jälgimiseks Nortali EUIF projekti infosüsteemi jaoks, et süsteemi haldajad saaksid efektiivsemalt probleeme leida ja parandada. Lahendus peab toetama mitmeid raamistikke, et monitoorida erinevaid süsteemi- ja rakendusemuutujaid võttes kasutusele monitoorimistööriistad Micrometer, Prometheus ja Grafana.

Esmalt tutvustati suurte monitoorimislahenduseta süsteemide võimalikke probleeme ning nende ohtlikkust. Kirjeldati erinevaid indikaatoreid, mille reaalajas nägemine on oluline, et süsteemi haldajad saaksid paremini tagada selle korrektse toimimise.

Töö teises peatükis kirjeldati mõningaid Nortali EUIF rakendusi, mille monitooring teeks arendajate tööd lihtsamaks. Tutvustati monitoorimislahenduses kasutatud monitoorimistööriistu Micrometerit, Prometheusi ja Grafanat. Kasutatud tööriistu võrreldi töö käigus ka alternatiivsete süsteemide monitoorimisplatvormide ning rakenduste jõudlusmonitooringu vahenditega ja kirjeldati, kuidas varem EUIF projektis monitooringut tehti.

Töö kolmandas peatükis kirjeldati kõigi kolme monitoorimistööriista - Prometheuse, Micrometeri ja Grafana - implementatsiooni kolme EUIF rakenduse jaoks. Implementatsiooniprotsess on piisavalt universaalne, et seda saaks korrata ka teiste süsteemide jaoks.

Tulemuseks on monitoorimislahendus, mille abil saab reaalajas jälgida Nortali EUIF projekti ja Eesti Töötukassa infosüsteemide korrasust. Implementeeritud lahendus on piisavalt lihtne, et seda saaks horisontaalselt skaleerida teistele EUIF rakendustele ning piisavalt paindlik, et seda saaks muuta vastavalt edasistele vajadustele. Monitoorimislahenduse edasiarendusvõimalusi on palju, et süsteemi haldamist veelgi sujuvamaks teha.

Viidatud kirjandus

[1] Romana Gnatyk. *Microservices vs Monolith: which architecture is the best choice for your business?* 2018. URL: https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/ (vaadatud 10.12.2020).

[2] Jorge Salamero Sanz. *How to instrument code: Custom metrics vs APM vs OpenTracing*. 2018. URL: https://sysdig.com/blog/how-to-instrument-code-custom-metrics-vs-apm-vs-opentracing/ (vaadatud 06.04.2021).

[3] *What is IT Monitoring?* URL: https://www.opsview.com/resources/how-to/whitepapers/what-is-it-monitoring (vaadatud 22.04.2021).

[4] *Ametniku soovitussõnastik. monitooring*. URL: https://www.eki.ee/dict/ametnik/index.cgi?Q=monitooring&F=M&C06=ee (vaadatud 22.04.2021).

[5] Aaron Leskiw. *Application Monitoring – What is it and Why its So Important for your Business*. URL: https://www.networkmanagementsoftware.com/application-monitoring-2/ (vaadatud 19.04.2021).

[6] Sander Pärn. „Suurte failide puhverdamine Nortali eHealth projekti näitel". Bakalaureusetöö. Tartu Ülikool. URL: https://dspace.ut.ee/bitstream/handle/10062/66281/thesis.pdf?sequence=1&isAllowed=y (vaadatud 19.04.2021).

[7] Ville Ahlgren *et al.* „Large-Scale System Monitoring Experiences and Recommendations". Teoses: september 2018, lk. 532–542. DOI: 10.1109/CLUSTER.2018.00069.

[8] *Prometheuse dokumentatsioon. Jobs and instances*. URL: https://prometheus.io/docs/concepts/jobs_instances/ (vaadatud 06.05.2021).

[9] Eric Boersma. *Memory leak detection - How to find, eliminate, and avoid*. 2020. URL: https://raygun.com/blog/memory-leak-detection/ (vaadatud 03.05.2021).

[10] *Garbage collection pauses*. URL: https://docs.datastax.com/en/dse-trblshoot/doc/troubleshooting/gcPauses.html (vaadatud 03.05.2021).

[11] *Why CPU monitoring matters (and what PRTG can do for you)*. URL: https://blog.paessler.com/why-cpu-monitoring-matters (vaadatud 03.05.2021).

[12] *IBM Watson Content Analytics. HTTP status codes returned to the Web crawler*. URL: https://www.ibm.com/docs/cs/wca/3.0.0?topic=activity-http-status-codes-returned-web-crawler (vaadatud 03.05.2021).

[13] *Key Server Performance Metrics For Actionable Monitoring*. URL: https://www.appdynamics.com/product/server-visibility/server-performance-metrics (vaadatud 03.05.2021).

[14] *Andmevahetuskiht X-tee*. URL: https://www.ria.ee/et/riigi-infosusteem/andmevahetuskiht-x-tee.html (vaadatud 12.04.2021).

[15] Matthew Tyson. *What is JDBC? Introduction to Java Database Connectivity*. 2019. URL: https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database-connectivity.html (vaadatud 03.05.2021).

[16] Chandra Shekhar Pandey. *Detecting and Resolving Database Connection Leaks with Java Applications*. 2020. URL: https://dzone.com/articles/detecting-and-resolving-database-connection-leaks (vaadatud 03.05.2021).

[17] Federico Plantera. *A story of resilience: tackling unemployment through innovation in Estonia*. 2018. URL: https://e-estonia.com/story-resilience-tackling-unemployment-through-innovation (vaadatud 20.04.2021).

[18] *Riigihanke dokument. Eesti Töötukassa tööturuteenuste ja -toetuste infosüsteemi (EMPIS/EMPIS2) jätkuarendus- ja hooldustööd. RD lisa 1a. Raamlepingu eseme tehniline kirjeldus*. 2019. URL: https://riigihanked.riik.ee/rhr-web/#/procurement/1589937/documents/source-document?documentOldId=13298530 (vaadatud 20.04.2021).

[19] *Aranea framework*. URL: http://nortal.github.io/araneaframework/ (vaadatud 19.04.2021).

[20] Jevgeni Kabanov. „Aranea-A Web Development and Integration Framework“. Magistritöö. 2007. URL: https://dspace.ut.ee/handle/10062/2544.

[21] Mats Juhanson. „Veebirakenduse monitoorimislahenduse realiseerimine Eesti Töötukassa infosüsteemi näitel“. Bakalaureusetöö. Tartu Ülikool, 2013.

[22] Priit Potter. *Plumbr has been acquired by Splunk*. 2020. URL: https://plumbr.io/blog/plumbr-blog/acquired-by-splunk (vaadatud 06.04.2021).

[23] *Prometheuse dokumentatsioon. Võrdlus alternatiividega*. URL: https://prometheus.io/docs/introduction/comparison/ (vaadatud 21.04.2021).

[24] *Graphite dokumentatsioon. Ülevaade*. URL: https://graphite.readthedocs.io/en/latest/overview.html (vaadatud 21.04.2021).

[25] Mateo Burillo. *Kubernetes monitoring with Prometheus, the ultimate guide*. URL: https://sysdig.com/blog/kubernetes-monitoring-prometheus/ (vaadatud 22.04.2021).

[26] *Prometheuse dokumentatsioon*. URL: https://prometheus.io/docs/introduction/overview/ (vaadatud 10.12.2020).

[27] *Micrometeri dokumentatsioon*. URL: https://micrometer.io/docs/concepts#_purpose (vaadatud 10.12.2020).

[28] *Grafana koduleheküIg*. 2016. URL: https://grafana.com/grafana/ (vaadatud 10.12.2020).

[29] *Spring Boot Actuator: Production-ready Features*. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-features.html#production-ready-enabling (vaadatud 07.04.2021).

Lisad

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Fred Hansen**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Modernse monitooringu kasutuselevõtt EUIF infosüsteemide näitel,

   mille juhendaja(d) on Piret Luik,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Fred Hansen
07.05.2021