TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tauri Tuuling 206489IAAB

OPENTELEMETRY STANDARDI PÕHISE SEIRESÜSTEEMI JUURUTAMINE REGISTRITE JA INFOSÜSTEEMIDE KESKUSES

Bakalaureusetöö

Juhendaja: Edmund Laugasson
MSc

Tallinn 2024

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.

Author: Tauri Tuuling

14.05.2024

Abstract

The objective of this bachelor's thesis is to create a monitoring system focused on applications for the Centre of Registers and Information Systems, which is based on the OpenTelemetry framework. The solution uses free and open-source components.

The thesis introduces the OpenTelemetry framework, its components, and the possibilities for instrumenting applications. It also provides an overview of the monitoring systems used in the institution and the general need for monitoring. The work describes the creation and configuration of the system, detailing the setup of Grafana, Tempo, Prometheus, and Loki. Additionally, it addresses the instrumentation of DotNet applications and their integration with the monitoring system.

The thesis analyzes the compliance of the created system with the institution's requirements and provides an assessment of the system. As a result of the analysis, the author found that the system meets 81.25% of the requirements and is a functional monitoring system that supports application logs, metrics, and traces, allowing them to be displayed to users of the monitoring system. In the final chapter, the author gives recommendations for improving the system.

The thesis is written in Estonian and is 39 pages long, including 8 chapters, 13 figures and 1 table.

Annotatsioon

Opentelemetry standardi põhise Seiresüsteemi juurutamine Registrite ja Infosüsteemide Keskuses

Käesoleva bakalaureusetöö eesmärgiks on luua rakendustele keskendunud seiresüsteem, Registrite ja Infosüsteemide keskusele, mis põhineb Opentelemtry raamistikul. Lahenduses kasutatkse vabavaralisi ja avatud lähtekoodiga komponente.

Lõputöös tutvustatakse OpenTelemetry raamistikku, selle komponente ja rakenduste instrumenteerimise võimalusi. Samuti antakse ülevaade asutuses kasutusel olevatest seiresüsteemidest ja üldisest seirevajadusest. Töös kirjeldatakse süsteemi loomist ja seadistamist, kirjeldades Grafana, Tempo, Prometheusi ja Loki seadistamist. Lisaks käsitletakse DotNet rakenduste instrumenteerimist ja seiresüsteemiga integreerimist.

Töös analüüsitakse loodud süsteemi vastavust asutuse nõuetele, ning antakse hinnang loodud süsteemile. Analüüsi tulemusena leidis autor, et süsteem vastab 81.25% nõuetest on töötav seiresüsteem, mis toetab rakenduste logisid, mõõdikuid ja jälgi, ning võimaldab neid kuvada seiresüsteemi kasutajatele. Töö viimases peatükis annab autorsoovitusi süsteemi täiendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 39 leheküljel, 8 peatükki, 13 joonist, 1 tabelit.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* liides, mis võimaldab pärida rakendust
IIS	*Internet Information Services*, Microsofti poolt pakutav veebiserver
Oauth	*Open Authorization*, avatud standard autentimiseks
VM	Virtual Machine, virtuaalmasin
DotNet	Microsofti poolt arendatud tarkvara raamistik
SLA	*Service level agreement*, leping, mis määratleb teenuse kvaliteedi, saadavuse ja pakkuja vastutuse
Powershell	Microsofti Microsofti välja töötatud käsuinterpretaator ja skriptimiskeel, mis põhineb .NET-raamistikul ja on peamiselt loodud süsteemiadministreerimise ja automatiseerimise ülesannete jaoks
SDK	*Software Development Kit*, kogum tarkvarakomponente, tööriistu ja dokumentatsiooni, mis võimaldavad arendajatel luua tarkvararakendusi konkreetsele platvormile, raamistikule või programmeerimiskeelele

Sisukord

Joonistе loetelu

Tabelite loetelu

1. Sissejuhatus

Registrite ja Infosüsteemide keskuse halduses on mitmeid E-riigi toimimiseks vajalikke rakendusi, näiteks e-äriregister, e-kinnisturaamat ja kohtuinfosüsteem. Asutuse rakendustel on erinev arhitektuur ning puudub ühtne süsteem, mis suudaks koguda, säilitada ja kuvada rakenduste seireandmeid. Nende probleemide lahendamiseks soovib asutus kasutusele võtta uue seiresüsteemi, mis põhineks OpenTelemetry raamistikul.

Käesolev bakalaureusetöö keskendub OpenTelemetry standardi põhise seiresüsteemi juurutamisele Registrite ja Infosüsteemide Keskuses. Töö eesmärgiks on luua süsteem, mis võimaldab rakenduste paremat jälgitavust ja seiret ning kasutab OpenTelemetry jälgitavuse raamistikku.

Töös antakse ülevaade Registrite ja Infosüsteemide keskusest ja OpenTelemetry jälgitavuse raamistikust. Töös kirjeldatakse asutuse praeguste seirelahenduste puudusi ja probleeme. Töö käigus kirjeldatakse süsteemi, selle nõuetele vastamist ja testimise metoodika. Töö käigus juurutatakse uus seiresüsteem ning analüüsitakse uue süsteemi vastavust asutuse nõuetele.

2. Taust

Selles peatükis tutvustatakse asutust, mille näitel lõputöö valmib. Peatükk käsitleb asutuse poolt valitud Opentelemetry komponente ja mõisteid.

2.1 Asutuse tutvustus

Registrite ja Infosüsteemide keskus on Justiitsministeeriumi haldusalas olev asutus, mille eesmärgiks on luua häid integreeritud e-teenuseid pakkuv innovaatiline keskkond riigihaldus-, õigus- ja kriminaalpoliitika efektiivsemaks rakendamiseks. RIK arendab ja haldab riigile ning kodanikele väga olulisi registreid ja infosüsteeme nagu näiteks e-äriregister, e-notar, e- kinnistusraamat, kohtuinfosüsteem, kriminaalhooldusregister, kinnipeetavate register, karistusregister, e-toimik, elektrooniline Riigi Teataja.[1] Registrite ja infosüsteemide keskus keskendub rakenduste arendusele ja haldamisele. Asutuse põhilised seiratavad objektid on:

- veebirakendused
- andmebaasid
- virtuaalmasinad

2.2 Mis on Opentelemetry

Opentelemetry on jälgitavuse raamistik ja tööriistakomplekt, mille eesmärk on luua ja hallata telemeetriaandmeid nagu jäljed, mõõdikud ja logid. Opentelemetry on müüja- ja tööristavaba, mis tähendab, et seda saab kasutada mitmessuguste vaatluse komponentidega.

Opentelemetry oluline eesmärk on võimaldada lihtsat rakenduste või süsteemide instrumenteerimist, sõltumata keelest või infrastruktuurist. Opentelemetry ei visualiseeri või hoiusta andmeid, selle jaoks on vaja kasutada teisi tööriistu[2].

2.2.1 Mis on insturmenteerimine

Süsteemi jälgimiseks on vaja see instrumenteerida ehk süsteemi komponentide kood peab väljastama jälgi, mõõdikuid ja logisid. Opentelemetry pakub kahte lahendust koodi instrumenteerimiseks. Need on koodipõhine- ja nullkoodilahendus [3].

Koodipõhine lahendus

Koodipõhine lahendus võimaldab genereerida rakenduselt rohkem andmeid. Antud lahendust kasutades on võimalik luua mõõdikuid, jälgi ja logisid ning neid väljastada, kasutades Opentelemetry rakenduseliidest ja tarkvaraarenduskomplekti. Koodipõhist lahendust kasutades on vajalik süsteemi või rakenduse arenduse poolelt lisatöö [3, 4].

Nullkoodilahendus

Nullkoodilahendus võimaldab rakendusi instrumenteerida lähtekoodi muutmata. Nullkoodi instrumenteerimine lisab OpenTelemetry API ja SDK võimalused rakendusele tavaliselt agendi või agendi-sarnase paigaldusena [5].

2.2.2 Opentelemetry Collector

Opentelemetry Collector on rakendus, mis kogub, töötleb ja saadab edasi telemeetria andmeid. Collector võimaldab koguda telemeetria andmeid agentidelt ja instrumenteeritud rakendustel ning need andmed edastada teistele seiresüsteemi tagaosa rakendustele. Rakendus toetab erinevaid avatud jälgitavuse andmevorminguid, näiteks prometheus ja Jaeger.[6]

OpenTelemetry Collectorit saab kasutata virtuaalmasinas või konteineris. Collectorit on võimalik kasutada agendina andmete kogumiseks või lüüsina teiste telemeetria andmeid eksportivate teenuste andmete kokku kogumiseks.[7]

2.3 Seireandmete tüübid

Seireandmeid saab liigitada kolmeks tüübiks:

Mõõdikud

Mõõdikud on numbrilised mõõtmisandmed. Mõõdikud võivad olla operatsioonisüsteemi poolt kogutavad andmed, näiteks protsessori ja muutmälu kasutus või kõrgema taseme andmed, näiteks veebiserveri päringute arv sekundis. Mõõdikud võimaldavad saada ülevaate süsteemi ressursikasutusest, toimimisest ja ajaloolistest trendidest. [8]

Logid

Logid on kirjed, mis kirjeldavad süsteemis toiminud sündmust, näiteks vigu või tehinguid.

Logisid genereerivad enamus IT infrastruktuuri seadmed ja süsteeme. Logide kaudu on võimalik administraatoril leida rakenduses tekkinud vigu, analüüsida jõudlust või tuvastada küberrünnakuid. [9]

Jäljed

Jälg kujutab päringu või toimingu teekonda sõlmede vahel. Jäljed võimaldavad leida probleeme või kitsaskohti erinevate süsteemide või komponentide vahel, andes ülevaate päringu teekonnast. [10]

3. Ülesandepüstitus

Asutuses on kasutusel mitmeid erinevaid monitooringulahendusi, enamus neist on sisseostetud teenused ja ei ole hallatud asutuse enda poolt. Süsteemides muudatuste tegemiseks tuleb see tellida teenusepakkujalt, see teeb protsessi tülikaks ja aeglaseks. Mõõdikute jaoks on asutuses kasutuses Zabbix, mis kogub andmeid serverite resursikasutusest ja ka veebiserveri andmeid. Logide jaoks on asutuses kasutusel Graylog, võrgukettad, andmebaasid. Erinevad rakendused ja süsteemid saadavad logisid erinevatesse kohtadesse ja ühte keskset lahendust ei ole. Jälgi asutuses ükski monitooringurakendus ei kogu. Varasemalt oli asutusel kasutusel Dynatrace, mis võimaldas jälgida rakendusi ja andmebaase. Eelmisel aastal lõpetas asutus Dynatrace'i litsentsi, kuna süsteem ei leidnud piisavalt kasutust ja litsentsi maksumus oli liiga kõrge. Asutusel puudub ühtne süsteem, millega jälgida rakendusi ja tuvastada probleeme nende töös.

3.1 Soovituslik lahendus

Probleemi lahendamiseks soovib asutus kasutusele võtta uue seiresüsteemi, mis suudaks monitoorida rakendusi. Uue süsteemi eesmärk on tõhustada rakenduste veatuvastus protsessi. Lahenduses soovib asutus kasutada vabavaralisi komponente, nagu Tempo, Loki, Grafana, Prometheus ja OpenTelemetry raamistikku. Algselt planeeriti süsteem paigaldada Kubernetese kobarasse, aga kobara jõudlusprobleemide tõttu soovitas Kubernetese haldur paigaldada eraldi virtuaalmasinale. Süsteemi asutusele sobivuse hindamiseks seatakse üles seiresüsteem testkeskkonda.

Uue seiresüsteemi juurutamisel tuleb lähtuda asutuse nõuetest ja vajadustest. Järgnevalt on väljatoodud nõuded, mida uus seiresüsteem peaks täitma:

Funktsionaalsed nõuded:

1. Rakenduste mõõdikute säilitamise, kogumise ja visualiseerimise tugi
2. Rakenduste logide säilitamise, kogumise ja visualiseerimise tugi
3. Rakenduste jälgede säilitamise, kogumise ja visualiseerimise tugi
4. Turvalisus ja ligipääsu haldamine
5. Paindlikus ja universaalsus

Mittefunktsionaalsed nõuded:

1. Vabavaralised ja avatud lähtekoodiga komponendid
2. Asutuse poolt hallatav ja muudetav

Rakenduste mõõdikute, logide ja jälgede tugi: Süsteem peab suutma koguda, säilitada ja visualiseerida rakenduste poolt genereeritud logisid. Antud nõue on oluline, et kõik kolm andmetüüpi oleksid ligipääsetavad ühest keskkonnast, et pakkuda mugavamat ja tõhusamat lahendust rakenduste veaotsinguteks. Süsteem peab suutma andmeid talletada kaks kuud.

Turvalisus ja ligipääsu haldamine: Süsteemi ligipääsu peab saama piirata. Süsteem peab toetama OAuth2 standardit, et ühilduda asutuses kasutusel oleva autentimisteenusega.

Paindlikus ja universaalsus: Lahendus peab toetama erinevaid programeerimiskeeli ja rakenduste arhitektuure, et pakkuda ühtset lahendust kõikide rakenduste jälgimiseks.

Vabavaralised ja avatud lähtekoodiga komponendid: Süsteem peab olema vaba ja avatud lähtekoodiga, et vähendada kulusid ja võimaldada kohandad süsteemi vastavalt asutuse vajadustele. Avatud lähtekoodiga süsteemidel on ka suurem kogukonna tugi nii turvauuenduste kui ka dokumentatsiooni osas.

Asutuse poolt hallatav : Lahendus peab olema hallatav asutuse poolt ja kõik andmed peavad jääma asutuse siseseks. See tagab andmete turvalisuse ja konfidentsiaalsuse. Lisaks peaks olema süsteem asutuse administraatorite poolt muudetav, et süsteemi oleks võimalik vajadusel kohandada vastavalt asutuse vajadustele.

Käideldavus: Lahendusele pole määratud eraldi SLAd ning süsteemi töös antud faasis on lubatavad katkestused, kuna arendus- ja haldustiimid soovivad süsteemi testida peab teatud käideldavus olema tagatud. Süsteemi käideldavuse protsent nädalas peab olema 95 ehk katkestust võib olla 8.4 tundi nädalas.

4. Metoodika

Käesolevas peatükis tutvustab autor süsteemi analüüsi metoodikat. Seiresüsteemi analüüsitakse 7. peatükis. Süsteemi hindamiseks koostatakse hindamismudel, mille järgi hinnatakse süsteemi vastavust nõuetega.

4.1 Süsteemi analüüsi metoodika

Seiresüsteemi analüüsimiseks püstitatakse virtuaallabor, kus seadistatakse kõik seiresüsteemi komponendid. Loodud süsteemi võrreldakse 3 peatükis väljatoodud nõuetega. Nõudeid kontrollitakse testides loodud süsteemi.

Süsteem paigaldatakse ühele Linuxi Rocky Linux viruaalserverile, antud serverile paigaldatakse Grafana, Tempo, Prometheus, Loki ja Opentelemetry Collector. Süsteem kasutab ka ühte Collectorit, mis on paigaldatud Kubernetese klastrisse. Kubernetese klastris asuv Collector on varem paigaldatud asutuse arhitekti poolt ning edastab rakendusserveritelt tulevaid andmeid ja Kubernetese klastris olevate Java rakenduste andmeid. Ühe rakendusserveri DotNet rakendused instrumenteeritakse ja seadistatakse andmeid edastama Kubernetese klastris olevale Opentelemetry Collectorile.

Lisas 2 on kujutatud süsteemi struktuuri. Rakendusserveritest tulevad andmed saadetakse Kuberneteses asuvasse Collectorisse. Kuberneteses asuv Collector kogub logisid ja instrumenteeritud rakenduste mõõdikuid ja jälgi ning edastab klastri ja väljast tulevate rakendusete seireandmed virtuaalserveri Collectorile. Viruaalserveril asuv OpenTelemetry Collector edastab mõõdikud Prometheusile, logid Lokile ja jäljed Tempole. Tempo, Loki ja Prometheus talletavad andmed ja päringu alusel edastavad andmed visualiseerimiseks Grafanale.

4.2 Süsteemi testimine ja hindamine

Süsteemi testitakse ja analüüsitakse vastavalt allpool välja toodud nõuetele:

Mõõdikute, logide ja jälgede tugi

Mõõdikute, logide ja jälgede testimisel vaadatakse, kas on võimalik näha hetkeseisu- ja varasemaid andmeid. Testitakse ka visualiseerimise võimalusi, kas on võimalik koostada

ja kuvada graafikuid ja tabeleid antud andmetega.

Turvalisus ja ligipääsu haldamine

Turvalisuse ja ligipääsu haldamise testimisel vaadatakse, kas andmeid on võimalik lugeda ilma kasutajakontot omamata. Katsetatakse, kas läbi asutuse autentimisteenuse on võimalik sisse logida. Administraatori vaatest proovitakse kasutajate ligipääse piirata ja anda erinevatele andmeallikatele, armatuurlaudadele ja üldistele andmetele. Testitakse ka lugemis- ja muutmisõiguste andmist kasutajatele.

Paindlikkus ja universaalsus

Paindlikkuse ja universaalsuse puhul testitakse kas süsteem on suuteline toetama erineva arhitektuuriga rakendusi. Lahendus peab suutma koguda andmeid monoliitrakenduste ja mikroteenuste kohta. Süsteem peab koguma andmeid DotNet raamistiku ja Java rakenduste kohta.

Vabavaralised ja avatud lähtekoodiga komponendid

Antud nõude kontrollimiseks vaadatakse kasutatud komponentide litsentse, et need vastaksid vabavara litsentsidele. Litsentside puhul analüüsitakse, mida litsents lubab teha lähtekoodiga ja millega peab edasiarenduste korral arvestama.

Hallatavus

Hallatavuse poole pealt testitakse, kas asutuse administraatoritel on ligipääs ja õigused süsteemis muudatusi teha. Testitakse, kas administraatorid saavad muuta konfiguratsioonifaile ja teha muudatusi komponentide kasutajaliideses. Vaadatakse kus asuvad andmed, kas need on täielikult asutuse sisesed või läheb andmeid ka asutusest välja.

Käideldavus

Käideldavuse testimiseks mõõdetakse süsteemi katkestusaega. Süsteemi katkestusaja jälgimiseks kasutatakse Zabbixis olevaid seireandmeid ja kasutajate teavitusi süsteemi tõrgete kohta. Süsteemi käideldavust loetakse rahuldavaks, kui katkestuste aeg nädalas on alla 8.4 tunni.

Hindamismudel

Süsteemi hindamiseks annab autor igale nõudele vastamise eest punkte. Punkte jaotatakse järgmiselt:

- Nõue on täidetud: 1 punkt
- Nõue on osaliselt täidetud või pole lõplikult implementeeritud: 0.5 punkti
- Nõue pole täidetud: 0 punkti

5. Uue seiresüsteemi juurutamine

Peatükis käsitletakse seiresüsteemi juurutamist Registrite ja Infosüsteemide keskuses. Selle peatüki eesmärk on selgitada, kuidas peatükis 4.1. kirjeldatud süsteemid on seadistatud ja paigaldatud.

5.1 OpenTelemetry Collectori paigaldamine ja seadistamine

Paigaldamiseks tuleb kõigepealt alla laadida rpm-paket OpenTelemetry Githubist[1] ja kasutades WinSCP programmi tõsta virtuaalmasina kohalikule kettale. Käsitsi paigaldamiseks tuleb kasutada järgmist käsku:

```
sudo rpm -i otelcol_0.90.0_linux_amd64.rpm
```

Pärast paigaldamist saab Collectorit seadistada "config.yaml" failist. Konfiguratsioonis määratakse vastuvõetavad otspunktid, andmete töötlejad, edastajad ja teenused.

Joonis 1. kujutab OpenTelemetry Collectori "Processors" konfiguratsiooni, mis lisab logidele teenuse nimega silte, et logisid oleks võimalik eristada ja paremini otsida.

```
processors:
  batch:
  attributes:
    actions:
      - cation: insert
        key: loki.attribute.labels
        value: event.domain
    resource:
      actions:
        - cation: insert
          key: loki.resource.labels
          value: service.name
```

Joonis 1. *Collectori töötlejad*

Joonis 2. kujutab eksportijate konfiguratsiooni. Konfiguratsioonis on määratud prometheusi otspunkt, ning määratud "resource_to_telemetry_conversion", mis muudab resursi atribuudid mõõdikute siltideks. Loki eksportija sätetes muudetakse "exporter", "instance" ja

[1] https://github.com/open-telemetry/opentelemetry-collector-releases/releases/

"level" atribuudid siltideks. Tempo suudab vastu võtta OpenTelemetry vormingus jälgi ning eraldi eksportijat ei vaja.

```
exporters:
  logging:
    verbosity: basic
  prometheus:
    endpoint: 'Prometheusi otspunkti aadress'
    const labels:
      environment: test
    send_timestamps: true metric_expiration: 5m
    resource_to_telemetry_conversion:
      enabled: true
  loki:
    endpoint: 'loki otspunkti aadress'
    default labels enabled:
      exporter: true
      instance: true
      level: true
  otlphttp:
    endpoint:   'Tempo otspunkti aadress'
```

Joonis 2. *Collectori eksportijate konfiguratsioon*

Joonis 3. kujutab teenuste konfiguratsiooni, kus määratakse ära "pipeline"."Pipeline" sektsioonis seadistatakse millistest komponentidest andmed pärinevad, millised süsteemi osad neid töötlevad ja edastavad.

```
service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [logging, otlphttp]
    metrics:
      receivers: [otlp]
      processors: [batch]
      exporters: [logging, prometheus]
    logs:
      receivers: [otlp]
      processors: [batch, attributes, resource]
      exporters: [logging, loki]
```

Joonis 3. *Collectori teenused*

5.2 Prometheusi paigaldamine ja seadistamine

Prometheus vastutab süsteemis mõõdikute talletamise eest. Prometheus saab andmeid OpenTelemetry Collectori käest, ning päringu alusel edastab mõõdikud Grafanale. Joonisel 4 on kujutatud Prometheusi teenuse konfiguratsiooni, kus on seadistatud parameetrid: mõõdikute talletamise asukoht, säilitamise aeg ja lubatud "Remote rewrite receiver", et Tempo saaks jälgedest genereeritud mõõdikuid Prometheusi salvesatada.

```
ExecStart=/usr/local/bin/prometheus
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /data2/prometheus \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries \
--enable-feature-remote-write-receiver \
--storage.tsdb.retention.time=60d
```

Joonis 4. *Prometheus teenuse sätted*

OpenTelemetry Collectorist andmete saamiseks kasutatakse Prometheus "scrape" metoodikat. Joonisel 5 on kujutatud prometheus.yml faili sisu, kus määratakse ära OpenTelemetry Collectori otspunkt, millelt Prometheus andmeid küsib. Lisaks on ära määratud "scrape_interval" muutujaga andmete kogumise intervalliks viis sekundit.

```
global:
  scrape_interval: 5s
scrape_configs:
  - job_name: 'opentelemetry_collector'
  statuc_configs:
    - targets: ['OpenTelemetry Prometheus eksportija otspunkt']
```

Joonis 5. Prometheus scrape sätted

5.3 Loki paigaldamine ja seadistamine

Loki on vajalik süsteemis, et hoiustada logisid. Lisaks sellele on võimalik Lokist logisid päringu alusel Grafanas visualiseerida ja lugeda. Lisa 8 kujutab Loki sätteid, kus on seadistatud Loki andmesalvestuspind kohalikule kettale. Loki salvestab antud konfiguratsiooniga failid kohalikule kettale.

Joonis 6 kujutab Loki andmete säilitamise sätted, kus määratakse logide säilitamise aeg. Sätetes määratakse ka kustutamise intervalliks 60 minutit ja viiteajaks üks tund. Viiteaeg on oluline, et kõik viited kustutatavatele logidele oleks aegunud, vastasel korral võib logide päring ebaõnnestuda.

```
compactor:
  working_directory: /data2/retention
  compaction_interval: 60m
  retention_enabled: true
  retention_delete_delay: 1h
  retention_delete_worker_count: 150
  delete_request_store: tsdb

limits_config:
  retention_period: 1440h
```

Joonis 6. *Loki logide säilitamise sätted*

5.4 Tempo paigaldamine ja seadistamine

Tempo võimaldab talletada ja teha päringuid jälgede kohta. Joonisel 7 on seadistatud tempo salvestama andmeid kohalikule kettale.

```
storage:
  trace:
    backend: local
    wal:
      path: /data2/tempo/wal
    local:
      path: /data2/tempo/blocks
```

Joonis 7. *Tempo andmete säilitamise konfiguratsioon*

Tempo suudab genereerida ka mõõdikuid jälgedest. See võimaldab kuvada Grafanas teenuse graafikuid, mis visualiseerivad jälgitava süsteemi erinevate komponentide vahelist suhtlust. Teenuse graafikute kuvamiseks tuleb seadistada Tempo mõõdikuid genereerima ning neid mõõdikuid sildistama. Lisaks tuleb seadistada Prometheusi otspunkt, et Tempo saadaks mõõdikud Prometheusi. Joonisel 8 on kujutatud Tempo mõõdikute genereerimise ja Prometheusile edastamise konfiguratsioon. Kõigile genereeritud mõõdikutele lisatakse nimeruumi ja keskkonna atribuudid, et neid oleks võimalik seostada jälgedega. Joonis 9 kujutab "overrides" sektsiooni, kus lülitatakse sisse eelnevalt konfigureeritud mõõdikute genereerimise töötlejad ja määratakse jälgede hoiustamise ajaks 60 päeva.

5.5 Grafana paigaldamine ja seadistamine

Grafana vastutab süsteemis andmete visualiseerimise eest. Paigaldatakse Grafana vabavaraline versioon. Andmebaasina kasutatakse PostgreSql andmebaasi. Grafana seadistatakse vaikeväärtustega, välja arvatud andmebaasi ja asutuse autentimisteenuse ühendused. Dot-

```
metrics_generator:
  processor:
    span_metrics:
      dimensions:
        - deployment.environment
        - service.namespace
    service graphs:
      dimensions:
        - deployment.environment
        - service.namespace
  storage:
    path: /data2/tempo/generator/wal
    remote write:
      - url: prometheus write aadress
      send_exemplars: true
```

Joonis 8. *Tempo mõõdikute genereerimise konfiguratsioon*

```
overrides:
  defaults:
    metrics_generator:
      processors: [service-graphs, span-metrics]
    compaction:
      block_retention:1440h
```

Joonis 9. *Tempo overrides sektsiooni konfiguratsioon*

Net rakenduste mõõdikute ja jälgede kuvamiseks imporditakse Grafana kogukonna poolt loodud armatuurlaud: [2]

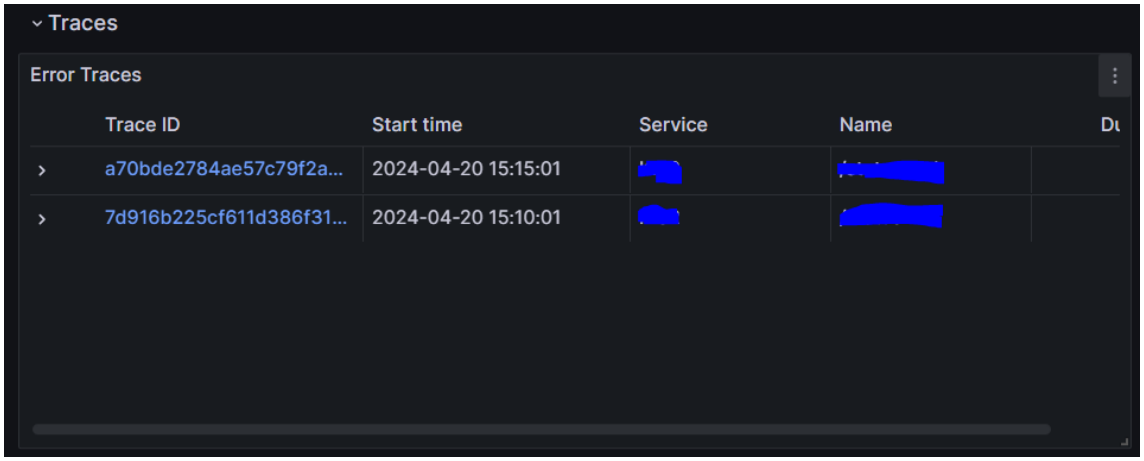
Joonis 10 kujutab loodud Grafana armatuurlaua protsessi mõõdikute graafikuid. Armatuurlaualt on võimalik valida ülevalt "job" nimekirjast instrumenteeritud DotNet rakendusi ning kuvada graafikutel andmeid rakenduse kohta. Antud joonisel on kuvatud protesssi mõõdikute graafikud

Joonised 11. ja 12. kujutavad armatuurlaua jälgede ja jälgedest genereeritud andmete osa. Vasakpoolne paneel kuvab jäljed, mille päringud sisaldasid viga. Parempoolne paneel kuvab rakenduse teenusegraafikut, kus on visualiseeritud rakendusega seotud komponendid ja süsteemid. Teenusegraafikud kuvavad komponentide vaheliste päringute vastamisaega ja päringute arvu sekundis.

[2]https://grafana.com/grafana/dashboards/17706-asp-net-otel-metrics/

Joonis 10. *Protsessi mõõdikutega Grafana paneelid*

Joonis 11. *Teenusegraafiku paneel*



Joonis 12. *Jälgede paneel*

5.6 DotNet rakenduste instrumenteerimine

Rakenduste andmete edastamiseks on vajalik rakendused instrumenteerida. DotNet IIS rakenduste instrumenteerimiseks nullkoodi lahendusega kasutatakse OpenTelemetry poolt pakutavat Powershelli moodulit. OpenTelemetry paigaldamiseks ja rakenduste instrumenteerimiseks kasutatakse järgnevaid samme:

1. Laetakse alla dotnet instrumenteerimise väljalase OpenTelemetry Githubi lehelt[3] ja

[3]https://github.com/open-telemetry/opentelemetry-dotnet-instrumentation/releases

liigutatakse võrgukettale, millele rakendusserver ligi saab:

2. Laetakse Powershell moodul ja paigaldatakse OpenTelemetry Core:

```
Import-Module ".\OpenTelemetry.DotNet.Auto.psm1"
Install-OpenTelemetryCore -LocalPath ".\OpenTelemetry.zip"
```

3. Seadistada keskkonnamuutuja Collectory otspunktiga, sest vaikeväärtus on kohalik aadress:

```
[System.Environment]::SetEnvironmentVariable('OTEL_EXPORTER_-
OTLP_ENDPOINT','https//opentelemetry-collector-endpoint:port')
```

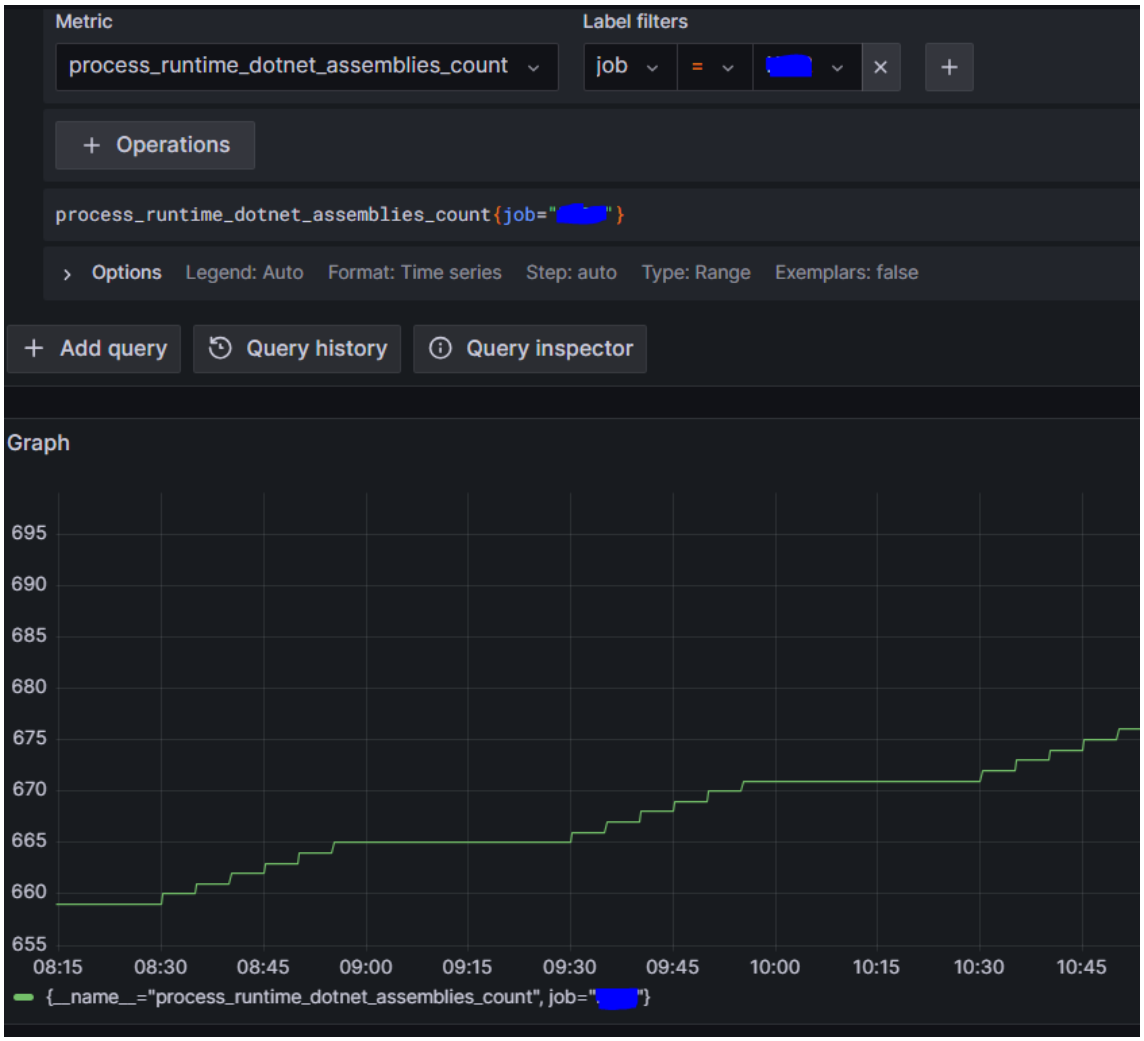
OpenTelemetry instrumenteerib kõik IIS rakendused serveris. Vanemate DotNet versioonidega võib tekkida teekidega konflikte ja seetõttu ei käivitu rakendused korrektselt. Autori ainuke leitud lahendus probleemile on määrata rakenduse *Apppool*i OpenTelemetry teekide asukoha keskkonnamuutujaks tühi sõne järgnevate käskudega:

```
{
$apphost = "system.applicationHost/applicationPools"

Add-WebConfigurationProperty -pspath 'MACHINE/WEBROOT/APPHOST ' `
-filter "$apphost/add[@name='pool']/environmentVariables" `
-name "." `
-value @{name='DOTNET\_ADDITIONAL\_DEPS';value=' '}}
```

4. Käivitatakse rakenduste instrumenteerimine käsuga:
`Register-OpenTelemetryForIIS`

Andmete korrektset edastamist saab kontrollida Grafana "Explore" vaates. Joonis 13 kujutab instrumenteeritud rakenduse mõõdikuseeriat Grafana "Explore vaates".



Joonis 13. *Instrumenteeritud rakenduse mõõdikud Grafanas*

5.7 Süsteemi tutvustus ja kasutuselevõtt

Töö autor tutvustas uut süsteemi tervele asutusele. Esitluses tutvustati süsteemi praeguseid võimekust, eesmärke mida soovitakse täita ja OpenTelemetry raamistikku. Hiljem toimus koosolek sisutiimide juhtidega, kus räägiti detailsemast plaanist süsteemi juurutamiseks. Süsteemi testkeskkonnaga liidestumise vastu tundsid huvi neli tiimi, kellest esimesteks valiti kahe meeskonna rakendused, mis põhinevad DotNet arhitektuuril.

Pärast esimeste rakenduste liidestamist süsteemiga tutvustati tiimidele Grafana kastuajaliidest ja näidati rakendustele loodud armatuurlaudu. Tiimidega vaadati ka üle andmed, mida OpenTelemetry rakenduste kohta kogub ja tutvustati võimalust rakenduste OpenTelemetry rakenduste koodi lisada, et oleks võimalik koguda lisa andmeid. Tiimide arendajatele ja halduritele loodi Grafana kasutajad, et nad saaksid ise luua armatuurlaudu ja vaadata andmeid. Arendajatele anti juhised, kuidas süsteemiga liidestada ka arenduses olevad rakendused.

Praeguses juurutamise etapis on süsteemis nelja DotNet ja kuue Kubernetese testkeskkonna rakenduse andmed. Arendustiimid planeerivad uurida võimalust OpenTelemetry SDKd kasutades rakendusi instrumenteerida, et saada rohkem andmeid rakenduse töö kohta.

6. Süsteemi analüüs

Selles peatükkis analüüsitakse 5. peatükis püstitatud süsteemi vastavalt 4 peatükis kirjeldatud metoodikale.

6.1 Rakenduste mõõdikute säilitamise, kogumise ja visualiseerimise tugi

Loodud süsteem suudab koguda ja säilitada rakenduste väljastatud mõõdikuid. Süsteemis on võimalik Grafana kasutajaliidese kaudu otsida ja kuvada andmeid Prometheusist ning neid andmeid kasutada visuaalsete graafikute moodustamiseks. Andmeid on võimalik filtreerida rakenduse nime või serveri järgi. Süsteem suudab kuvada hetkeseisu, kui ka varasemaid andmeid.

Süsteem võimaldab kuvada rakenduste mälukasutust ja protsessori kasutust, mis aitab diagnoosida jõudluse probleeme või mälulekkeid. Opentelemetry kogub ka andmeid erandite ja HTTP vastuse koodide kohta, mis võimaldab tuvastada probleeme, näiteks ümbersuunamis tsükleid. Kogutakse ka veebiserveri vastamise aega, mille kaudu on võimalik tuvastada ülekoormatust või jõudluse probleeme. Väljatoodud mõõdikud aitavad anda parema ülevaate rakenduse toimimisest ja kiirendada veatuvastusprotsessi. Rakenduste mõõdikute kuvamine, pakub asutuse halduritele ja arendajatele lisavõimekust, mida varasemad seiresüsteemid ei võimaldanud.

Varasemalt muutmälu puuduse korral pidid administraatorid tõrkuva serveriga ühenduma ja leidma rakenduse, mis kasutas ülemäära ressurssi, et tuvastada problemaatiline rakendus. Mõnel juhul, kui ressurssi oli kriitiliselt vähe ei olnud võimalik serveriga ühendust luua ja serverile tuli teha taaskäivitus. Pärast taaskäivitust ei olnud võimalik tuvastada milline komponent või rakendus tõrkus, mis tegi kogu veatuvastusprotsessi aeglasemaks. Uus süsteem võimaldab kuvada ajaloolisi andmeid iga rakenduse kohta eraldi, mis võimaldab tuvastada tõrkunud rakenduse ka pärast serveri taaskäivitamist.

6.2 Rakenduste logide säilitamise, kogumise ja visualiseerimise tugi

Süsteem suudab koguda ja säilitada rakenduste väljastatud logisid. Logisid on võimalik otsida ja kuvada logisid tabeli kujul. DotNet rakendustel logide saatmise funktsioon pole sisse lülitatud ning see funktsionaalsus on praegu alles eksperimentaalses etapis.

Grafanas on võimalik logisid liigendada JSON või logfmt kaudu. Logide vormingud võivad rakenduseti erineda, mis tähendab Grafana pool lisa seadistust. Eelnevalt kirjutasid paljud rakendused logisid kohaliku- või võrgukettale ning kettal asusid erinevates logfailides. Uue seiresüsteemi kaudu on võimalik logisid lugeda ja otsida ühtsest liidest Grafana kaudu, mis teeb administraatorite veatuvastus otsingu tõhusamaks. Logide põhjal saab genereerida mõõdikuid, mille kaudu on võimalik moodustada graafikuid. Logide mõõdikute kaudu on võimalik koostada graafikuid, kus kuvatakse logis esinevate vigade või hoiatuste arvu, mille kaudu saab ülevaate üldisest rakenduse tervisest.

6.3 Rakenduste jälgede säilitamise, kogumise ja visualiseerimise tugi

Süsteem suudab koguda ja säilitada rakenduste väljastatud jälgi. Süsteemis on võimalik otsida jälgi ning nende sisu ja andmeid visualiseerida, teksti või mõõdikute vormingus. Peatükis 5.4 seadistatud mõõdikute genereerimine võimaldab teenusgraafikute kuvamist ja rakenduste erinevate komponentide mõõdikute genereerimist jälgede põhjal. Teenusegraafikute abil on võimalik tuvastada vigu, mis mõjutab rakenduse tööd, teises süsteemides või komponentides.

Opentelemetry kogub DotNet rakenduste jälgi teekide kaudu. Jälgede kaudu on võimalik kuvada päringute teekonda süsteemis ning võimaldavad kuvada päringute kestvusaega, staatus koodi, teenuse nime ja päringu nime. Eelnevalt väljatoodud siltide kaudu saab jälgi filtreerida, et moodustada mõõdikuid või tabeleid. Jälgede abil saab kuvada rakenduse poolt teostavaid tegevusi, näiteks avalehe avamine või dokumendi kinnitamine ning mõõta toimingute kiirust. Päringute kiiruse mõõtmise võimalus aitab tuvastada probleeme ja leida pudelikaelu rakenduse toimimisel.

6.4 Turvalisus ja ligipääsu haldamine

Süsteemi on võimalik sisse logida kasutades asutuse OAuth 2 teenust, kuid teenusega ühendatud kontode õigusi pole võimalik muuta. Kõik Oauthiga sünkroniseeritud kontodele saab anda ühe kolmest Grafana rollist:

- **"Viewer"** roll annab õiguse vaadata kõiki armatuurlaudu. Vaikimisi on keelatud vaataja rollis olevatel kasutajatel kasutada "Explore" vaadet, kuid seda on võimalik konfiguratsioonis muuta. "Explore" vaade võimaldab kuvada jäljes talletatud päringuid, nende staatust ja kestvusaega. Juhul kui lubada vaataja õigustega kasutada "Explore" vaadet on kasutajal õigus pärida kõiki süsteemis olevaid andmeid.
- **"Editor"** roll annab õiguse muuta armatuurlaudu ja kaustu. Kasutaja saab ligi

"Explore" vaatele ning saab kasutada jälgede vaadet.

- **"Admin"** roll annab õiguse hallata kasutajaid ja tiime ning kõikide eelnevalt väljatoodud rollide õigused. Administraatori rollil on ka lubatud pluginate ja andmeallikate konfigureerimine.

Grafana rollid annavad väga laiad õigused ja nende kaudu pole võimalik piirata ligipääsu üldistele andmetele või funktsioonidele peale "Explore" vaate. Rollide vaatamis- või muutmisõigust on võimalik määrata armatuurlaua või kausta põhiselt, kuid ainult kolme rolliga on keeruline piirata ligipääsu rakenduste andmete põhiselt. Sünkroniseeritud kasutajaid pole võimalik lisada tiimidesse ning selle kaudu pole sammuti võimalik kasutajate õigusi piirata [11].

Administraatoritel on võimalik luua Grafana kontosid. Grafana kontode õigusi on võimalik piirata Grafana üleselt andes neile rolli või piirata õigusi eraldi iga armatuurlaua kohta. Lisaks sellele on võimalik Grafana kontosid lisada tiimidesse, mille kaudu saab piirata või anda kasutajatele õigusi. Vabavaralise versiooni puudulik integratsiooni võimalused on puudulikud ning teevad õiguste haldamise sünkroniseeritud kontode õiguste haldamise keeruliseks, kui mitte võimatuks. Grafana lokaalseid kontode õigusi on lihtsam hallata, kuid kontosid peab tegema käsitsi, mis teeb kasutajate haldamise keeruliseks ja aeganõudvaks.

6.5 Paindlikus ja universaalsus

DotNet rakenduste instrumenteerimine OpenTelemetry kaudu oli kiire ja lihtne protsess. DotNet rakenduste instrumenteerimisel asendab OpenTelemetry rakenduse teegid, teekidega, mis on OpenTelemetry poolt seadistatud andmeid koguma ja edastama. Üldised teegid, mida OpenTelemetry instrumenteerib on, näiteks SQL ühenduskihi teek ja HTTP päringute teek.

Süsteem suudab kuvada andmeid nii töö käigus instrumenteeritud DotNet , kui ka varem instrumenteeritud Kubernetese kobaras olevate Java rakenduste puhul. Vanemal DotNet Core versioonil olevaid rakendusi ei olnud võimalik instrumenteerida ning instrumenteerimine mõjutas ka nende käideldavust. Kubernetese kobaras asuvate Java rakendustel probleeme ei esinenud.

6.6 Avatud lähtekoodiga ja vabavaralised komponendid

Kõik seiresüsteemi komponendid on vabavaralised ja avatud lähtekoodiga. Opentelemetry Collector, Tempo, Loki, Prometheus on kõik vabavaralised lahendused ja kasutavad Apache 2.0 või AGPL litsentsi[12, 13, 14].

Prometheusi ja OpenTelemetry poolt kasutatav Apache 2.0 litsents lubab muuta lähtekoodi ja kasutada edasiarendatud versiooni ärilisteks eesmärkideks. Muudetud versiooni koodist ei pea tegema avalikuks. Grafana tooted on litsentseeritud AGPL tarkvaralitsentsiga, mis on piiravam kui Apache 2.0 litsents. AGPL lubab muuta lähtekoodi ja kasutada seda enda tarbeks, kui muudetud versiooni pakutakse, ärilisel eesmärgil, teenusena, siis peab lähtekoodi avalikustama. Juhul kui tulevikus soovitakse kohandada Grafana tooteid tuleb asutusel selle piiranguga arvestada.[15]

6.7 Hallatavus

Loodud süsteem on ligipääsetav ja muudetav asutuse administraatorite poolest. Süsteem ei sõltu teistest asutustest ja on muudetav vastavalt asutuse vajadustele. See võimaldab probleemidele kiiremini reageerida ja lahendus leida.

Süsteem ei asu avalikus pilvekeskkonnas ja andmed asuvad asutuse sisevõrgus. Andmete konfidentsiaalsus on oluline, kuna kogutavad logid võivad sisaldada tundlikuid andmeid ning neile võivad rakenduda isikuandmete kaitse seadused.

6.8 Käideldavus

Süsteem vastas käideldavuse nõudele ja kõige pikem katkestus, mis süsteemiga seoses toimus oli neli tundi, mis oli põhjustatud kettamahu täitumisest. Lühemaid katkesusi toimus sätete muutmisel, mis kestsid kuni 15 minutit.

Praeguses seisus süsteem ei ole kõrgkäideldav ning ühe komponendi rikke korral võib kogu süsteem olla kasutamatu. OpenTelemetry Collectori katkesuse ajal ei uuene andmed ja Grafanas on võimalik vaadata ainult mineviku andmeid. Grafana veebiliidese katkesuse ajal seireandmeid talletatakse, kuid kasutajatel pole võimalik andmeid vaadata. Loki, Prometheusi ja Tempo katkesuse korral ei ole saadaval vastava komponendi seireandmed.

6.9 Analüüsi tulemused

Püstitatud seiresüsteem lisab võimekuse arendajatel või halduritel näha rakenduse spetsifilisi mõõdikuid, logisid või jälgi. Mõõdikute, logide ja jälgede kogumine, säilitamine ja talletamine aitab leida pudelikaelu ja vigu süsteemis.

Grafana vabavaraline versioon on kasutajate haldamise ja autentimissüsteemide integreerimise osas väga piiratud. Grafanas on väga keeruline kasutajate ligipääsu andmetele piirata. Kasutajatele saab anda ühe kolmest rollist, mis annavad väga laiad õigused. "Explore" vaate kasutamise õigus on vajalik, et vaadata jälgede päringute analüütikat, kuid vaatest on võimalik pärida kõiki süsteemis olevaid andmeid. OAuth 2 kasutva autentimisteenuse integratsioon on puudulik. Kasutajaid on võimalik teenusest skünroniseerida, kuid antud kasutajatele saab anda õigusi ainult läbi rollide. Haldurid saavad teha Grafana kasutajaid, kuid neid tuleb teha käsitsi, mis ajakulukas ja läheb keerulisemaks kasutajate arvu kasvuga. Grafanas loodud kasutajate õigusi saab hallata tiimide või rollide kaudu.

Opentelemetry kaudu on võimalik instrumenteerida DotNet rakendusi, kuid esines probleeme vanemate rakendustega, mis ei töödanud korrektselt pärast instrumenteerimist. IISi rakenduste insturmenteerimisel ei ole võimalik sätetes määrata milliseid rakendusi instrumenteeritakse. Sätetes on võimalik määrata Opentelemetry teekide asukohaks tühi sõne, mille kaudu saab vältida problemaatilise rakenduse täieliku instrumenteerimist, kuid Opentelemetry loeb rakendust siiski instrumenteerituks ning logib vigu teenuse kohta.

Seiresüsteem põhineb vabavaralistel kompnentidel, mis aitavad asutusel kokku hoida litsentsi kulusid. Kõik kasutatud komponendid on avatud lähtekoodiga ja vajadusel on asutusel võimalik komponentides teha edasiarendusi.

Seiresüsteemi kasutatakse ainult asutuse siseselt, mis võimaldab muuta süsteemi vastavalt asutuse vajadustele. Süsteem pole sisseostetud teenus ja süsteemi saavad muuta asutuse administraatorid, mis võimaldab süsteemi muuta vastavalt vajadusele ja kiiremini.

Süsteem täitis esialgselt määratud käideldavuse nõuet. Süsteemi töös olid katkestused, millest üks oli planeerimata katkestus. Tulevikus kõrgema käideldavuse tagamiseks on vaja muuta süsteem kõrgkäideldavaks, mis nõuaks teise õla tekitamist.

Tabel 1 on koostatud vastavalt peatükk 4.2 kirjeldatud hindamismudelile ning annab ülevaate nõuetele vastavuse kohta.Süsteem vastas viiele nõudele täielikult ja kolmele osaliselt ehk protsentuaalselt täideti nõuded 81.25 % ulatuses. Analüüsis tuli välja, et kõige problemaatilisem nõue süsteemile on ligipääsu haldamine, mida tõkestab Grafana

vabavaralise versiooni piirangud. Seiresüsteem võimaldab halduritel ja arendajatel märgata ja tuvastada probleeme. Jälgede kaudu on võimalik saada hea ülevaade päringute kestvusest ja pudelikaeladest päringute teostamisel. Rakenduste logisid on võimalik süsteemis lugeda ja otsida ning koostada graafikuid üldisest logide seisust.

Tabel 1. Seiresüsteemi hinnang

Nõue	Hinne	Puudused
Rakenduste mõõdikute tugi	1	
Rakenduste logide tugi	0.5	DotNet rakenduste logide kogumine ei ole stabiilne ja toodangukeskkonna kõlbulik
Rakenduste Jälgede tugi	1	
Turvalisusja ligipääsu haldamine	0.5	OAuth2st sünkroniseeritud kasutajate haldamine problemaatiline, andmeallikatele ligipääsu haldamine pole võimalik
Paindlikus ja universaalsus	0.5	Vanemat versiooni kasutavate DotNet rakenduste instrumenteerimine ei tööta
Vabavaraline ja tasuta	1	
Asutuse poolt hallatav	1	
Käideldavus	1	
Kokku	6.5(81.25%)	

7. Edasised tegevused

Uus seiresüsteem vastas enamustele nõuetele, kuid analüüs ei käsitlenud nõudeid, millele peaks süsteem vastama toodangukeskkonnas. Eesolevas peatükis pakub autor välja neli soovitust süsteemi täiendamiseks:

1. **Kõrgkäideldavus**
   Töös kirjeldatud süsteem ei ole kõrgkäideldav ja kõik andmete talletamise ja visualiseerimise komponendid asuvad ühel serveril. Süsteemi kasutajatearvu kasvades ja toodangu keskkonda liikumisel, muutub kõrgkäideldavus oluliseks nõudeks. Süsteemi käideldavuse tagamiseks tuleks süsteemile luua vähemalt veel üks õlg, mis asuks geograafiliselt teises asukohas. Grafana Kasutajaliidese ja andmeid koguvate otspunktide ette saaks panna koormusjaoturi ja Grafana. Tempo, Loki ja OpenTelemetry Collector võimaldavad horisontaalset skaleerimist[16, 17, 18, 19]. Prometheusi skaleerimine võib kujuneda keeruliseks ülesandeks. Prometheus ei võimalda kobara tekitamist, mis tähendaks andmete saatmist mitmesse erinevasse Prometheusi intstantsi ning nende vahel olevad andmeid ei ole sünkroniseeritud[20]. Prometheusi asemel oleks võimalik kasutusele võtta Thanose või Mimiri, mis võimaldavad lihtsamat skaleerimist ja luua kobaraid.
2. **Andmete hoiustamine**
   Praeguse seisuga on andmed salvestatud kohalikule kettale. Ainukene andmete taastamise viis on virtuaalmasina hetktõmmise taastamine, mis juhul tekib andmekadu kõikidel süsteemi komponentidel. Lisaks sellele võib andmemahu kasvades võib süsteemis tekkida jõudlusprobleemid.
   Autor soovitab süsteemi andmed kolida objektihoidlasse. Objekti hoidlas asuvaid andmeid oleks võimalik versioneerida ja eraldada komponendi või andmetüübi põhiselt. Ainuke komponent, mis ei toeta objekti põhist salvestamist on Prometheus. Tuleks kaaluda Prometheusi väljavahetamist Thanose või Grafana Mimiri vastu, mis pakuvad sarnast funktsionaalsust ja võimaldavad kasutada objektihoidlat.
3. **Alarmeerimine**
   Püstitatud seiresüsteem hetkel ei saada teavitusi Grafana Alertmanager võimaldaks saata välja teavitusi emailina või Microsoft Teamsi sõnumina. Alarmide määramisel tuleks analüüsida ja välja valida mõõdikud, mis annavad kõige parema ülevaate rakenduse tervisest. Piirmäärade määramiseks saab kasutada juba süsteemis olevaid andmeid, et määrata normaal- ja tõrkeolukorrad. Alarmide seadistamisel tuleks arvestada teiste seiresüsteemide alarmidega, et vältida topelteavitusi ja alarmide

üleküllust.

4. **Turvatestimine**

   Kogutavates logides võivad kajastuda tundlikud andmed, mille tõttu oleks vajalik süsteemi turvatestimine. Testida tuleks andmete saatmise turvalisust, et otspunktide vahelist suhtlust ei oleks võimalik pealt kuulata. Uurida tuleks ka Loki turvalisust, kas Loki kaudu on võimalik ilma autentimata andmeid pärida või salvestatud andmeid otse kettalt lugeda.

8. Kokkuvõte

Käesoleva lõputöö eesmärk oli juurutada seiresüsteem, mis suudaks jälgida rakendusi, et muuta hõlbustada rakenduste vigade tuvastamist ja koguda kokku ühte süsteemi rakenduste seireandmed. Töös kirjeldatakse asutuse praeguse seirelahenduse probleeme, nagu seireandmete killustatus teiste seiresüsteemide vahel ja rakenduste seireandmetele keskendunud süsteemi puudumine, mis tekkisid pärast eelmise seiresüsteemi Dynatrace litsentsi lõpetamist.

Töö käigus anti ülevaade asutusest Registrite ja Infosüsteemide keskus. Lisaks tutvustati OpenTelemetry raamistikku; OpenTelemetry Collectorit, rakenduste instrumenteerimist ja andmetüüpe.

Lõputöös kirjeldati seiresüsteemi komponentide seadistamist ja paigaldust. Töös käsitleti Grafana, Loki, Tempo ja Prometheusi paigaldamist ja seadistamist. Lisaks kirjeldati ühe rakendusserveri DotNet rakenduste instrumenteerimist ja integreerimist seiresüsteemiga.

Süsteemi edasiseks täiendamiseks tõi autor välja neli teemat, mida tuleks veel arendada, et süsteem oleks kasutatav ka toodangukeskkonnas: teenuse kõrgkäideldavaks tegemine, mille puhul oleks vaja tekitada veel üks õlg ja uurida komponentide sobivust kõrgkäideldavuseks; andmete hoiustamine, mille käigus tuleks uurida võimalust salvestada andmeid objektihoidlasse; alarmeerimise seadistamine, mille puhul tuleb analüüsida süsteemis olevaid mõõdikuid ja valida piirväärtused, mille ületamise korral teavitusi saata; süsteemi turvatestimine, et saada ülevaade süsteemi praegusest turvalisusest ning nõrkuste võimalikest lahendustest.

Uue seiresüsteemi nõuetele vastavuse kontrollimiseks viidi töö käigus läbi eksperiment, mille käigus testiti ja analüüsiti loodud süsteemi vastavust asutuse nõuetele. Analüüsi tulemus näitas, et loodud seirelahendus vastas 81.25% ulatuses seatud nõuetele, ning aitab luua parema ülevaate rakenduste toimimisest, võimaldades kuvada rakenduste poolt genereeritud mõõdikuid, logisid ja jälgi. Süsteemi nõrkuseks olid kasutajate ligipääsu haldamine Grafanas ja OAuth2 integratsiooni piirangud vabavaralises versioonis. OpenTelemetry rakendamine vanematele DotNet rakendustele ei töötanud korralikult ja mõjuta rakenduse toimimist.

Kasutatud kirjandus

[1] Registrite ja Infosüsteemide Keskus. *„Asutusest*. [Kasutatud: 29-03-2024]. URL: https://www.rik.ee/et/asutusest.

[2] OpenTelemetry Authors. *„What is OpenTelemetry?* [Kasutatud: 29-03-2024]. URL: https://opentelemetry.io/docs/what-is-opentelemetry.

[3] OpenTelemetry Authors. *„Instrumentation*. [Kasutatud: 29-03-2024]. URL: https://opentelemetry.io/docs/concepts/instrumentation.

[4] OpenTelemetry Authors. *„Code-based*. [Kasutatud: 29-03-2024]. URL: https://opentelemetry.io/docs/concepts/instrumentation/code-based.

[5] OpenTelemetry Authors. *Zero-code*. [Kasutatud: 29-03-2024]. URL: https://opentelemetry.io/docs/concepts/instrumentation/zero-code.

[6] OpenTelemetry Authors. *Collector*. [Kasutatud: 29-03-2024]. URL: https://opentelemetry.io/docs/collector.

[7] Martin Thwaites. *Ask Miss O11y: Is the OpenTelemetry Collector useful?o*. [Kasutatud: 11-05-2024]. URL: https://www.honeycomb.io/blog/ask-miss-o11y-opentelemetry-collector.

[8] Justin Ellingwood. *An Introduction to Metrics, Monitoring, and Alerting*. [Kasutatud: 13-04-2024]. URL: https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting.

[9] Arfan Sharif. *Log Files Explained*. [Kasutatud: 22-04-2024]. URL: https://www.crowdstrike.com/cybersecurity-101/observability/log-file/.

[10] Grafana Labs. *What are traces?* [Kasutatud: 13-04-2024]. URL: https://grafana.com/docs/tempo/latest/traces/.

[11] Grafana Labs. *Configure authentication*. [Kasutatud: 22-04-2024]. URL: https://grafana.com/docs/grafana/latest/setup-grafana/configure-security/configure-authentication/.

[12] Grafana Labs. *Licensing*. [Kasutatud: 22-04-2024]. URL: https://grafana.com/licensing/.

[13] Prometheus Authors. *Frequently Asked Questions*. [Kasutatud: 22-04-2024]. URL: https://prometheus.io/docs/introduction/faq/.

[14] Jaana Dogan. *License*. [Kasutatud: 22-04-2024]. URL: https://github.com/open-telemetry/opentelemetry-collector/blob/main/LICENSE.

[15] TalTech IT Kolledži wiki. *Litsentsimise kitsaskohtade juhtumipõhine uurimine*. [Kasutatud: 07-05-2024]. URL: https://wiki.itcollege.ee/index.php/Litsentsimise_kitsaskohtade_juhtumip%C3%B5hine_uurimine.

[16] Scaling the Collector. *OpenTelemetry Authors*. [Kasutatud: 11-05-2024]. URL: https://opentelemetry.io/docs/collector/scaling/.

[17] Grafana Labs. *Litsentsimise kitsaskohtade juhtumipõhine uurimine*. [Kasutatud: 11-05-2024]. URL: https://grafana.com/oss/tempo/.

[18] Grafana Labs. *Set up Grafana for high availability*. [Kasutatud: 11-05-2024]. URL: https://grafana.com/docs/grafana/latest/setup-grafana/set-up-for-high-availability/.

[19] Grafana Labs. *About Grafana Tempo*. [Kasutatud: 11-05-2024]. URL: https://grafana.com/docs/loki/latest/get-started/deployment-modes/.

[20] Julius Volz. *High Availability for Prometheus and Alertmanager: An Overview*. [Kasutatud: 11-05-2024]. URL: https://promlabs.com/blog/2023/08/31/high-availability-for-prometheus-and-alertmanager-an-overview/.

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

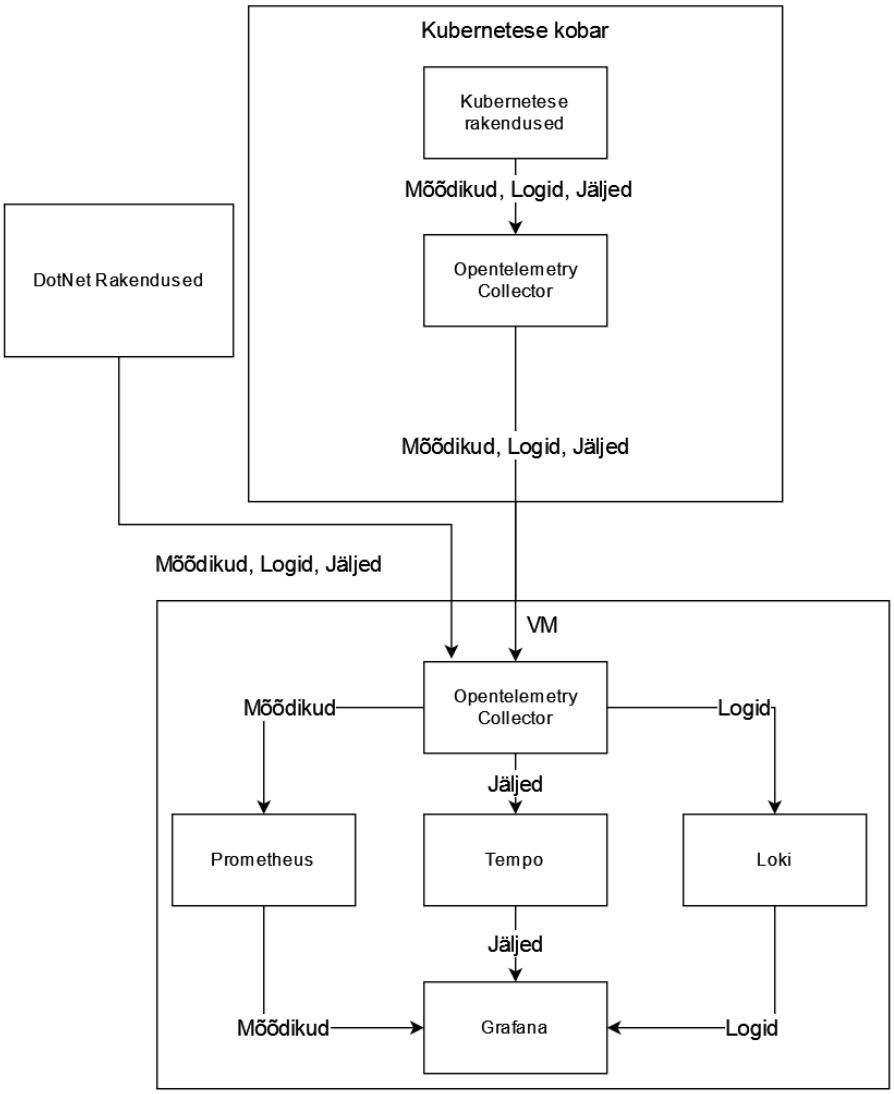
Mina Tauri Tuuling

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Open-telemetry standardi põhise Seiresüsteemi juurutamine Registrite ja Infosüsteemide Keskuses", mille juhendaja on Edmund Laugasson
   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.

14.05.2024

[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.

Lisa 2 - Süsteemi struktuur



Lisa 3 - Loki sätted

```
server:
  grpc_listen_port: loki port
common:
  instance addr: loki aadress
  path_prefix: /data2/loki
  storage:
    filesystem:
      chunks_directory: /data2/loki/chunks
      rules_directory: /data2/loki/rules
    replication_factor: 1
    ring:
      kvstore:
        store: inmemory
query_range:
  results_cache:
    cache:
      embedded cache:
        enabled: true
        max_size mb: 100
schema_config:
  configs:
    - from: 2025-10-24
    store: tsdb
    object_store: filesystem
    schema: v13
    index:
      prefix: index_
      period: 24h
```