TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Siim Düüna 192217IAPM

SOAP-I JA GRAPHQL-I VÕRDLUS X-TEE NÄITEL

Magistritöö

Juhendaja: Tarvo Treier
MsC

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: Siim Düüna

Kuupäev: 10.05.2021

Annotatsioon

Magistritöö eesmärgiks on SOAP-i ja GraphQL-i API-de võrdlus, mis on sooritatud X-tee keskkonna baasil. Tulemused aitavad kaasa võimalikule GraphQL-i kasutuselevõtule.

Jõudluste võrdlemiseks loodi X-tee turvaserveritesse tugi saata ja vastu võtta GraphQL sõnumeid. Lisaks sellele loodi konverterteenus, mis oskab automaatselt teenusepakkuja turvaserveris konverteerida GraphQL sõnumeid SOAP sõnumiteks ja seeläbi kasutada olemasolevaid X-tee SOAP teenuseid.

Näiteteenus arendati olemasoleva X-tee teenuse baasil ning kahes erinevas programmeerimiskeeles, et tõsta tulemuste usaldusväärsust. Mõlemas keeles võrreldi kolme teenust: SOAP, GraphQL ning konverterit kasutav teenus.

Eksperimentide tulemustest selgus, et GraphQL API ja SOAP API ajad on X-tee kontekstis väga lähedased. Konverterteenus oli oodatult SOAP teenusest aeglasem, kuid enamikel juhtudel oli vahe väike ja seega on konverterteenus enamike X-tee sõnumite juures kasutatav. Samuti on läbi konverterteenuse võimalik sõnumite mahtusid vähendada ning sooritada vajadusel mitu päringut korraga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 8 peatükki, 31 joonist, 3 tabelit.

Abstract

The purpose of this thesis is to compare the performance of SOAP and GraphQL API-s. Comparison is done in the context of the X-tee, which is a data exchange layer used in Estonia.

Evaluating the performances required to develop a new feature for the X-tee security server, which allows it to receive and send GraphQL messages. In addition, a converter is developed, which allows to use existing SOAP services with GraphQL messages. Converter is located inside service provider's security server in order to get maximum benefits.

Services that are used for testing are developed in two different programming languages, C# and Java, in order to make sure the results apply more generally. Both services have two endpoints: one for SOAP messages and another for GraphQL messages. Third endpoint is added automatically by the converter service.

Results show that GraphQL and SOAP are really close when measuring response times with small messages. With larger messages, GraphQL is faster. When measuring message sizes GraphQL messages are often more than two times smaller.

The implemented converter service is usable without too much additional time, given that messages are not too large. It also allows to reduce the message sizes and to query multiple resources with a single message.

The thesis is in Estonian and contains 48 pages of text, 8 chapters, 31 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
CA	*Certificate authority*, sertifitseerimisasutus
CSV	*Comma Separated Values*, tekstifail, kus erinevad väärtused on omavahel eraldatud komaga
Docker	Arvutiprogramm, mis teostab operatsioonisüsteemi tasemel virtualiseerimist ehk konteineriseerimist
GraphQL	Päringukeel API-de loomiseks
JSON	*JavaScript Object Notation*, JavaScriptil põhinev andmevahetusvorming
OCSP	*Online Certificate Status Protocol*, kehtivuskinnitusteenus
ORM	*Object–relational mapping*, objekt-relatsioonvastendus
REST	*Representational state transfer*, tarkvaraarhitektuuri laad
SOAP	*Simple Object Access Protocol*, lihtne objektipöördusprotokoll
TSA	*Time Stamping Authority*, ajatempliteenus
WCF	*Windows Communication Foundation*, Windowsi kommunikatsiooniteenuste raamistik
WSDL	*Web Services Description Language*, veebiteenuste kirjelduskeel
XML	*Extensible Markup Language*, laiendatav märgistuskeel
XML-RPC	*Extensible Markup Language Remote Procedure Call*, kaugprotseduurikutse protokoll XML-i baasil
X-Road	Keskkond turvalise ja tõestusväärse internetipõhise andmevahetuse jaoks
X-tee	Eestis kasutusel olev andmevahetuskiht, mis põhineb X-Road tehnoloogial

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Veebiteenuste andmete vahetamiseks on mitu konkureerivat võimalust. Tuntumad nendest on SOAP protokoll XML-i baasil ning REST arhitektuuristiil enamasti JSON-i baasil. Viimasel ajal on nende kõrvale juurde tulnud uuem tehnoloogia, GraphQL, mis vahetab andmeid samuti JSON-i kujul, kuid pakub REST-i ja SOAP-iga võrreldes mitmeid lisaväärtusi.

SOAP protokoll on väga vana ja loodi üle 20 aasta tagasi. Samuti on see keeruline kasutada ning on XML-i tõttu üsna mahukas. Siit tekib küsimus, ehk oleks aeg SOAP välja vahetada, mida paljudes kohtades on juba tehtud. Teatud valdkondades aga on SOAP veel laialt kasutusel. Üheks selliseks kohaks on Eestis kasutusel olev andmevahetusplatvorm X-tee, kuhu küll hiljuti lisati REST-i tugi, kuid siiski praktiliselt kõik teenused toimivad SOAP protokolliga.

Kuna praegu ei ole X-tee puhul veel REST-i laialt kasutusele võetud (2021 aasta aprilli seisuga kasutab REST-i alla 10 teenuse 2939-st teenusest [1], [2]), siis seda antud töös ei vaadata. Pigem uuritakse võimalust minna kohe SOAP-ilt üle otse GraphQL teenustele. Ülemineku õigustamiseks oleks vaja aga teada, kas ja millist kasu on GraphQL-iga võimalik üldse saavutada.

1.1 Ülesanne

Antud töö tegeleb SOAP-i ja GraphQL-i võrdlemisega X-tee baasil. Leitakse, kas GraphQL on X-tee keskkonnas teostatav ja kuidas see erineb SOAP-ist.

Uurimisküsimused on järgmised:

- Millised sammud on vajalikud GraphQL-i kasutamiseks X-tee keskkonnas?
- Kas ja kuidas on võimalik kasutada juba olemasolevaid teenuseid GraphQL-i abil?
- Kuidas erineb GraphQL-i jõudlus ja sõnumite suurus võrreldes SOAP-iga?
- Milliseid lisaväärtusi GraphQL pakub?

1.2 Eesmärk

Töö eesmärk on GraphQL-i eeliste ja puuduste analüüsimine võrreldes olemasoleva SOAP protokolliga, et uurida võimaliku ülemineku kasulikkust. Selle tõttu toimub võrdlus just X-tee keskkonna sees ja võimalikult lähedaselt reaalselt kasutatavatele teenustele.

Töö eesmärkide saavutamiseks tuleb täita järgmised alameesmärgid:

- GraphQL-i võimekuse lisamine turvaserverisse. Võimalus turvaserverisse lisada ja seal kasutada GraphQL teenuseid.
- Automaatse konverteri loomine olemasolevate SOAP teenuste kasutamiseks läbi GraphQL liidese.
- Lahenduse valideerimine ning eksperimentide läbiviimine, millest selguvad GraphQL-i võimalikud eelised ja puudused.

Kõigepealt tuleb GraphQL-i võimekus lisada X-tee turvaserverisse. See tähendab, et peab olema võimalus X-tee turvaserverisse lisada GraphQL teenuseid ja peab saama saata GraphQL sõnumeid läbi turvaserveri.

Antud magistritöö kirjutamise hetkel toetab X-tee ainult SOAP ja REST teenuseid. GraphQL teenuste kasutamiseks tuleb luua X-tee turvaserveritesse laiendus, mis võimaldaks edastada ka GraphQL sõnumeid. Selle saab teha sarnaselt olemasolevate REST teenustega, kuna mõlemad kasutavad andmevahetuseks JSON-i.

Seejärel luuakse automaatne konverter, mille kaudu saaks läbi GraphQL-i kasutada olemasolevaid SOAP teenuseid. Selle jaoks konverteeritakse SOAP teenuste kirjeldused GraphQL skeemiks. Samuti on vaja teisendada GraphQL päringud SOAP päringuteks ja vastused seejärel teistpidi tagasi.

1.3 Ülesehitus

Töö algab X-tee ning hetkel kasutusel olevate SOAP ja REST sõnumite tutvustamisega. Seejärel näidatakse, kuidas võimalikud GraphQL sõnumid neist erinevad. Näidete abil tuuakse välja ka lisaväärtusi, mida GraphQL-i kasutuselevõtuga saavutaks: peamiselt sõnumite arvu ja andmemahu kokkuhoidu.

Teiseks selgitatakse tehtud lahendust ehk GraphQL-i implementatsiooni tehnilisemat poolt. Siia kuuluvad lisatud funktsionaalsuse arhitektuur ning tehtud koodimuudatused, mis on

vajalikud eksperimentide läbiviimiseks.

Järgmiseks tutvustatakse loodud konverterit ja selgitatakse selle tööpõhimõtet ning eesmärki. Konverter on eelkõige mõeldud kasutamiseks hetkel olemasolevate X-tee SOAP teenustega ning kujutab endas lisavõimalust, et aidata kaasa GraphQL-i kasutuselevõtule.

Edasises osas valideeritakse tehtud konverter ning eksperimenteeritakse olemasolevate X-tee teenuste peal. Võrreldakse nii jõudlusi kui ka sõnumite mahtusid. Tutvustatakse testkeskkonda ning kirjeldatakse täpsemalt läbiviidavaid katseid.

Töö lõpp sisaldab endas tulemuste analüüsi ning nende pealt tehtavaid järeldusi. Samuti arutletakse võimalike tulevikulahenduste osas, et paika panna edasine suund.

2. Taust

Antud peatükis tutvustatakse X-tee olemust ning ülesehitust. Tuuakse mõned näitesõnumid nii SOAP-i, REST-i kui ka GraphQL-iga ning seletatakse nende erinevusi. Samuti arutletakse võimalike tulemuste üle varasemate tööde baasil. Kõik need teadmised on vajalikud ülejäänud osa lihtsamaks arusaamiseks.

2.1 X-tee

X-tee[1] on keskkond, mis võimaldab turvalist internetipõhist andmevahetust. Üks osapool pakub välja oma teenuse ja teised X-teega liitunud liikmed saavad seda kasutada. Eestis on olnud X-tee kasutusel alates 2001. aastast ning seda kasutavad peamiselt riigiinfosüsteemid omavahelisel suhtlusel. Kui ise midagi näiteks eesti.ee lehelt pärida, siis jõuavad samuti andmed kohale läbi X-tee. Pikemat ülevaadet X-teest on võimalik lugeda Riigi Infosüsteemi Ameti (RIA) blogist[2]. [3] [4]

Praegusel hetkel toimub X-tee arendus Eesti ja Soome koostöös, MTÜ Nordic Institute for Interoperability Solutions (NIIS) kaudu. Lähtekood on avalikult saadaval GitHub-is[3] ja selle projekti ametlikuks nimeks on X-Road. X-teeks nimetatakse selle Eestis kasutusel olevat versiooni. Antud töös kasutatakse edaspidi neid mõlemaid mõisteid nende õiges tähenduses. [4]

2.1.1 Ajalugu

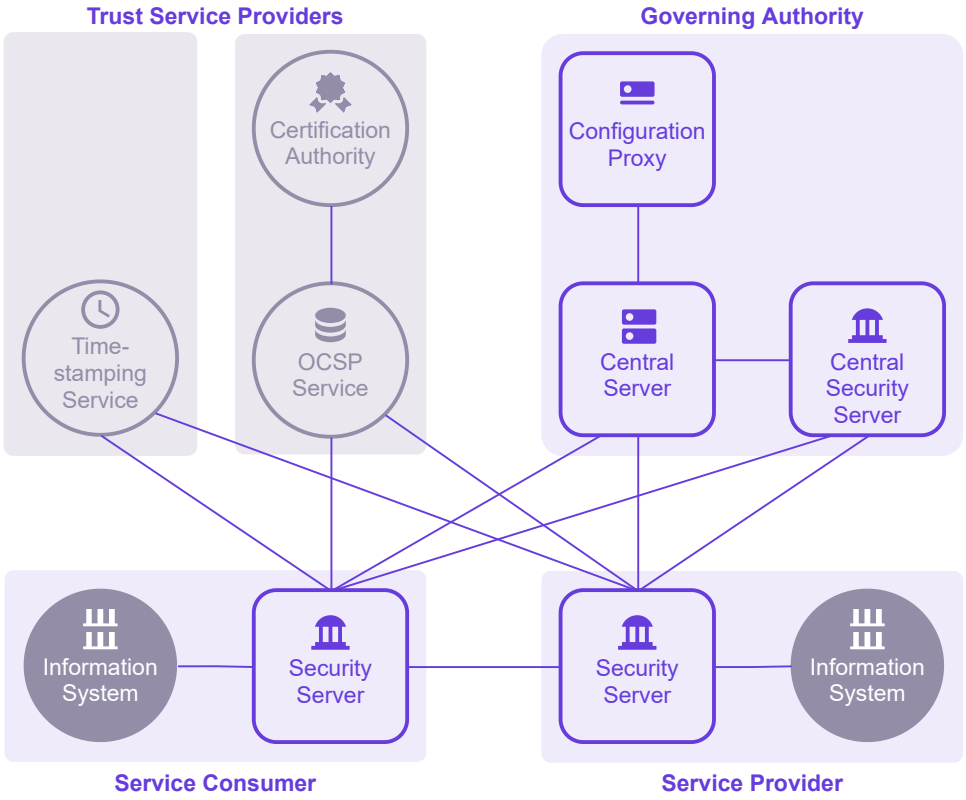
Esimene X-tee versioon tuli välja 2001. aastal ning põhiarendajateks olid Tanel Tammet, Hannu Krosing ja Vello Kadarpik. Andmevahetus X-teel toimus siis üle XML-RPC (*Extensible Markup Language Remote Procedure Call*), kuna arvati, et SOAP protokoll pole veel piisavalt küps. Hiljem (alates 2002) vahetati see SOAP-i vastu välja, mis on ka praeguseni X-tees kasutuses. Iseenesest ei leiutatud X-tee jaoks ühtegi uut tehnoloogiat, vaid pandi mitu olemasolevat tehnoloogiat omavahel koos töötama. Sellist tehnoloogiat riigiasutuste vahel ei oldud kusagil mujal veel kasutatud. [5], [6]

[1] `https://www.ria.ee/et/riigi-infosusteem/andmevahetuskiht-x-tee.html`
[2] `https://blog.ria.ee/sissejuhatus-x-teesse-osa-1/`
[3] `https://github.com/nordic-institute/X-Road`

Sel ajal oli SOAP väga populaarne tehnoloogia, mis oli kasutusel väga paljudel veebilehtedel andmevahetuseks. Praeguseks on selle juba kuskil 10 aastat tagasi välja vahetanud REST, millel on mitmeid eeliseid. Peamised probleemid SOAP-iga, mida REST parandab, on liigne keerukus ja XML-i kasutamisega kaasnev lisakulu nii ajas kui ka mahtudes. Tänapäeval on vähe uusi projekte, mis valivad REST-i asemel SOAP-i. [7]

2.1.2 Arhitektuur

X-Road koosneb paljudest eraldiseisvatest serveritest, mis omavahel suhtlevad. Üldine arhitektuurijoonis on toodud Joonisel 1.



Joonis 1. X-Roadi üldine arhitektuur.[1]

Tähtsaim osa X-Roadist on keskus, kus asub keskserver (*Central Server*), mis koordineerib X-Roadi toimimist. Sellega võtavad otse ühendust kõik turvaserverid (*Security Server*), et alla laadida vajalikke konfiguratsioonifaile. [8]

Turvaserverid on kõige enam esinevad komponendid ja need on vajalikud igale teenusepakkujale ning igale kasutajale. Turvaserverid suhtlevad omavahel otse ehk tegelikult ei vaja keskserveri abi. See tähendab, et nii kaua kuni konfiguratsioonifailid on kehtivad, jääb X-Road toimima ka keskserveri katkestuse korral. [8]

[1] `https://github.com/nordic-institute/X-Road/blob/develop/doc/Architecture/img/arc-g_deployment_view_of_x_road.svg`

Lisaks eelnevale on vaja X-Roadi toimimiseks ajatempliteenust (*Timestamping Service*), sertifitseerimisasutust (*Certification Authority*) ning OCSP (*Online Certificate Status Protocol*) teenust sertifikaatide valideerimiseks. [8]

2.2 SOAP

SOAP[1] (*Simple Object Access Protocol*) ehk lihtne objektipöördusprotokoll on protokoll XML-i kujul teenuste andmete vahetamiseks. X-Roadi enamus teenuseid baseeruvad SOAP-il. Konkreetset informatsiooni saab X-Roadi sõnumiprotokollist[2].

2.2.1 SOAP-i näidissõnumid

SOAP sõnum koosneb üldiselt päisest (*header*) ja kehast (*body*). Järgnevalt on Joonisel 2 toodud näidissõnum X-Roadi teenuse väljakutsest.

Päises olevad väljad on antud juhul kõik vajalikud X-Roadi turvaserverite jaoks. Nende alusel saab otsustada, millist teenust selle sõnumiga soovitakse välja kutsuda. Samuti on seal olemas informatsioon väljakutsuja ja teenusepakkuja tuvastamiseks. Kui turvaserver otsustab, et kliendil on õigus teenust kasutada, saadetakse sõnum edasi serverisse, mis teenust osutab.

XML-i keha on mõeldud teenuse jaoks ja X-Roadiga seotud informatsiooni ei oma. Antud näide kutsub välja Ravimiregistri pakendite teenuse, mille kaudu on võimalik saada pakendite koode.

Vastussõnum on toodud Joonisel 3. Sõnumipäis on muidu sama, kuid lisandunud on üks väli. Turvaserver lisab automaatselt juurde päringu räsi (*Request Hash*) välja, mille alusel seotakse päring ja vastus omavahel kokku. Vastuse keha sisaldab seda informatsiooni, mida teenus ette näeb, ehk antud juhul pakendite koodide nimekirja. Lisaks on selles vastuses ka päringu parameetrid.

2.2.2 Teenuste kirjeldus WSDL-i kujul

WSDL (*Web Services Description Language*) on XML-i baasil keel teenuste kirjeldamiseks. See määrab ära teenuses olevad operatsioonid, sisendid, väljundid ja kõik muu vajaliku teenuste kasutamiseks. [9]

[1] `https://www.w3.org/TR/2007/REC-soap12-part1-20070427/`

[2] `https://www.x-tee.ee/docs/live/xroad/pr-mess_x-road_message_protocol.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xro="http://x-road.eu/xsd/xroad.xsd"
      xmlns:iden="http://x-road.eu/xsd/identifiers"
      xmlns:kood="http://ravimiregister.x-road.eu">
   <soapenv:Header>
      <xro:protocolVersion>4.0</xro:protocolVersion>
      <xro:issue/>
      <xro:userId>EE37305120234</xro:userId>
      <xro:id>ID123456</xro:id>
      <xro:service iden:objectType="SERVICE">
         <iden:xRoadInstance>ee-test</iden:xRoadInstance>
         <iden:memberClass>GOV</iden:memberClass>
         <iden:memberCode>70003477</iden:memberCode>
         <iden:subsystemCode>ravimiregister</iden:subsystemCode>
         <iden:serviceCode>pakendid</iden:serviceCode>
         <iden:serviceVersion>v1</iden:serviceVersion>
      </xro:service>
      <xro:client iden:objectType="SUBSYSTEM">
         <iden:xRoadInstance>ee-test</iden:xRoadInstance>
         <iden:memberClass>GOV</iden:memberClass>
         <iden:memberCode>74000091</iden:memberCode>
         <iden:subsystemCode>rets</iden:subsystemCode>
      </xro:client>
   </soapenv:Header>
   <soapenv:Body>
      <kood:pakendid>
         <keha>
            <muudatusedAlates>
               <kuupaev>2020-12-01</kuupaev>
            </muudatusedAlates>
            <ravimiLiik>inim</ravimiLiik>
            <retseptinoueLiik>kxik</retseptinoueLiik>
         </keha>
      </kood:pakendid>
   </soapenv:Body>
</soapenv:Envelope>
```

Joonis 2. Ravimiregistri pakendite teenuse XML-i näidissõnum.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xro="http://x-road.eu/xsd/xroad.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soapenv:Header>
      <xro:protocolVersion>4.0</xro:protocolVersion>
      <xro:issue/>
      <xro:userId>EE37305120234</xro:userId>
      <xro:id>ID123456</xro:id>
      <xro:service iden:objectType="SERVICE">
         <iden:xRoadInstance>ee-test</iden:xRoadInstance>
         <iden:memberClass>GOV</iden:memberClass>
         <iden:memberCode>70003477</iden:memberCode>
         <iden:subsystemCode>ravimiregister</iden:subsystemCode>
         <iden:serviceCode>pakendid</iden:serviceCode>
         <iden:serviceVersion>v1</iden:serviceVersion>
      </xro:service>
      <xro:client iden:objectType="SUBSYSTEM">
         <iden:xRoadInstance>ee-test</iden:xRoadInstance>
         <iden:memberClass>GOV</iden:memberClass>
         <iden:memberCode>74000091</iden:memberCode>
         <iden:subsystemCode>rets</iden:subsystemCode>
      </xro:client>
      <xro:requestHash
         algorithmId="http://www.w3.org/2001/04/xmlenc#sha512">
            cusoCBD5lDmioM491/P56KVG/RtggonFCMx2SYy3FjbW
            CgsC4+W7H+/4RkDEWdLH0I4u8bhAcyoDwiECfGUYgg==
        </xro:requestHash>
   </soapenv:Header>
   <soap:Body>
      <pakendidResponse xmlns="http://ravimiregister.x-road.eu">
         <paring xmlns="">
            <muudatusedAlates>
               <kuupaev>2020-12-01</kuupaev>
            </muudatusedAlates>
            <ravimiLiik>inim</ravimiLiik>
            <retseptinoueLiik>kxik</retseptinoueLiik>
         </paring>
         <keha xmlns="">
            <Item>1001136</Item>
            <Item>1005455</Item>
            <Item>1007514</Item>
         </keha>
      </pakendidResponse>
   </soap:Body>
</soap:Envelope>
```

Joonis 3. Ravimiregistri pakendite teenuse XML-i näidisvastus.

X-Roadi turvaserver võimaldab SOAP teenuseid lisada ainult WSDL-i abil. Selle alusel oskab turvaserver automaatselt valmis teha kõik seal kirjeldatud teenused ning võimaldab neid eraldi konfigureerida. X-Road määrab WSDL dokumentidele mõned lisapiirangud[1], mida need täitma peavad. [10]

2.3 REST

REST (*Representational State Transfer*) on tarkvaraarhitektuuri laad, mis sätestab konkreetsed reeglid teenuste loomiseks. Seega ei ole REST-i puhul tegemist mitte protokolliga, vaid stiilikogumikuga.

X-Roadi tugi REST-ile on tulnud üsna hiljuti, suuremas osas 2019. aasta jooksul. Sarnaselt SOAP liidesele on olemas X-Road protokoll[2] ka REST teenuste jaoks. [11]

REST sõnumites on varasemalt XML-i päises asuv X-Road info jagatud HTML-i päise ning URL-i vahel ära. Väljakutsuja alamsüsteemi identifikaator läheb HTML-i päisesse „X-Road-Client“ ning teenuse identifikaator sisaldub URL-is. Sõnumite sisu jääb seega puhtalt teenuste informatsiooni jaoks, kusjuures tegelikult on lubatud mis iganes tüüpi päringu keha. Kuigi enamasti kasutatakse JSON-i, võib tegelikult saata kehana ka XML sõnumit. [11]

JSON-i kasutamine XML-i asemel omab aga mitmeid eeliseid. JSON sõnumid on kompaktsemad, kasutavad vähem mälu ning neid saab kiiremini töödelda [12].

SOAP-i puhul kasutasid kõik päringud HTTP POST meetodit, kuna keha oli alati vajalik. REST-i puhul on võimalus kasutada erinevaid meetodeid. Andmete küsimiseks tuleks kasutada GET meetodit, andmete muutmiseks PUT ja POST meetodeid ning kustutamiseks DELETE meetodit.

Erinevalt SOAP-ist puudub REST-il ametlik viis teenuseid kirjeldada. Olemas on vaid teatud stiilijuhised, kuidas peaks tegema ja mida ei tohi teha. Nii on X-Roadis võimalik lisada REST teenuseid manuaalselt operatsioonide kaupa. Sellegipoolest on olemas ka võimalusi REST teenuseid kirjeldada sarnaselt WSDL-ile. Tuntum neist on OpenAPI 3[3], mille tugi on X-Roadil samuti olemas.

[1] https://www.x-tee.ee/docs/live/xroad/pr-mess_x-road_message_protocol.html#32-describing-services-with-wsdl

[2] https://www.x-tee.ee/docs/live/xroad/pr-rest_x-road_message_protocol_for_rest.html

[3] http://spec.openapis.org/oas/v3.1.0

2.3.1 REST-i näidissõnumid

Joonisel 2 toodud näidissõnum oli ainult pakendite koodide kuvamiseks ja ei muutnud andmeid. Seega REST-i puhul oleks otstarbekas päringuks kasutada GET meetodit ja anda sisendparameetrid päringustringi (*query parameter*) kaudu. Seega tuleks saata GET päring, mis on antud Joonisel 4.

```
/GOV/70003477/ravimiregister/pakendid
    ?kuupaev=2020-12-01&ravimiliik=inim&retseptinoueLiik=kxik
```

Joonis 4. Ravimiregistri pakendite näidispäringu päringustring.

Vastuseks tuleb JSON-i kujul sisu, mis on toodud Joonisel 5. Võrreldes seda eelnevalt toodud XML-i kujul vastusega (Joonis 3), on see tunduvalt väiksem ja lihtsamini loetavam. Siiski sisaldab see vastus vähem informatsiooni, kuna X-Roadi jaoks vajalikud päise elemendid on viidud mujale.

```
{
  "pakendid": {
    "keha": {
      "Item": [
          "1001136",
          "1005455",
          "1007514"
        ]
    },
    "paring": {
      "muudatusedAlates": {
        "kuupaev": "2020-12-01"
      },
      "ravimiLiik": "inim",
      "retseptinoueLiik": "kxik"
    }
  }
}
```

Joonis 5. Ravimiregistri pakendite teenuse JSON-i näidisvastus.

2.4 GraphQL

GraphQL[1] on avatud lähtekoodiga andmete pärimise ja muutmise keel. Algselt on see arendatud Facebooki poolt 2012. aastast alates. Avalikult tuli välja 2015. aastal. Võib öelda, et tegemist on uuema alternatiiviga REST-ile ning SOAP-ile. [13]

GraphQL-il on SOAP-i ning REST-iga võrreldes mitmeid eeliseid. On võimalik päringus määrata, milliseid välju vastuses tahetakse ja seeläbi märgatavalt vähendada vastuste mahtusid. Samuti on võimalik teha paralleelselt mitu päringut koos ja saada vastused

[1] https://graphql.org/

korraga tagasi. Arendusele endale aitab kaasa GraphQL-i kohustuslik skeem, mis sisaldab endas API poolt pakutavaid väljasid.

2.4.1 GraphQL näidissõnumid

GraphQL-is on olemas kolme liiki sõnumeid: andmepäringud (*query*), mutatsioonid (*mutation*) ja tellimused (*subscription*). Kuna viimase jaoks ei ole X-Roadil tuge olemas, siis tellimusi siin töös ei vaadata.

Andmepäringud on mõeldud andmete küsimiseks ilma muudatusi tegemata. Need on sisuliselt siis REST-is olevad GET päringud. Mutatsioonid on mõeldud andmete lisamiseks, muutmiseks ning kustutamiseks. REST-is vastavad neile PUT, POST ja DELETE. SOAP-i puhul sellisel moel sõnumeid ei jagata.

GraphQL näidissõnum on toodud Joonisel 6. Sõnum ütleb, et tahame saada informatsiooni pakendite kohta. Päringus on olemas sisendparameetrid ja väljade loetelu, mida vastuses tahetakse. Need viimased küll suurendavad päringu mahtu, kuid võimaldavad vastuse mahtu kokku hoida eeldusel, et tihti ei ole kõiki välju vaja.

```
query {
  pakendid(keha: {
    ravimiLiik: inim,
    retseptinoueLiik: retseptiravim
    muudatusedAlates: {kuupaev: "2020-01-01"}
  }) {
    keha {
      Item
    }
    paring {
      muudatusedAlates {
        kuupaev
      }
      ravimiLiik
      retseptinoueLiik
    }
  }
}
```

Joonis 6. Ravimiregistri pakendite teenuse GraphQL näidissõnum.

Üle HTTP andmete saatmisel üldjuhul kodeeritakse GraphQL päringud JSON-i kujule. Vastused on juba niikuinii JSON-i kujul ja seega üpris samasugused, kui need olid REST API-s. Näidisvastus on toodud Joonisel 7. REST-i vastusest eristab seda ainult välimine „data“ väli, mille sisse GraphQL andmed pakib. See on vajalik, kuna samal tasemel saab vajadusel tagastada veainformatsiooni.

```
{
  "data": {
    "pakendid": {
      "keha": {
        "Item": [
          "1001136",
          "1005455",
          "1007514"
        ]
      },
      "paring": {
        "muudatusedAlates": {
          "kuupaev": "2020-12-01"
        },
        "ravimiLiik": "inim",
        "retseptinoueLiik": "kxik"
      }
    }
  }
}
```


Joonis 7. Ravimiregistri pakendite teenuse GraphQL näidisvastus.

2.5 Jõudlus

Üks punkt, mis jõudluse koha pealt REST-i ja GraphQL-i kasuks räägib, on JSON-i kasutamine XML-i asemel. See aga ei tähenda veel, et terviklikuna oleksid REST ja GraphQL kiiremad. Erinevate API-de jõudlusi on uurinud J. Sayago Heredia, E. Flores-García ja A. R. Solano. Uuringu käigus tehti kolmes erinevas keeles (C#, PHP, Java) valmis kolm erinevat CRUD (*create, read, update, and delete*) API-t (SOAP, REST, GraphQL) ja uuriti nende vastuste aegu koormuse all. Igas programmeerimiskeeles olid tulemused sarnased: kõige kiirem oli GraphQL ja kõige aeglasem SOAP. Leiti veel, et suure koormuse juures tekkis SOAP API-s vigu, mida teistega ei esinenud. Siit lähtuvalt võiks eeldada, et GraphQL oleks jõudluse poole pealt ka X-tee keskkonnas parem või vähemalt mitte halvem kui SOAP. [14]

Oluline müügiargument GraphQL-i kasuks on ainult vajaminevate väljade küsimine. Sellest saadavat võitu on uuritud artiklis „Migrating to GraphQL: A Practical Assessment“. Leiti, et GraphQL-i kasutuselevõtt vähendas päringute vastuste suurusi rohkem kui 99%, kuna uuritud API-del sai koguni 94% väljadest pärimata jätta. Nii hea tulemus saavutati, kuna artiklis kirjeldatud API-del oli palju väljasid, millest läks ainult väheseid vaja. Kindlasti annaks sõnumite suurusi vähendada ka olemasoleva REST API disaini muutes, aga siis jälle suureneks API operatsioonide arv. Lisaks päringute vastuste suurusele on olulised veel päringute enda suurused, mida artiklis aga võrreldud ei ole. Eelduste kohaselt peaksid päringuid ise olema GraphQL API puhul suuremad võrreldes REST API-ga, kuna päringutes sisaldub väljade loetelu, mida vastuseks tahetakse. [15]

3. GraphQL-i toe lisamine X-tee turvaserverisse

Töö esimene osa sisaldab endas X-Road turvaserverisse laienduse kirjutamist, et võimaldada X-Roadi GraphQL teenuste lisamist. See on vajalik töö põhieesmärgi ehk SOAP ja GraphQL teenuste võrdlemiseks. Turvaserver on ainukene komponent, mida tuleb X-Roadi keskkonnas muuta, kuna teenuste sõnumid liiguvad otse kahe turvaserveri vahel.

3.1 Keskkonna püstitus

X-Roadi kood on avalikult kättesaadav GitHubi repositooriumis[1] ja seda jagatakse MIT litsentsi all. Lühidalt tähendab see seda, et kõigil on lubatud koodi kasutada ja muuta tingimusel, et vastav litsents koodi kohta jääb alles. X-Roadi arenduse jaoks on veel teinegi repositoorium[2], kus on olemas arenduseks vajaminev informatsioon.

Arenduskeskkond püstitati Ubuntu 20.04 masinas suuresti X-Road repositooriumi enda juhendi[3] järgi. Klooniti Giti repositoorium ning kasutati Ansible[4] tarkvara keskkonna püstitamiseks. Samasse masinasse pandi tööle keskserver ning üks turvaserver kasutades LXD[5] konteinereid. Samuti sai selle abil püstitatud sertifitseerimisasutus koos ajatemplining OCSP teenustega.

3.2 Loodud lahendus

GraphQL-i teenustes liiguvad andmed JSON-i kujul. Seega oli võimalik teha lahendus sarnaselt olemasolevale REST lahendusele. Samuti sai ära kasutada olemasolevaid andmebaasitabeleid, kus ühtegi muudatust tegema ei pidanud. Põhilised erinevused tekkisid GraphQL teenuse kirjelduse parsimisega ning teenuste lisamisega.

GraphQL teenuse kirjelduse parsimiseks kasutati GraphQL Java[6] teeki. Sealt saadavast kirjeldusest otsiti välja kõik andmepäringud ning mutatsioonid. Need mõlemad lisati turvaserveri andmebaasi koos teenuste nimedega. Seega tekkis kahte liiki GraphQL teenuseid:

[1] https://github.com/nordic-institute/X-Road
[2] https://github.com/nordic-institute/X-Road-development
[3] https://github.com/nordic-institute/X-Road/tree/develop/ansible
[4] https://github.com/ansible/ansible
[5] https://linuxcontainers.org/lxd/introduction/
[6] https://www.graphql-java.com/

andmepäringud (*query*) ja mutatsioonid (*mutation*). Võrdluseks, REST teenustes oli liike rohkem: GET, PUT, POST ja DELETE.

3.3 Ligipääsuõigused

X-Roadi turvaserver teostab muuhulgas õiguste kontrollimist. SOAP teenuste puhul on võimalus seal ära määrata iga kliendi kohta, milliseid teenuseid on tal õigus välja kutsuda. Samuti saab määrata ka ühe teenuse korral, milliseid meetodeid sellel teenusel saab kasutada. REST teenuste puhul on lahendus sarnane: õigusi saab määrata iga väljakutsutava URL-i ja HTTP meetodi paari jaoks.

Õiguste kontrollimine lisati ka GraphQL teenustele, et hiljem läbiviidavad katsed saaksid toimuda võimalikult ausas keskkonnas. SOAP-i puhul piisas õiguste kontrollimiseks ainult sõnumi päise lugemisest, kust oli vaja leida väljakutsutava teenuse kood. REST-i puhul pole vaja sõnumit isegi mitte lugeda, kuna vastav kood on kättesaadav teenuse aadressist. GraphQL puhul tuleb õiguste kontrollimiseks lugeda kogu päringu keha. Eesmärk on leida sealt seest väljakutsutava teenuse nimi. Samuti ei saa päringu sisu lugemist varem lõpetada, kuna üks päring võib endas sisaldada mitme teenuse väljakutset.

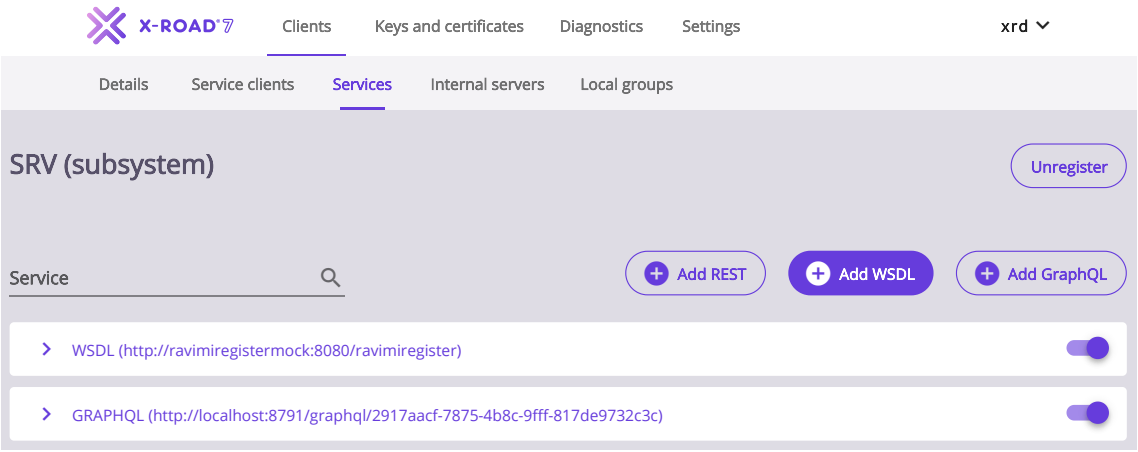
Töö käigus loodud GraphQL teenuste õiguste kontrollimine toimub kokkuvõtvalt järgnevalt. Kõigepealt loetakse saadetava päringu keha ja leitakse sealt kõikide väljakutsutavate teenuste nimed kasutades GraphQL Java teeki. Järgnevalt kontrollitakse, sarnaselt SOAP ja REST teenuste lahendusele, andmebaasist, kas on luba vastavaid teenuseid välja kutsuda. Kuna päring võib sisaldada mitut teenuse väljakutset, siis tehakse kontroll iga teenuse nimega. Kui päring sisaldab endas mõnda keelatud teenust, siis jääb terve päring täitmata.

Alternatiivina oleks võimalus mitme teenuse korral päringust eemaldada keelatud teenused ja ülejäänud teenused lasta välja kutsuda. Otsustati selle variandi vastu, kuna see poleks tegelikult mingit kasu andnud. Võib öelda, et selle variandiga oleks saanud pooliku päringu kiiremini kätte, kuid on kaheldav, kas klientidel on pooliku päringuga midagi peale hakata ning poolikud muudatused võivad tekitada ka teatud vigu. Samuti on „kõik või mitte midagi“ lahenduse abil võimalik alati uus päring saata, milles keelatud väljad on eemaldatud.

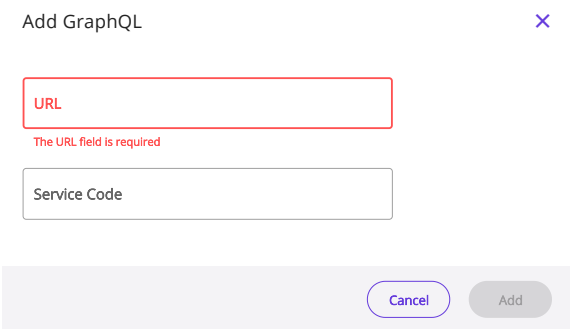
3.4 Kasutajaliides

X-Road turvaserveril on haldamise jaoks olemas kasutajaliides, mille kaudu saab määrata ligipääsuõigusi ning lisada uusi teenuseid. Ekraanitõmmis sellest on toodud Joonisel 8.

Töö käigus on sinna lisatud nuppude „Add REST“ ja „Add WSDL“ kõrvale nupp „Add GraphQL“. Sellele vajutades on võimalik sisestada teenuse aadress ja teenuse kood (Joonis 9). Lisamise aken on lahendatud väga sarnaselt olemasolevatele lahendustele.

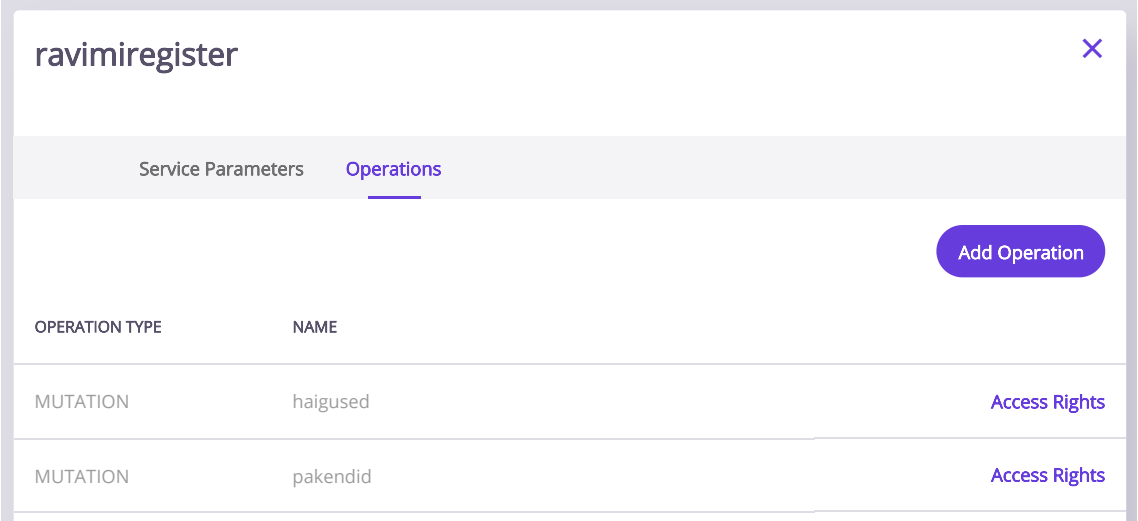


Joonis 8. X-Road turvaserveri teenuste vaade.



Joonis 9. X-Road turvaserveris GraphQL teenuse lisamise aken.

Lisaks teenuste lisamisele tuli muuta õiguste andmise alamlehte. Suures osas sai taaskasutada REST-i jaoks olemasolevat funktsionaalsust, aga selguse huvides oli vaja lihtsalt muuta see vaade GraphQL teenustele vastavaks. Vastav lehekülg on nähtaval Joonisel 10. Varasemalt kuvatavale tekstile „HTTP Request Method“ vastab sisuliselt GraphQL teenuste puhul „Operation type“ ning tekstile „Endpoint“ vastab „Operation“.



Joonis 10. X-Road turvaserveri õiguste vaade GraphQL teenuste jaoks.

4. Olemasolevate X-tee teenuste automaatne teisendamine GraphQL-i

Töö käigus loodi automaatne konverter, mis oskab WSDL-i alusel genereerida GraphQL teenuse. Konverter aitab teha ülemineku SOAP teenustelt GraphQL teenustele sujuvamaks ja mugavamaks. Kuigi üleminek ei ole otseselt magistritöö osa, on konverter vajalik jõudluste võrdlemiseks.

Antud töö käigus loodud konverter ei pruugi olla kõige efektiivsem ja kindlasti saab seda optimeerida. Sellest hoolimata on loodud konverter piisavalt töökindel ja võimaldab hinnata, milliseid tulemusi on vähemalt võimalik saavutada.

Konverteri testimiseks on kõige parem kasutada X-tees kasutusel olevaid teenuseid. WSDL failid on kõik vabalt kättesaadavad X-tee kataloogist[1]. Antud peatükis on enamasti näitena kasutatud Ravimiregistri teenuste WSDL faili[2].

4.1 Olemasolevad lahendused

Enne konverteri kirjutama hakkamist uuris ning katsetas töö autor olemasolevaid lahendusi, mida tegelikult oli väga vähe. Lõpuks otsustati siiski hakata ise lahendust realiseerima. Järgnevalt selgitatakse leitud lahenduste puudusi.

Java programmeerimiskeeles leiti ainult üks projekt, mis tegeles WSDL-i konverteerimisega GraphQL-iks. Selleks oli Emil Crumhorn´i poolt kirjutatud „WSDL to GraphQL Generator“[3]. Katsetamiseks võeti X-tees kasutusel olev Ravimiregistri WSDL fail, millega antud projekt juba ei toiminud. Ka projekti enda kirjelduses oli, et tulemus ei ole täielik ning võib tekkida vajadus GraphQL teenuse kirjeldust ümber järjestada. Nende probleemide tõttu ei ole antud lahendus kasutatav ning otsustati mitte proovida seda parandama hakata.

[1] https://x-tee.ee/catalogue/EE

[2] https://x-tee.ee/catalogue-data/EE/EE/GOV/70003477/ravimiregister/6.wsdl

[3] https://github.com/crumhorn/wsdl2graphql

Veel on olemas veebiteenus APIMATIC[1], mis pakub WSDL-i konverteerimist GraphQL teenuse kirjelduseks. See sai küll eelnevalt mainitud Ravimiregistri WSDL failiga hakkama, kuid firma on pigem orienteeritud tasulise lahenduse pakkumisele. Kuna lähtekood ei ole saadaval, siis pole seda võimalik käesolevas töös kuidagi kasutada.

Kõige paremat lahendust pakkus „node-soap-graphql“[2]. Lahendus saadi lihtsamapoolse WSDL-iga küll tööle, kuid Ravimiregistri WSDL-iga lahendus ikkagi ei toiminud. Lisaks on antud projekt kirjutatud TypeScript keeles ja ei ole seepärast otseselt kasutatav. Polnud ka võimalik seda otse Java keelde ümber kirjutada, kuna see sõltus teekidest, mida sellisel kujul Java jaoks ei leitud.

4.2 Lahenduse piirangud

X-Road määrab WSDL dokumentidele täiendavaid piiranguid, millega ka konverter peab arvestama. Õnneks teevad need piirangud konverteri tööd pigem lihtsamaks ning aitavad seega koodi kokku hoida. Kõik piirangud on olemas X-Road SOAP sõnumiprotokollis[3]. Järgnevalt on toodud välja neist olulisemad.

WSDL dokumendid peavad olema „document/literal“ tüüpi ning võivad sisaldada lisaväljasid X-Road teenuste kirjeldamiseks. Näiteks saab ära märkida X-Road teenuse versiooninumbri ning kirjelduse.

Sisend- ja väljundsõnumite kirjelduses on nõutud, et seal võib olla ainult ühe osa (*part*) kirjeldus. Teisisõnu on igal sõnumil kõige kõrgemal tasandil ainult üks element. See element ise võib juba loomulikult sisaldada endas mitmeid teisi elemente.

Veel on toodud mõned reeglid nimetamise kohta. Päringut sisaldava elemendi nimi peab olema täpselt sama, mis operatsiooni nimi. Vastust sisaldava elemendi nimi peab olema operatsiooni nimi, mille lõppu on lisatud „Response“. Nii on lihtsam vajalikke elemente üles leida.

X-Road võimaldab muuhulgas saata manuseid (*attachments*) kasutades mitmeosalisi (*multipart*) sõnumeid. See on võimalik nii SOAP-i kui ka REST-i puhul ja manused ei sisaldu otseselt XML-is ega JSON-is, vaid saadetakse sõnumis nende järel. GraphQL-i jaoks on olemas sarnane lahendus[4], mis pole küll ametlik, kuid on siiski implementeeritud paljudes

[1] https://www.apimatic.io/

[2] https://github.com/sevenclev/node-soap-graphql

[3] https://www.x-tee.ee/docs/live/xroad/pr-mess_x-road_message_protocol.html#32-describing-services-with-wsdl

[4] https://github.com/jaydenseric/graphql-multipart-request-spec

GraphQL implementatsioonides [16]. Konverteris kasutatud GraphQL Java võimaldab samuti seda lahendust kasutada [16].

Eelnevalt mainitud lahenduse korral oleksid manused nii SOAP kui ka GraphQL sõnumite puhul samal kujul ja seega pole põhjust arvata, et nende jõudlus oleks kuidagi erinev. Seetõttu on manused magistritööst välja jäetud ning ka konverterteenus nendega ei tegele.

4.3 Kasutatud teegid

WSDL-i parsimiseks sai valitud teek Membrane SOA[1], kuna see pakkus tulemusi kõige mugavamal kujul. Järgnevalt analüüsitakse leitud alternatiive ning nende puudusi.

Esimene alternatiividest oli teek „soap-ws“[2]. Selgus aga, et teeki on viimati uuendatud 7 aastat tagasi ning on seega maha jäetud. Teegi installeerimisel tekkis kohe probleem, et Maven-i repositoorium, kus kohast seda peaks saama alla laadida, pole enam aktiivne. Sellel põhjusel otsustati antud teeki mitte kasutada.

Apache WODEN[3] on samuti teek WSDL-i parsimiseks. Teegil on olemas tugi WSDL 2.0-i jaoks, aga praegu ei ole tuge WSDL-i vanemate versioonide jaoks. Kuna X-Road kasutab just vanemaid versioone, siis pole antud teegi kasutamine võimalik.

EasyWSDL[4] võimaldab parsida nii WSDL 2.0 kui ka 1.1 versioone. Võrreldes Membrane SOA teegiga puudub sellel võimalus SOAP sõnumite automaatseks genereerimiseks. Kuna nende genereerimine on konverteri jaoks oluline, siis otsustati kasutada siiski Membrane SOA teeki.

Peale WSDL-i parsimise peab konverter oskama pakkuda välja GraphQL-i teenust. Selle jaoks kasutatakse teeki GraphQL Java[5], mida kasutati ka turvaserveris õiguste kontrollimiseks. Muid alternatiive Java keeles ei leitud.

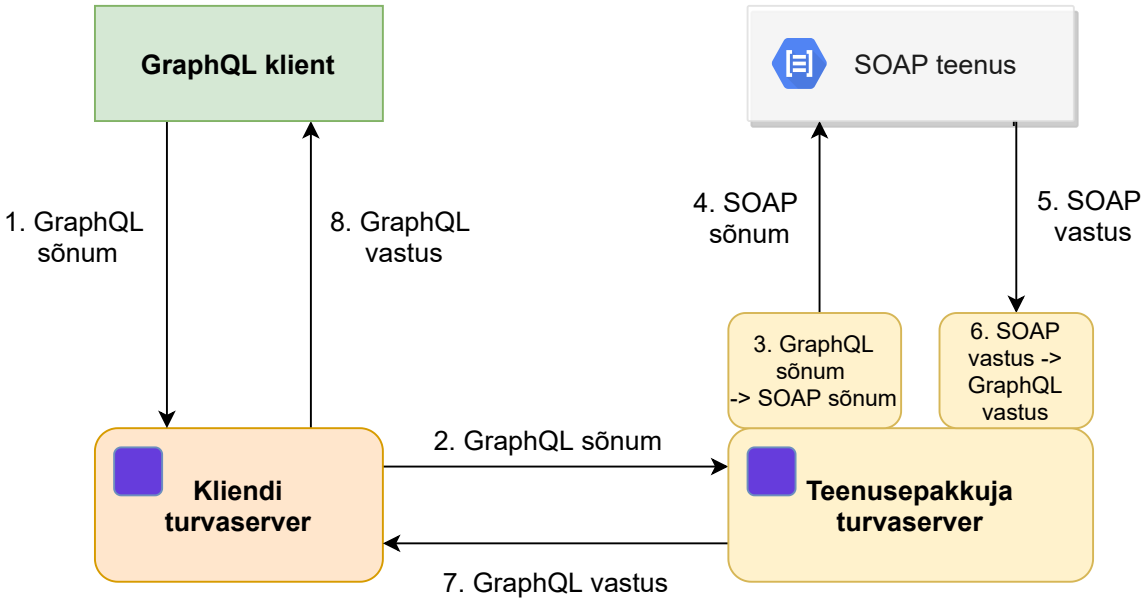
SOAP vastuste parsimiseks on kasutusel XStream [6], mis aitab lugeda XML-ist väljasid. Saadud vastuse väljad teisendatakse GraphQL Java jaoks sobivale kujule ning sealt tekib sobilik GraphQL vastus JSON kujul.

[1] https://www.membrane-soa.org/
[2] https://github.com/reficio/soap-ws
[3] https://ws.apache.org/woden/index.html
[4] http://easywsdl.ow2.org/
[5] https://www.graphql-java.com/
[6] https://x-stream.github.io/

4.4 Konverteri ülesehitus

Lahendus hakkab paiknema turvaserveris, täpsemalt teenusepakkuja poolses turvaserveris. Selle eeliseks on, et üle võrgu liiguvad just GraphQL sõnumid, mis eelduse kohaselt on väiksemad ning seega võimaldab konverter võrguliikluse efektiivsemat kasutust.

Konverteri tööpõhimõte on toodud Joonisel 11. GraphQL sõnum saadetakse läbi kliendi turvaserveri teenusepakkuja turvaserverisse. Seal kontrollitakse kliendi õigusi teenuseid kasutada. Õiguste olemasolul konverteeritakse GraphQL sõnum ringi kas üheks või mitmeks SOAP sõnumiks, mis siis paralleelselt SOAP teenuse abil täidetakse. Saadud üks või mitu vastust teisendatakse jälle ringi GraphQL vastuseks, mis saadetakse kliendini tagasi.



Joonis 11. Loodud konverterteenuse kasutamise joonis.

Sisemiselt koosneb konverter mitmest osast. Esmalt on vaja ühekordselt konverteerida etteantud WSDL teenuse kirjeldus GraphQL teenuseks. See leiab aset teenuse lisamisel. Kui tahetakse lisada uut SOAP teenust, siis tegelikult luuakse selle kõrvale samaaegselt GraphQL teenus, mis sisemiselt kasutab seda SOAP teenust. Olemasolevate SOAP teenuste konverteerimine võiks aset leida turvaserveri versiooniuuenduse ajal.

Teine osa on korduv tegevus iga sõnumi jaoks. Selle jaoks kõigepealt parsitakse GraphQL sõnumit ja leitakse sealt soovitud teenused. Selle osaga tegeleb GraphQL Java teek. Edasi otsitakse sõnumist välja sisendparameetrid, mille alusel koostatakse valmis SOAP sõnum. Sõnumi koostamisega tegeleb Membrane SOA teek, millele tuleb ette anda vastavad sisendparameetrid. Järgnevalt sooritatakse üks või mitu SOAP päringut ja sealt saadavad vastuse väljad pannakse sobivasse kohta GraphQL vastuses.

4.5 Probleemid

Konverteri ehitamise juures tekkis mõningaid probleeme, millest suuremaid järgnevalt analüüsitakse. Selgitatakse probleemi olemust ning loodud lahendust.

4.5.1 Päringute jagamine

GraphQL sõnumid on jaotatud andmepäringuteks (*queries*) ja mutatsioonideks (*mutations*). See jaotus tuleb teenuse tegijal endal teha ning soovitav on järgida reeglid, et andmepäringud on andmete küsimiseks ilma muudatusi tegemata. Mutatsioonid on siis vastavalt andmete lisamiseks, muutmiseks ja kustutamiseks. SOAP-i puhul aga sellist erisust ei tehta, mis tähendab, et pole võimalik kuidagi otsustada, kumba gruppi mingi SOAP päring kuulub.

Üks võimalik lahendus oleks vaadata SOAP sõnumi nime ning selle järgi otsustada. Kui nime alguses on nt „get...“ ja midagi veel, siis oleks tegemist andmepäringuga ning muudel juhtudel mutatsiooniga. Siiski tundub see lahendus väga ebakindel, kuna kuskil pole kohustust sõnumeid niimoodi nimetada. Varasemalt toodud sõnuminäide Joonisel 2, sisaldas endas päringut nimega „pakendid“, mille järgi on raske automaatselt otsustada, kas tegemist on muudatusega või mitte. Mainitud põhjuste tõttu antud lahendust kasutusele ei võetud.

Alternatiivne lahendus, mille kasuks otsustati, on lihtsalt kõik SOAP sõnumid määrata mutatsioonideks. Samasugune lahendus on kasutusel eelnevalt mainitud alternatiivides nagu APIMATIC ja „node-soap-graphql“. Põhiline miinus sellel lahendusel on, et puudub informatsioon, kas päringuga toimuvad andmemuudatused, kuid seda informatsiooni ei olnud olemas ka SOAP teenuses. Ehk siis võrreldes SOAP-iga ei kaotata otseselt midagi.

4.5.2 Elementide nimetamine

Vastavalt GraphQL spetsifikatsioonile[1] peavad kõik nimetused teenuse kirjelduses vastama Regex-ile "`/[_A-Za-z][_0-9A-Za-z]*/`". See tähendab, et nimetused peavad algama kas tähe või alakriipsuga ning ülejäänud märgid peavad olema kas tähed, alakriipsud või numbrid. Samuti on määratud, et lisatavad nimetused ei tohi alata kahe alakriipsuga, kuna seda kasutab GraphQL-i sisemine tüübimääramise süsteem. Mainitud nimetamise reeglid kehtivad muuseas ka loendite (*enum*) liikmetele. [17]

[1] `http://spec.graphql.org/June2018/#sec-Names`

WSDL ja XML on piirangute mõttes vabam ja seal on lubatud rohkem erinevaid märke nagu koolon, sidekriips ja punkt [18]. See tähendab, et WSDL võib sisaldada endas nimetusi, mida GraphQL ei toeta. Esimene ja kõige lihtsam lahendus oleks need keelatud märgid lihtsalt ära eemaldada. Seda on lihtne teha, kui on vaja konverteerida SOAP sõnum GraphQL sõnumiks. Vastupidi tehes peab aga need märgid tühjast kohast juurde tekitama ehk on eelnevalt vaja kõik teisendused kuhugi kirja panna. Teine probleem selle lahenduse puhul on, et märkide ärakaotamine võib tekitada olukorra, kus kaks erinevat WSDL-is sisalduvat nime erinevad teineteisest ainult keelatud märkide poolest ja sellisel puhul oleks mitu nime GraphQL-is ühesugused.

Alternatiivne lahendus oleks asendada keelatud märgid mingisuguse lubatud märkide jadaga. Nii saab teha teisendust mõlemas suunas, ilma et peaks tehtavaid teisendusi kuhugi üles märkima. Negatiivse poole pealt muudab see lahendus päringutes väljade nimesid segasemaks, kuid selle saab ära lahendada klientrakenduse sees. Konverterteenuses on valitud keelatud märkide asendamiseks jada „_x_“, kus x on keelatud märgi ASCII kood kuueteistkümnendsüsteemis. Näiteks väärtus „Maaletoomis-JaKasutusloagaPakend“ asendatakse väärtusega „Maaletoomis_2D_JaKasutusloagaPakend“.

5. Eksperimentide läbiviimine

Antud peatükk sisaldab endas läbiviidavaid eksperimente, mille alusel võrreldakse omavahel SOAP ning GraphQL teenuseid. Nende alusel saadavad tulemused ongi antud magistritöö peamine eesmärk. Kõigepealt seletatakse eksperimentite läbiviimise korda ja püstitatakse hüpoteesid. Tutvustatakse näiterakendusi ning loodud teenuseid.

5.1 Läbiviimise kord

Eksperimentite läbiviimiseks otsitakse olemasolev X-tee teenus, mis võiks olla kõige rohkem kasutatavate teenuste hulgas. Ei ole mõtet valida teenust, mida väga vähe kasutatakse. Teenus püütakse realiseerida vastavalt WSDL-ile ning muu internetis leitava informatsiooni alusel.

Näiteteenusest kirjutatakse kaks erinevat varianti kahes erinevas programmeerimiskeeles. Nii saab eemaldada võimaluse, et ühes keeles on lihtsalt kas SOAP või GraphQL teenused paremini realiseeritud. Siinkohal oleks hüpoteesiks, et mõlemas keeles oleksid SOAP-i ja GraphQL-i omavahelised aegade suhted lähedased.

Loomulikult tekib kohe võimalus omavahel SOAP-i ning GraphQL-i võrrelda. See aga ei ole töö eesmärk ning jäetakse välja. Osalt ka sellepärast, et on raske kindlustada, et mõlemad lahendused on implementeeritud parimal võimalikul viisil.

Kuna GraphQL teostab automaatselt päringute valideerimist, siis võrdsuse huvides on SOAP teenustes samuti päringute valideerimine sisse pandud. Vastuste valideerimist ei teostata, kuna eeldatavasti moodustavad näiterakendused alati korrektsed vastused.

Näiteteenusest valitakse välja mingi hulk operatsioone, mille jaoks kirjutatakse valmis nii SOAP kui ka GraphQL teenus. Mõlemad teenused kasutavad täpselt sama andmebaasi ning suhtlevad sellega kasutades repositooriumi mustrit (*Repository pattern*[1]). See tähendab, et mõlemad teenused saavad oma tulemuse kasutades täpselt sama meetodi väljakutset. Nii kindlustatakse mõlema lahenduse võrdsus.

[1] `https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design`

5.2 Läbiviidavad eksperimendid

Iga valitud operatsiooniga sooritatakse mõlemas keeles kolm päringut:

1. SOAP teenuse päring;
2. GraphQL päring läbi konverterteenuse;
3. GraphQL teenuse päring.

Kõiki päringuid sooritatakse mitmeid kordi ning võrdluseks kasutatakse päringuaegade mediaani, mis kujutab endast keskmise päringu jaoks kuluvat aega. Alternatiiv oleks kasutada aegade keskmist, kuid seda võib liialt mõjutada isegi üksainus väga aeglane päring, mis mediaani väärtust ei mõjuta. Antud töös eeldatakse, et mõni üksik suure ajaga päring on juhuslik testsüsteemi kõikumine ning ei ole kuidagi põhjustatud kasutatavast SOAP või GraphQL raamistikust. Sellel põhjusel tuleks äärmusi ignoreerida.

Lisaks eelnevatele sooritatakse mõlema GraphQL teenusega päringuid, millega tuuakse korraga ära mitu ressurssi. Ehk siis koostatakse GraphQL päring, mis sisuliselt vastab mitmele SOAP päringule. Nii on võimalik võrrelda selleks kuluvat aega ja SOAP päringute aegade summat. Siinkohal tekib küsimus, et miks kasutatakse summat, kui SOAP päringud saaks teha paralleelselt. Paralleelsete päringute efektiivsus sõltub palju koormusest ja seega on raske seda mõõta. Testsüsteemis madala koormuse korral võtaks kaks paralleelset päringut eeldatavasti sama aja kui aeglasem päring. Seega GraphQL-i mitme korraga pärimisel võitu nii ei saaks. Nende testide korral eeldatakse, et kliendi poole pealt ei ole paralleelsus võimalik või on liiga keerukas realiseerida.

Eksperimentides sisaldub veel üks tühi päring teenustpakkuva turvaserveri vastu. Nii on võimalik mõõta võrguliikluseks kuluvat aega.

5.3 Hüpoteesid

Katsete läbiviimiseks püstitatakse järgnevad hüpoteesid:

1. Kõikide väljade pärimise korral ei ole GraphQL teenus SOAP teenusest halvem.
2. Läbi konverteri tehtavad päringud ei ole eriti palju aeglasemad kui SOAP päringud.
3. GraphQL teenuses mitme päringu korraga tegemisel on võimalik saada oluline ajaline kokkuhoid.

Esimene hüpotees sisaldab endas kõikide väljade pärimist. Seega ei kasutata ära GraphQL-i

tugevust väljade valimise mõttes. Kuna mõlema teenuse sõnumid sisaldavad seega täpselt sama informatsiooni, pole põhjust arvata, et GraphQL võiks kiirem olla.

Läbi konverteri tehtavad päringud on kindlasti aeglasemad kui SOAP päringud, kuna konverter sisemiselt kasutabki SOAP teenust. Seega hüpoteesiks on püstitatud, et konverter ei tohi olla nii palju aeglasem, et see hakkaks teenuse kasutamist segama. Mõistlikku piiri, kui palju aeglasem võib konverter olla, on keeruline määrata, kuna see sõltub tehtavate päringute kogustest.

SOAP teenuseid on võimalik välja kutsuda vaid ühekaupa. GraphQL võimaldab teha aga mitu päringut korraga. Nii võiks saada kokkuhoiu isegi konverterit kasutades eeldusel, et SOAP päringud on tehtud järjestikku. Kokkuhoiu suurus sõltub sellest, mitu ressurssi korraga tahetakse ja on tugevalt seotud kõige aeglasema ressursi ajaga. Lähedaste aegade juures võiks kahe operatsiooni korraga pärimine olla vähemalt 1,5 korda kiirem ning kolme operatsiooni korral vähemalt 2 korda kiirem.

5.4 Teenuse kirjeldus

Testimise jaoks on aluseks võetud Ravimiregistri infosüsteem[1]. Teenuse kirjeldus WSDL kujul on saadaval X-tee kataloogist[2]. Ravimiregistri infosüsteem on kasutatavuselt päris populaarne. Uurides X-tee avaandmeid[3], selgub et nädala (10.04.2021 - 16.04.2021) lõikes on tehtud päevas keskmiselt 32 086 päringut.

Originaalteenuse arhitektuur on saadaval Riigi Infosüsteemi Haldussüsteemi (RIHA) leheküljelt[4]. Andmebaasina on kasutusel MS SQL Server 2008. Sellest lähtuvalt oleks testimissüsteemis hea kasutada sarnast lahendust. On võimalik kasutada täpselt sama versiooni, aga see on aegunud ning sellel puudub ametlik Dockeri konteiner. Seetõttu võeti testimissüsteemis kasutusele MS SQL Server 2019[5] ehk praegusel hetkel kõige uuem versioon.

Andmebaasiga suhtlemiseks on kasutusel ORM (*Object–relational mapping*) raamistikud, kuna need vähendavad arenduseks kuluvat aega. Samuti ei ole andmebaasiga suhtluse kiirus eriti oluline, kuna on lahendatud nii SOAP-i kui ka GraphQL-i jaoks identselt.

[1] `https://www.riha.ee/Infos%C3%BCsteemid/Vaata/176e3cbf-46c4-6024-d6a7-e26b7a4365d1`

[2] `https://x-tee.ee/catalogue-data/EE/EE/GOV/70003477/ravimiregister/6.wsdl`

[3] `https://logs.x-tee.ee/EE/gui/`

[4] `https://www.riha.ee/api/v1/systems/ravimiregister/files/5bf25dbd-a344-b688-a50e-91ff12fe4af5`

[5] `https://www.microsoft.com/en-us/sql-server/sql-server-2019`

Valitud programmeerimiskeelteks osutusid C# ning Java, kuna need on kuulunud juba viimased 10 aastat viie kõige populaarsema programmeerimiskeele hulka [19]. Lisaks on C# keelel väga hea tugi MS SQL Serveriga, kuna mõlemad on arendatud Microsofti poolt. Java valikule aitas kaasa veel see, et X-Road on ise selles kirjutatud.

5.5 ASP .NET 5 rakenduse kirjeldus

Esimene rakendus on kirjutatud ASP .NET 5 raamistiku abil C# programmeerimiskeeles. Teenuse jaoks kasutatavad mudelid on genereeritud WSDL faili põhjal kasutades dotnet-svcutil tööriista[1]. Andmebaasiga suhtlemiseks on kasutusel Entity Framework Core raamistik[2].

SOAP teenuse arendamiseks on kasutusel SoapCore[3] raamistik, mis on implementeeritud Microsofti blogipostituse[4] alusel. Varasemalt oli SOAP teenuste tegemiseks Microsofti poolt WCF (*Windows Communication Foundation*) raamistik[5], kuid selle serveripoolt pole .NET 5-e jaoks implementeeritud. Iseenesest toetab selle raamistiku eemaldamine eelnevalt töös mainitud arvamust, et SOAP on juba vana ning hakkab aeguma.

GraphQL teenuse arendamiseks on valida kahe erineva raamistiku vahel: Hot Chocolate[6] ja GraphQL.NET[7]. Varasema kogemuse alusel tundus Hot Chocolate raamistiku kasutamine lihtsam ning mugavam. Samuti on see raamistik kiirem ning väiksema mälukasutusega [20]. Seetõttu osutus valituks Hot Chocolate raamistik.

Kasutatava repositooriumi mustri kohta on toodud koodinäited Joonistel 12 ja 13. Nii SOAP kui ka GraphQL teenus kasutavad sama klassi „Repository“, mille käest andmeid küsitakse. Näitel toodud pakendite küsimine on mõlemal teenusel täpselt samamoodi, ainus väike erinevus on päringu mudelis.

[1] https://docs.microsoft.com/en-us/dotnet/core/additional-tools/dotnet-svcutil-guide?tabs=dotnetsvcutil2x

[2] https://github.com/dotnet/efcore

[3] https://github.com/DigDes/SoapCore

[4] https://devblogs.microsoft.com/dotnet/custom-asp-net-core-middleware-example/

[5] https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf

[6] https://github.com/ChilliCream/hotchocolate

[7] https://github.com/graphql-dotnet/graphql-dotnet

```
public class SoapReferenceImpl : webServiceInterface
{
    private readonly Repository _repository;

    public SoapReferenceImpl(Repository repository)
    {
        _repository = repository;
    }

    // ...

    public async Task<pakendidResponse>
            pakendidAsync(pakendidRequest1 request)
    {
        return await _repository.pakendidAsync(request.keha);
    }

    // ...
}
```

Joonis 12. C# nädisrakenduse repositooriumi mustri koodinäide SOAP sõnumite jaoks.

```
    public class Query
    {
        private readonly Repository _repository;

        public Query(Repository repository)
        {
            _repository = repository;
        }

        // ...

        public async Task<pakendidResponse>
                Pakendid(PakendidRequest keha)
        {
            return await _repository.pakendidAsync(keha);
        }

        // ...
    }
```

Joonis 13. C# nädisrakenduse repositooriumi mustri koodinäide GraphQL sõnumite jaoks.

5.6 Java rakenduse kirjeldus

Teine rakendus on realiseeritud kasutades Java 8-t ehk siis sama versioon, mida X-Road kasutab. Kasutatavad mudelid on samuti genereeritud WSDL faili abil kasutades JAXB xjc[1] tööriista. Andmebaasiga suhtlemiseks on kasutusel Hibernate[2] raamistik.

SOAP teenuste arendamiseks on kasutatud Spring Boot[3] raamistikku. Selle juurde on lisatud GraphQL teenus, mis on realiseeritud GraphQL Java abil. Seda raamistikku kasutab ka magistritöö raames valminud konverter. GraphQL Java raamistikule lisaks on veel kasutusel GraphQL SPQR[4], mis teeb GraphQL teenuse lisamise palju lihtsamaks ja automaatsemaks, kuna ei pea eraldi hakkama kõiki GraphQL teenuse kirjelduse välju märkima. C# raamistik Hot Chocolate juba sisaldab sarnast võimekust.

5.7 Andmete kogumine

Andmebaasitabelid on koostatud Ravimiregistri andmebaasi dokumentatsiooni[5] alusel. Uuema MS SQL Serveri kasutuselevõtuga muudatusi teha ei olnud vaja. Testandmebaas erineb dokumentatsioonist kahe erinevusega. Esiteks on mitmes tabelis „EID“ väljade NOT NULL piirang eemaldatud. Neid välju testrakendus ei kasuta ja puudusid andmed nende väljade täitmiseks. Teiseks on mitmes tabelis lisatud „Deleted“ ja „Version“ väljadele vaikimisi väärtuseks „0“, mis lihtsustas andmete lisamist.

Testimisandmed on alla laaditud Ravimiregistri kodulehelt[6]. Saadud CSV failides sisalduvad andmed on sisestatud andmebaasitabelitesse Pythoni skriptifailide abil. Siiski pole avalikult saadaval andmeid kõikide tabelite jaoks, aga neid on piisavalt testimiseks kasutatavate meetodite jaoks.

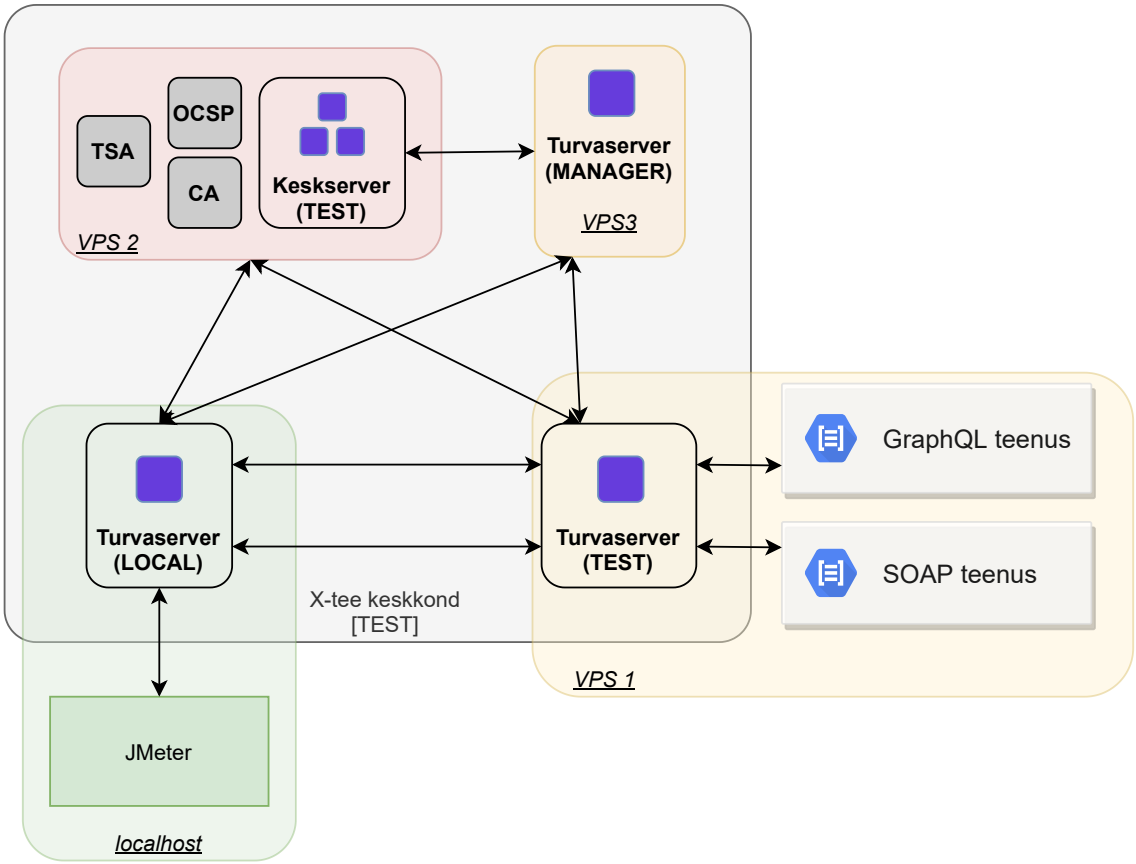
5.8 Testkeskkonna püstitus

Testkeskkond on püstitatud virtuaalserverite abil. Igaühes on operatsioonisüsteemina kasutusel Ubuntu 20.04 ning peale on installeeritud Docker, mille abil on kõik vajalikud rakendused tööle pandud. Nii on lihtne keskkonda hallata ning vajadusel versioone uuendada. Samuti on vajadusel võimalik Dockeri abil hiljem täpselt samasugune keskkond

[1] https://docs.oracle.com/javase/8/docs/technotes/tools/unix/xjc.html
[2] https://hibernate.org/
[3] https://spring.io/projects/spring-boot
[4] https://github.com/leangen/graphql-spqr
[5] https://www.riha.ee/api/v1/systems/ravimiregister/files/29081a92-c375-0408-ccba-e1a45527d3f7
[6] https://www.ravimiregister.ee/?pv=PublicDownloads

luua.

Testkeskkonna arhitektuur on toodud Joonisel 14. Loodud on eraldiseisev X-Road keskkond, kus on üks keskserver koos sertifitseerimisasutusega (CA), ajatembeldusserveriga (TSA) ning OCSP teenusega. Selle kõrval on turvaserver (MANAGER), mis tegeleb liikmete haldamisega.



Joonis 14. Testkeskkonna joonis.

Eelmises lõigus kirjeldatud komponendid on vajalikud X-Road keskkonna seadistamiseks, kuid ei oma olulist tähtsust eksperimentide läbiviimisel, sest turvaserverid vahetavad omavahel sõnumeid otse. Seega nende serverite jõudlus testimisel tähendust ei oma.

Keskkonna tähtsaim komponent on testimise virtuaalserver (VPS 1), mis sisaldab endas turvaserverit ning testimiseks mõeldud teenust koos andmebaasiga. Kõik komponendid on eraldi Dockeri konteinerites, kuid ühises võrgus. Teenusele pääseb ligi ainult turvaserveri kaudu. Virtuaalserveril on 2x2.6 GHz protsessor, 8 GB RAM-i ning võrgukiirus 100 Mbps.

Kolmas komponent (localhost) sisaldab endas samuti turvaserverit (LOCAL), mille abil X-Road keskkonnaga suheldakse. Eksperimentide läbiviimiseks on kasutusel Apache

JMeter[1], mille abil mõõdetakse päringute ja vastuste suurusi ning kulunud aegasid.

5.9 X-Road keskkonna seadistamine

X-Road keskkond nõuab omajagu seadistamist, et kõik osad omavahel suhtlema hakkaksid. Kõigepealt tuleks alustada keskserverist, kuhu saab siseneda veebiliidese kaudu pordilt 4000. Seadistamiseks tehtud sammud on järgmised:

1. keskserveri initsialiseerimine;
2. liikmeklasside (*Member Classes*) lisamine administraatori (MAN) ja liikmete jaoks (MEMBER);
3. sertifitseerimisasutuse lisamine koos OCSP serveriga;
4. ajatempliteenuse lisamine;
5. administratiivse liikme ja alamsüsteemi lisamine (MANAGER);
6. haldusteenuste seadistamine;
7. sisemise ning välimise konfiguratsiooni allkirjastamise võtmete genereerimine;
8. sisemine konfiguratsiooniankru (*configuration anchor*) alla laadimine turvaserveri jaoks.

Nüüdseks on keskserveri seadistamine lõpule viidud ning järgmiseks tuleb seadistada haldusteenuste turvaserver. Selleks tuleb logida sisse turvaserverisse (samuti port 4000) ning sooritada järgnevad sammud:

1. eelnevalt alla laaditud sisemise konfiguratsiooniankru importimine;
2. turvaserveri algseadistamine, PIN koodi sisestamine;
3. ajatempliteenuse seadistamine;
4. allkirjastamis- ja autentimissertifikaatide genereerimine ning nende allkirjastamine sertifitseerimisasutuse abil;
5. keskserveris eelnevalt lisatud liikmele turvaserveri lisamine;
6. keskserveris registreerimistaotluse aksepteerimine.

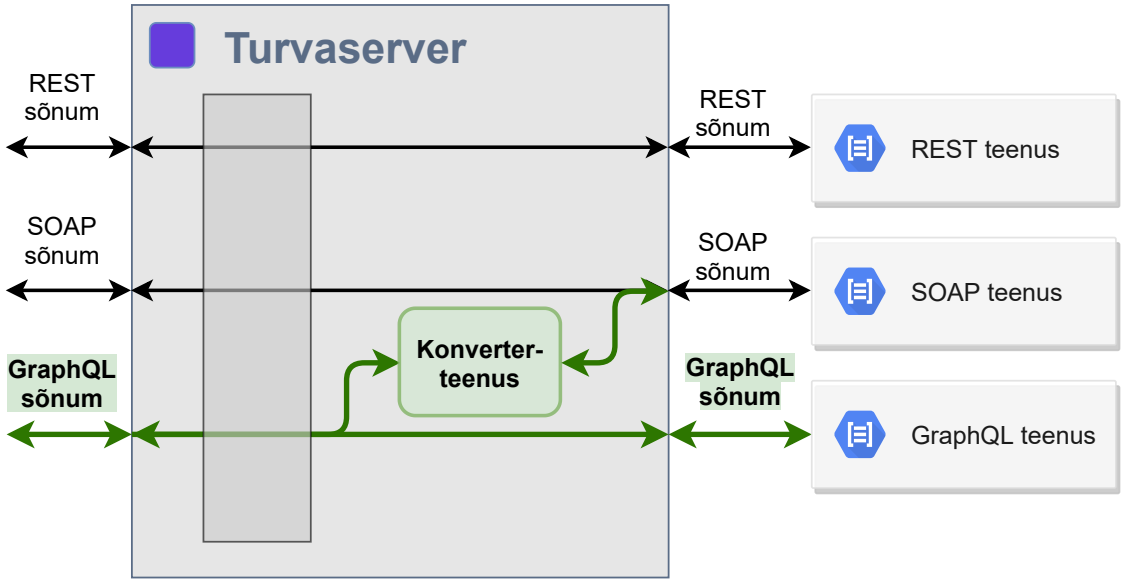
Haldusteenuste turvaserverile on vaja lisada ka alamsüsteem ning see keskserveris registreerida kui haldusteenuseid pakkuv süsteem. Viimaks tuleb lisada turvaserverisse keskserveris kirjas oleva haldusteenus ning see korrektselt seadistada. Kõigi eelneva läbitegemiseks on samm-sammult järgitud Kivimäki poolt tehtud juhised[2].

Lisaks eelnevale on testkeskkonas veel kaks turvaserverit (TEST ja LOCAL), mis ka-

[1] `https://jmeter.apache.org/`
[2] `https://confluence.niis.org/pages/viewpage.action?pageId=6783483`

sutavad magistritöö käigus valminud uuendusi. Teisisõnu sisaldavad need turvaserverid võimalust saata GraphQL sõnumeid ning sisaldavad endas ka konverterit. Tehtud turvaserveri muudatused on välja toodud Joonisel 15 rohelise värviga. Haldusteenuste turvaserver neid osi ei vaja.

Need kaks turvaserverit tuleb samuti seadistada sarnaselt halduseteenuste turvaserveriga (viimasest loetelust punktid 1-6). Teenuseid pakkuvasse turvaserverisse (TEST) on lisatud kaks SOAP teenust (Java ja C#) ning neli GraphQL teenust, millest kaks kasutavad konverterit ning tekkisid automaatselt SOAP teenuse lisamisega.



Joonis 15. Turvaserveris tehtud muudatused.

5.10 Teenuste kirjeldus

Ravimiregistri teenuste WSDL-i ja X-tee liidestusjuhise saab alla laadida kodulehelt[1]. Kokku sisaldab Ravimiregister endas 24 teenust, mille hulgast sai valitud 6 operatsiooni, mis testrakendustes realiseeriti ning millega viidi läbi katsed:

1. ATCKlassifikaatorid,
2. ATCKlassifikaatoridByIDArray,
3. Pakendid,
4. PakendidByIDArray,
5. Soodustused,
6. SoodustusedByIDArray.

Üldiselt on kõik operatsioonid (kahe erisusega) Ravimiregistris antud kahekaupa. Esimene

[1] `https://www.ravimiregister.ee/default.aspx?pv=Kasutajale.XTeeJuhend`

päring tagastab sisendparameetritele sobivate vastete koodid ning teine tagastab koodidele vastavad andmed. Näiteks päring „Soodustused“ tagastab etteantud kuupäevast hiljem muutunud soodustuste koodid ning päring „SoodustusedByIDArray“ tagastab etteantud koodidele vastavad soodustuste andmed. Kõikidele „ByIDArray“ päringutele saab ette anda koodide nimekirja ehk andmeid ei pea küsima ükshaaval.

Kuna operatsioonide tegelik realisatsioon ei ole teada, siis on lahendus tehtud teenuse kirjelduse ning andmebaasiväljade abil. Suurem osa andmebaasiväljadest vastasid üks ühele teenuse kirjelduses sisalduvatele väljadele ning võetakse seega otse andmebaasist. Mõnele üksikule väljale sobivat vastet ei leitud ja seega tagastatakse selle asemel konstante väärtus. Nagu juba varem mainitud, ei mõjuta see testitulemusi, kuna andmete küsimise osa on nii SOAP-i kui GraphQL-i puhul identne.

Lisaks eelnevale kuuele operatsioonile sooritatakse GraphQL-iga veel kaks päringut:

- ATCKlassifikaatorid + Soodustused
- ATCKlassifikaatorid + Soodustused + Pakendid

Need sooritatakse nii konverteri kui puhta GraphQL teenusega ning sisaldavad endas mitme ressursi korraga pärimist.

5.11 Testide läbiviimine

Kõiki päringuid sooritatakse 500 korda ja ükshaaval suvalises järjekorras. Enne testide alustamist jooksutatakse kõik testid ühe korra ilma mõõtmata läbi, et kindlustada testrakenduste ühtlane valmisolek.

Kõikide testide läbiviimiseks on päringud võrdsed, kuid kirjapildis natukene erinevad. Mõlemale SOAP API-le antakse ette täpselt sama XML sõnum, kuna mõlemad kasutavad täpselt sama WSDL skeemi. Pakendite päringu näitesõnum on varasemalt toodud Joonisel 2. GraphQL teenuste puhul on päring teisendatud ringi GraphQL-ile sobivale kujule, mis vastab suuremas osas Joonisel 6 toodule.

Nelja erineva GraphQL API vahel esinevad väikesed erinevused. Erinevuste näitamiseks on soodustuste päringud toodud välja Joonistel 16, 17 ja 18, vastavalt siis konverteri, C# API ning Java API päring. Nagu eelnevas peatükis mainiti, kasutab konverter alati mutatsioonide päringuid. Isetehtud GraphQL kasutab seevastu andmepäringut.

Teine erinevus on „keha“ välja valikus. Konverter jätab WSDL-i alusel „Item“ välja alles.

GraphQL SPQR raamistik, mis Java API-s on kasutusel, teeb genereeritud mudeli põhjal samamoodi. Hot Chocolate raamistik C# API jaoks aga oskab selle välja automaatselt välja jätta, kuna tegelikult ei ole seda vaja. Konverterteenused võtavad mõlemas keeles täpselt sama sisendi, kuna mõlemad neist on tehtud täpselt sama WSDL-i baasil.

```
mutation {
  soodustused(keha: {kuupaev: "2020-01-01"}) {
    paring {
      kuupaev
    }
    keha {
      Item
    }
  }
}
```

Joonis 16. Soodustuste päringu näide GraphQL konverterit kasutades.

```
query {
  soodustused(keha: {kuupaev: "2020-01-01"}) {
    paring {
      kuupaev
    }
    keha
  }
}
```

Joonis 17. Soodustuste päringu näide GraphQL C# API-t kasutades.

```
query {
  soodustused(keha: {kuupaev: "2020-01-01"}) {
    paring {
      kuupaev
    }
    keha {
      item
    }
  }
}
```

Joonis 18. Soodustuste päringu näide GraphQL Java API-t kasutades.

5.12 Lisaeksperimendid

Eelduse kohaselt peaksid GraphQL-i kasutades nii päringute kui ka vastuste mahud tunduvalt vähenema. GraphQL päringud on suured siis, kui küsida on hästi palju erinevaid välju. Samas loendi elemente peaks jällegi olema võimalikult vähe, kuna need on JSON-is kompaktsemad võrreldes XML-iga. Selleks on võetud „PakendidByIDArray“ päring, millel on kõikidest kasutatavatest päringutest kõige rohkem välju ning küsitakse andmed ainult ühe pakendi kohta. Kokkuvõtvalt testitakse selliselt GraphQL päringu suurust selle jaoks kõige halvemates tingimustes.

Jõudluse testimiseks on tehtud teine hulk teste, mille aluseks on võetud „ATCKlassifikaatoridByIDArray“ päring. Päringule saab ette anda ATCKlassifikaatorite koodide loendi ja

selle alusel tagastatakse nende andmed. Testimise käigus tehakse seda ühte päringut läbi erineva hulga elementidega. Nii on võimalik võrrelda SOAP-i ning GraphQL-i tulemusi eri suurusega sõnumite korral. Samuti saab mõõta konverterteenuse lisakulu sõltuvust sõnumi suurusest.

Lisaks eelnevale on koostatud veel kahte sorti teste, et mõõta eraldi päringu ja vastuse suuruse mõju ajale. Nende testide läbiviimiseks on vajalikud muudatused mõlemas testrakenduses.

Vastuse mõju uurimiseks on muudetud API-t nii, et ATCKlassifikaatorite koodide asemel antakse sisse arv, mitu vastust soovitakse. Vastuseks tuleb siis vastavalt nii mitu elementi. Nii on päring hästi väike ning aeg on tugevalt mõjutatud vastuse suurusest ning SOAP või GraphQL raamistiku töötlemise ajast.

Päringute mõju uurimiseks on tehtud vastupidine muudatus. API-le antakse ette loend koodidest ja vastuseks tulevad andmed ainult esimese kohta. See annab vastupidise efekti, et peaaegu kogu aeg on sõltuv ainult päringu suurusest.

6. Tulemused ja järeldused

Antud peatükis antakse ülevaade läbiviidud eksperimentide peamistest tulemustest. Otsitakse neile põhjendusi ning tehakse tulemuste alusel järeldusi. Tuuakse välja statistikat X-tee kasutusest, et analüüsida tulemuste mõju X-tee kontekstis.

6.1 Tulemused

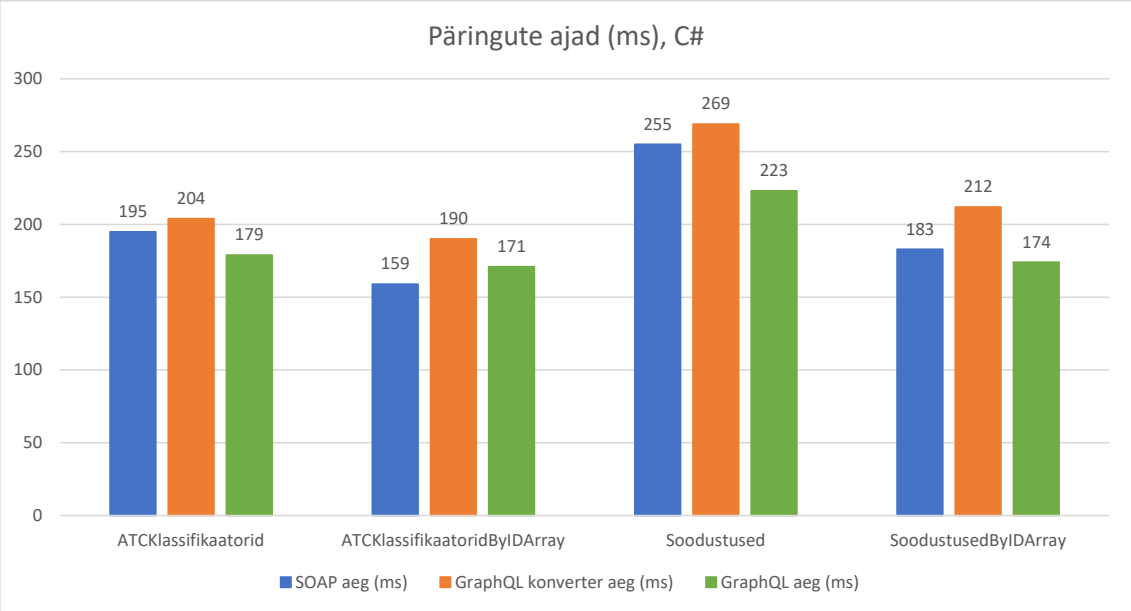
Kõik tulemused on kolmes tabelis on välja toodud Lisas 2. Tabelis 1 on toodud tulemused esialgse näiterakenduse kohta. Teises kahes tabelis on tulemused modifitseeritud rakenduste jaoks, mis on lahti kirjeldatud eelmise peatüki lõpus. Tabelis 2 olevad tulemused on mõjutatud vastuse suurusest. Tabeli 3 tulemused sõltuvad vastupidiselt enamjaolt päringu suurusest.

6.2 Päringute ajad

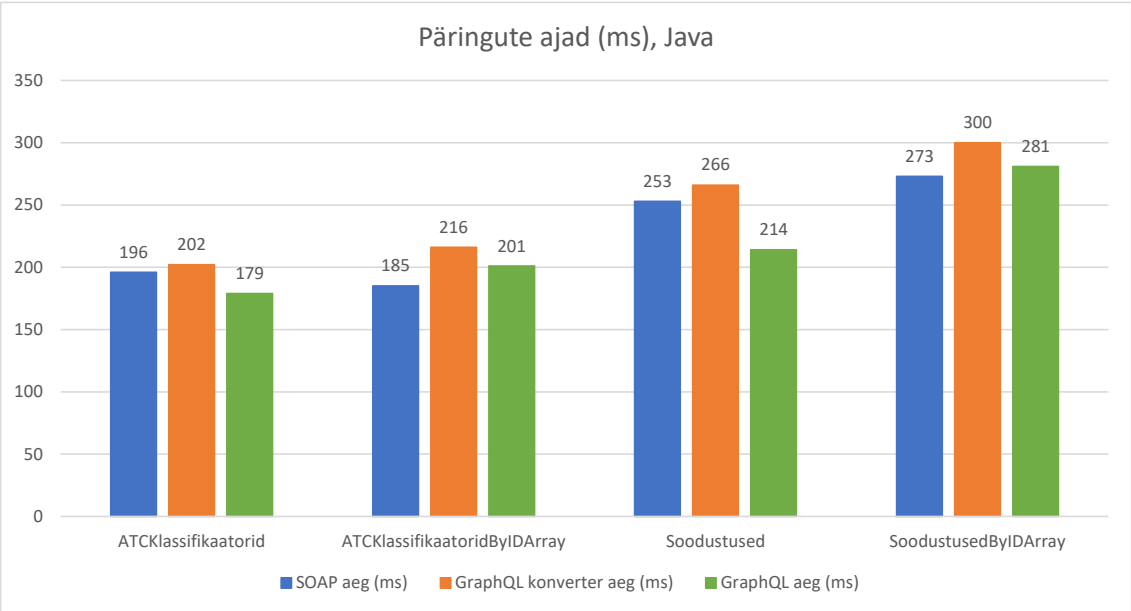
Päringuaegade võrdlused C# API jaoks on toodud Joonisel 19 ja Java API jaoks Joonisel 20. Tulemused on üsna võrdsed ja väga suuri erinevusi näha ei ole. Siiski on konverterit kasutav API alati kõige aeglasem, mis on täiesti loogiline, kuna see lisabki SOAP API ajale juurde konverteerimise aja. Konverteri kasuks räägib aga see, et lisanduv aeg ei ole kuigivõrd suur: nende päringute korral on see keskmiselt 20 ms, mis on umbes 10% suurem SOAP päringu ajast.

GraphQL-i ja SOAP-i omavaheline võrdlus on üsna tasavägine ja on mõnes päringus ühe kasuks ja teises teise kasuks. Kõige suuremad vahed on soodustuste päringus, kus vahe GraphQL-i kasuks on 32 ms (C#) ja 39 ms (Java). SOAP on parem „ATCKlassifikaatoridByIDArray“ päringus, kuid väiksema edumaaga.

C# API ja Java API omavahelised ajad on samuti väga sarnased. Neid aga otseselt ei võrrelda, kuna ei saa garanteerida implementatsioonide võrdsust. Selle tõttu on aga lihtne näha, et päringute suhted jäävad mõlema keele puhul samaks. Kui ühes on GraphQL kiirem, siis nii on ka teises keeles. Ainukene erinevus on „SoodustusedByIDArray“ päring, kuid seal on erinevus üsna väike. Sarnased suhted kinnitavad, et erinevus on tõesti tingitud SOAP-i ja GraphQL-i erinevusest ning et see jääb kehtima erinevates programmeerimiskeeltes.



Joonis 19. Päringute aegade võrdlus C# rakendusega.



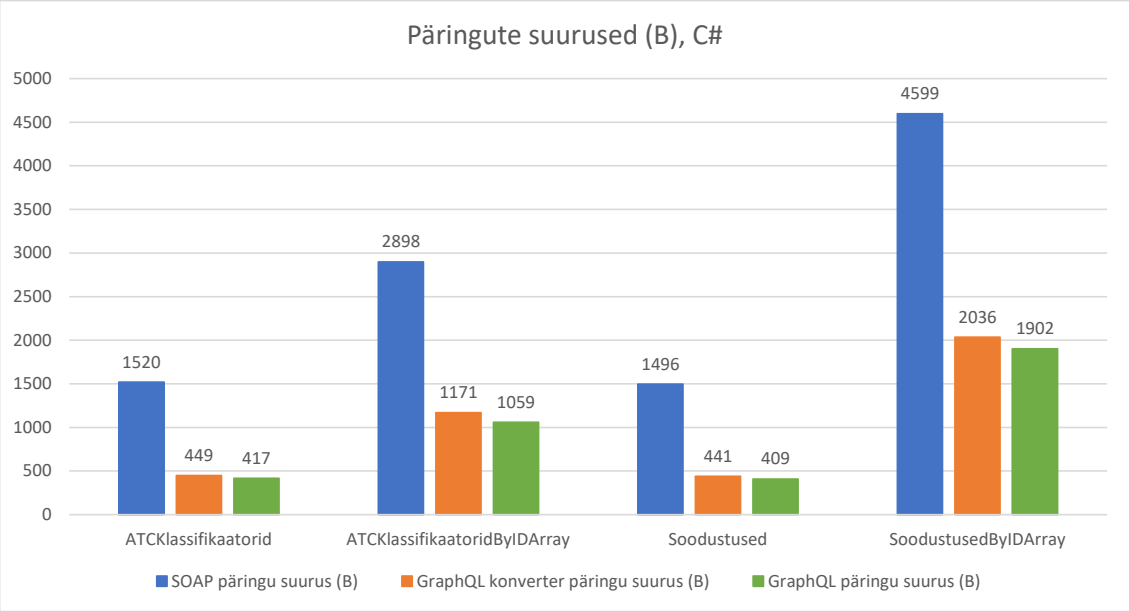
Joonis 20. Päringute aegade võrdlus Java rakendusega.

6.3 Sõnumite suurused

Päringute ning vastuste suurused on C# API ja Java API omavahelisel võrdlusel väga sarnased. Päringute erinevused on lahti seletatud alampeatükis 5.11. Mahtude mõistes jääb erinevus kõikide päringute korral alla 10 protsendi ning suuremate päringute korral on erinevus veel väiksem.

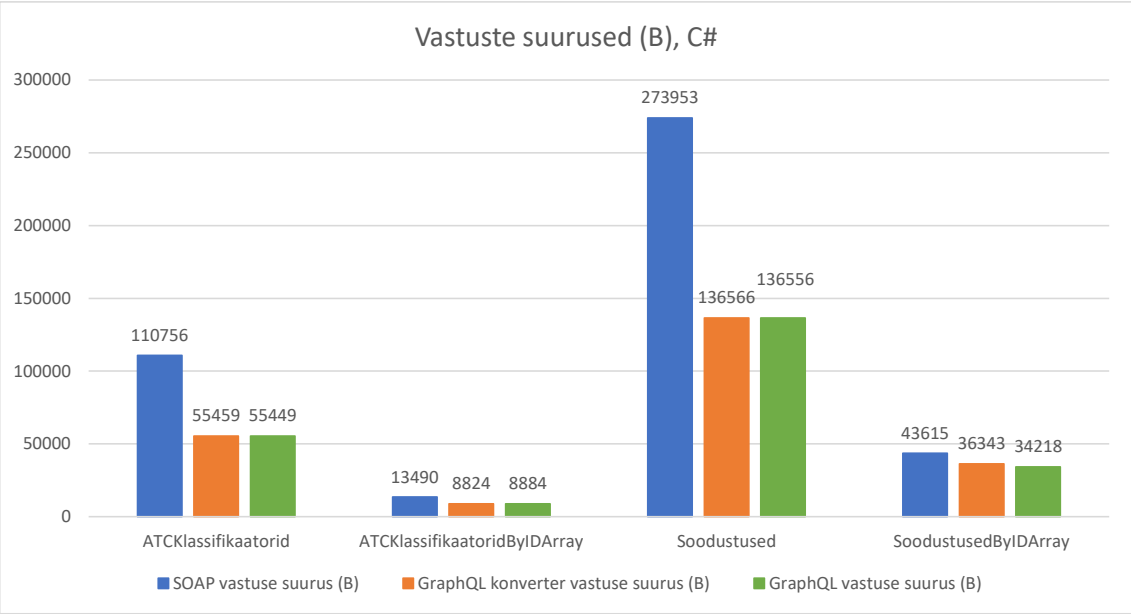
Vastuste suuruse erinevus GraphQL API-de vahel on seletatav sarnaste põhjustega, et ka vastustes on vastavalt need „Item“ elemendid puudu või olemas. SOAP-i korral tekib väike erinevus XML-i päises ning nimeruumide (*namespace*) erinevast kasutamisest, kuid sõnumite sisu need ei muuda. Vastused erinevad samuti väga vähe: kõige rohkem on erinevus 3,5%, kuid jääb enamasti alla 1 protsendi.

Tähtsam on aga SOAP API, konverterteenuse ja GraphQL API omavaheline sõnumite suuruste võrdlus. Päringute suurused C# API korral on toodud Joonisel 21 ning vastuste suurused Joonisel 22. Kuna Java API korral on tulemused väga sarnased, siis ei ole mõtet neid eraldi graafikuna välja tuua.



Joonis 21. Päringute suuruste võrdlus C# rakendusega.

Tulemustest on näha, et konverterteenuse ja GraphQL API korral on sõnumite suuruse erinevus üsna tühine, erinevuste põhjused on juba eelnevalt välja toodud. SOAP-i korral on aga nii päring kui ka sõnum mitmekordselt suurem. Võrguliikluses on GraphQL-iga seega võimalik saavutada märkimisväärne kokkuhoid.



Joonis 22. Vastuste suuruste võrdlus C# rakendusega.

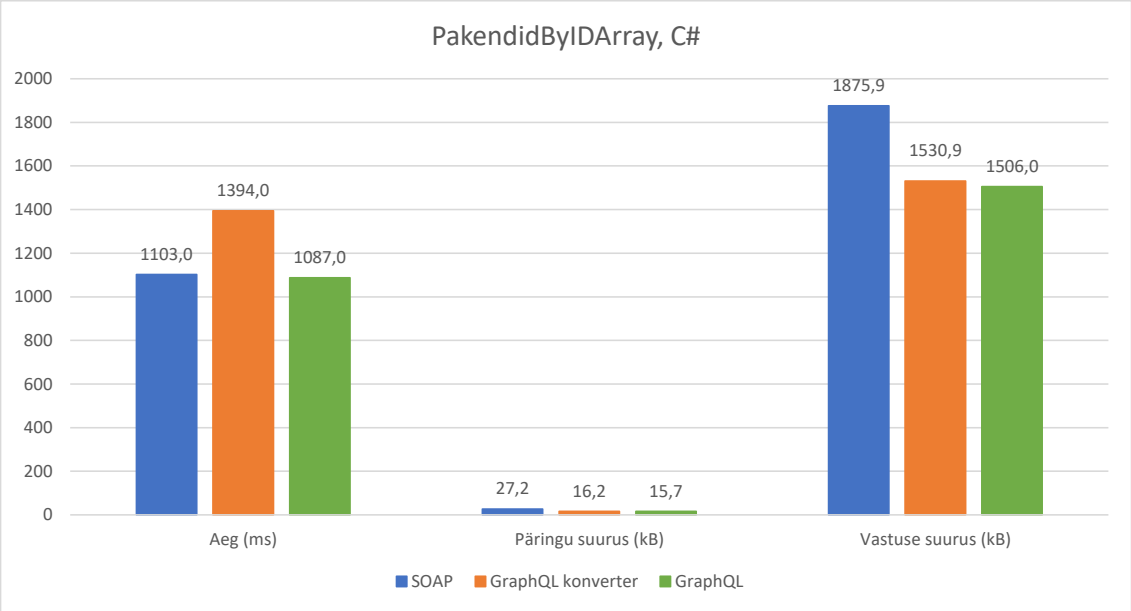
Erinevuste põhjus on põhiliselt seotud JSON-i kompaktsema esitusega võrreldes XML-iga. Viimases on iga elemendi nime jaoks vaja kaks korda rohkem ruumi („<kuupaev>2020-01-01<kuupaev>“ võrreldes „kuupaev: "2020-01-01"“). Lisaks omavad kasutatavates sõnumites suurt rolli loendi elemendid, mis JSON-i puhul on lihtsalt komaga eraldatud, aga XML-is on iga element eraldi XML-sildi vahel („<Item>1593521<Item>“).

Päringute suuruste erinevust, mis on mõnel korral koguni kolmekordne, mõjutab veel XML-i päises asuvad X-Road päiseelemendid. Sama informatsioon on küll olemas ka GraphQL sõnumites, kuid oluliselt kompaktsemal kujul.

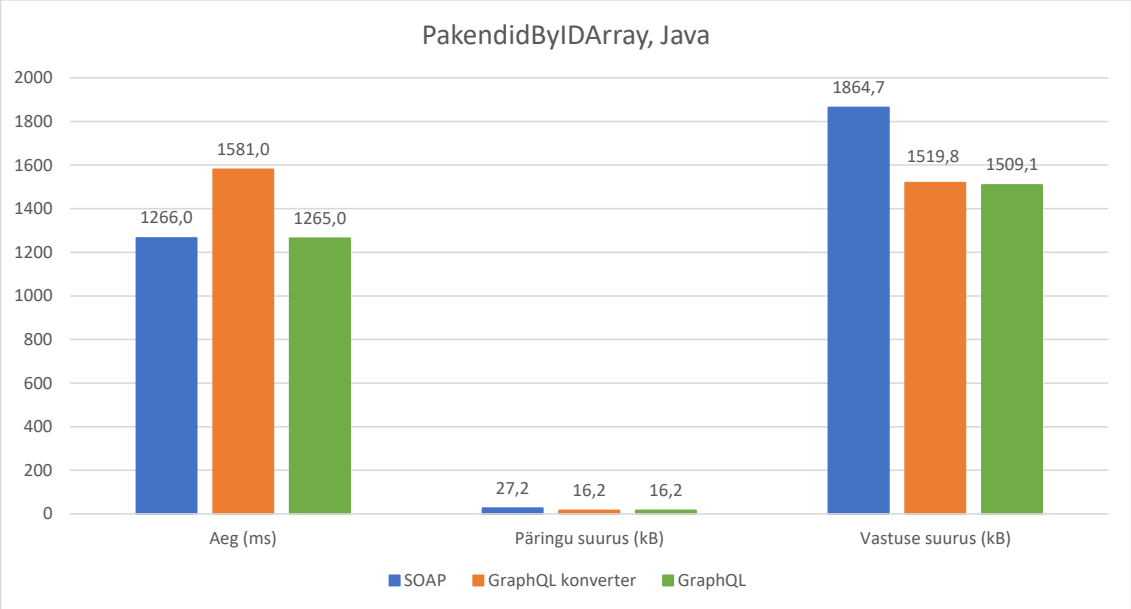
6.4 Suuremad päringud

Eelnevalt toodud näidetes jäi vastuste suurus alla 0,3 kB. Järgnevalt on toodud Joonistel 23 ja 24 tulemused suuremamahulise päringu kohta. Kahte graafikut omavahel võrreldes on jälle näha, et kuigi Java API on kuskil 170 ms aeglasem, on graafikute kujud samasugused. Ehk siis SOAP ja GraphQL API-d käituvad erinevates keeltes sarnaselt.

GraphQL konverter lisab mõlemas programmeerimiskeeles umbes 300 ms SOAP-i päringuajale juurde. Seega on konverteri aeg võrreldes SOAP API-ga keskmiselt 25,5% suurem. Võrreldes väikeste sõnumitega on see juba märgatavam lisaaeg.



Joonis 23. „PakendidByIDArray“ päring C# rakendusega.



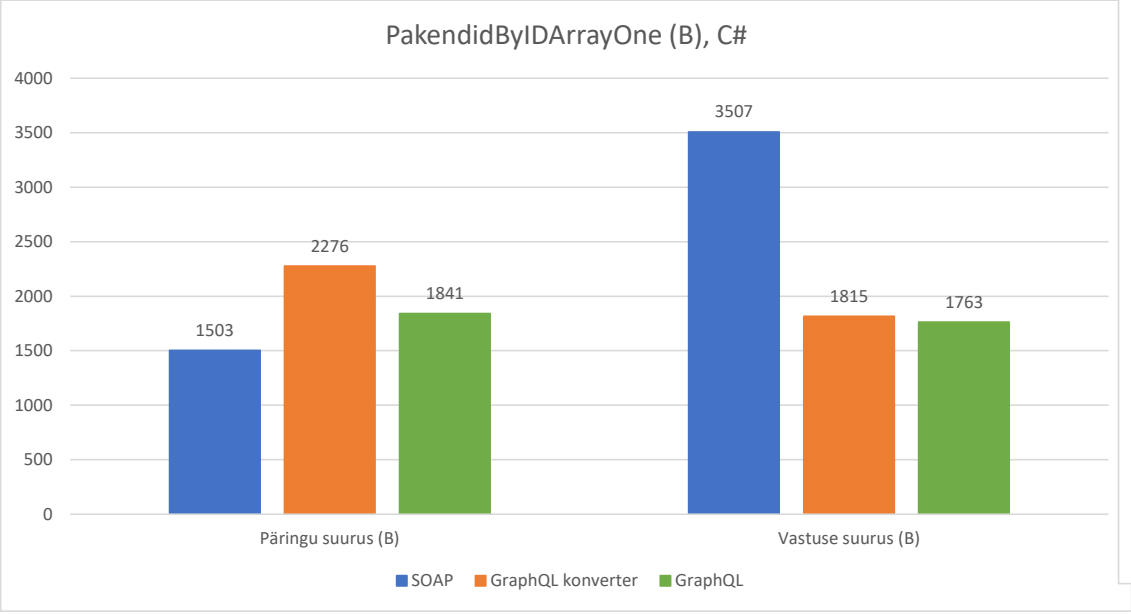
Joonis 24. „PakendidByIDArray“ päring Java rakendusega.

Päringute suurused on üsna tühised võrreldes vastustega, kuid GraphQL-i päring on siiski 1,7 korda väiksem. Huvitavam on aga vastuse suurused, kus GraphQL-i vastus on vaid 1,2 korda väiksem. Vastusi uurides näeb, et põhjuseks on mitmed puuduvad väljad, mis GraphQL tagastab „null“ väärtustena (näiteks „"turvaelemendid":null“). SOAP API jätab seevastu need elemendid üldse vastusest välja ja hoiab seeläbi ruumi kokku.

6.5 Sõnumi suurused

Joonisel 25 on toodud päringu tulemused, mis tekitavad suuruse mõistes GraphQL-i jaoks halvima olukorra. Lühidalt öeldes on päringus palju väljasid, mida kõike ka küsitakse ning andmeid tahetakse ainult ühe koodi alusel. Eksperimenti kirjeldati alampeatüki 5.12 alguses.

Tulemustest selgub, et GraphQL-i päring on 328 B ehk 22,5% suurem kui SOAP päring. Konverteri päring on lausa 51,4% suurem. Vastuste suurused on aga ikkagi GraphQL-i kasuks ja seda peaaegu kahekordselt. Kui nüüd päringu ja vastuse suurused kokku liita, siis saame SOAP sõnumite kogusuuruseks 5010 baiti, konverterteenuse sõnumite suuruseks 4091 baiti ja GraphQL-i sõnumite suuruseks 3604 baiti. Seega kokkuvõttes on SOAP ikkagi mahukam isegi GraphQL-i jaoks halvas olukorras.



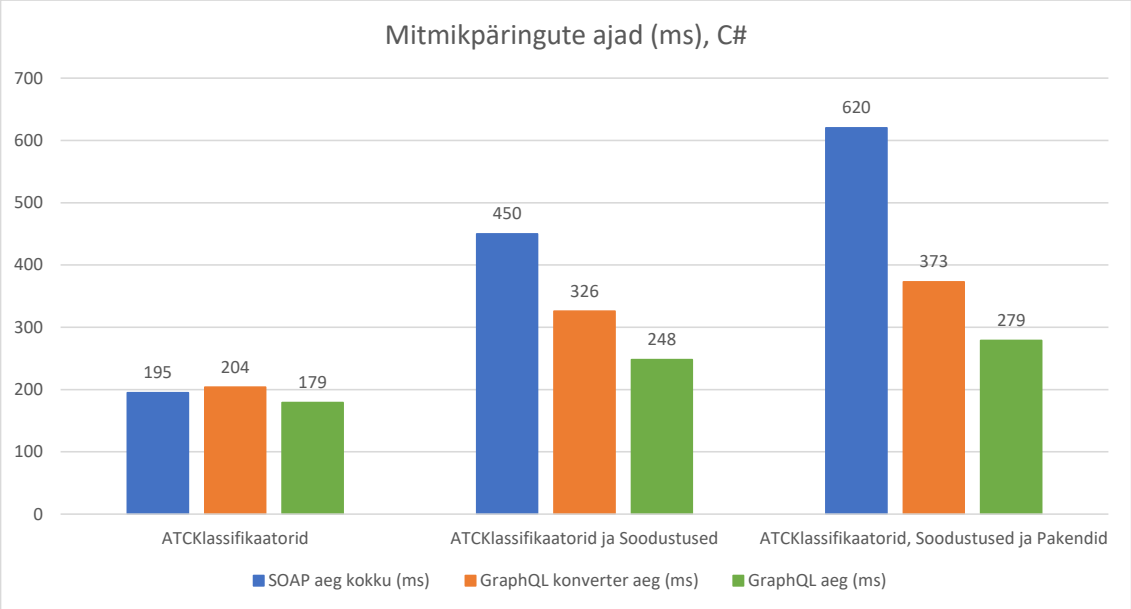
Joonis 25. „PakendidByIDArray“ päring ühe elemendiga C# API-ga.

6.6 Mitmikpäringud

Joonisel 26 on toodud välja mitmikpäringute aegade võrdlus eeldusega, et SOAP päringud tehakse järgepanu. SOAP päringute ajad on toodud kokkuliidetuna. GraphQL API-i ja

konverterteenuse tulemuse jaoks on küsitud mitu ressurssi korraga. On näha, et sellistes tingimustes on isegi konverterit kasutades võimalik saada aja suhtes võitu. Tulemused näitavad, et mida rohkem korraga pärida, seda rohkem ajas võidab. Samas on see väga loogiline tulemus.

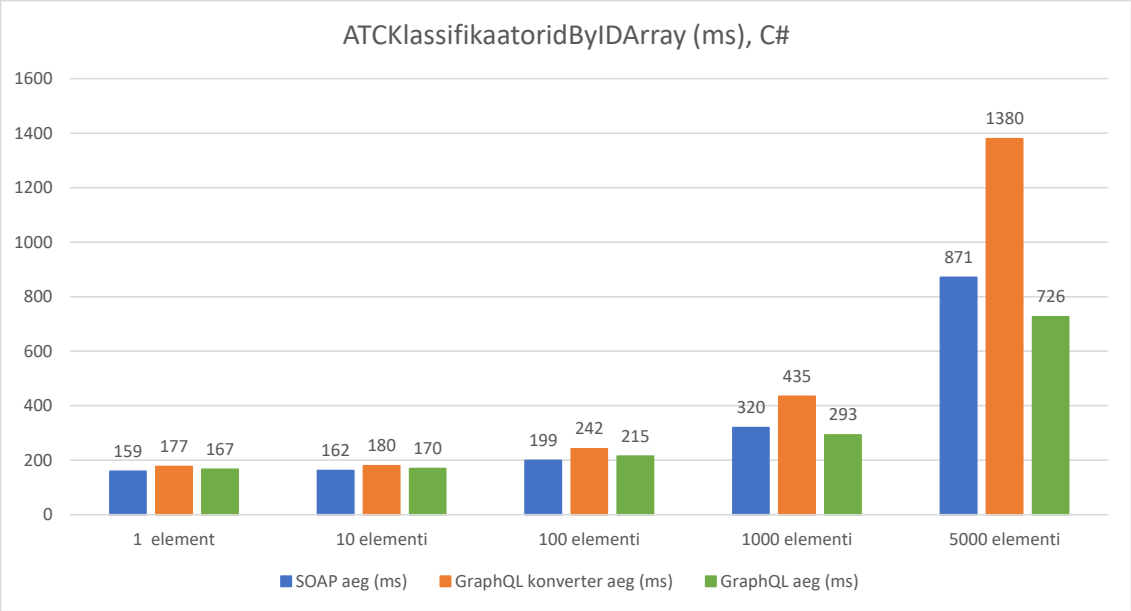
Tulemustest on veel näha, et GraphQL API on efektiivsem kui konverterteenus. Iga lisanduva ressursiga nende aegade omavaheline vahe suureneb. Vahed on vastavalt ühe, kahe ja kolme ressursiga 25 ms, 78 ms, 94 ms. Lisaaeg tekib, kuna konverterteenus peab iga ressursi jaoks tegema eraldi päringu, mille saab kokku panna alles siis, kui kõik vastused on tagasi jõudnud.



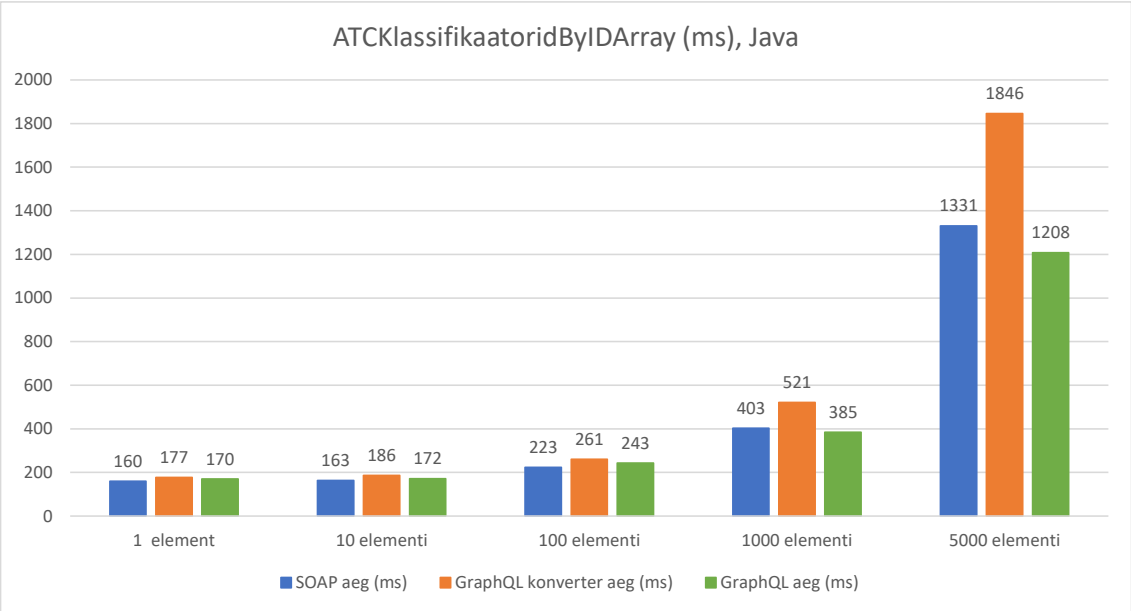
Joonis 26. Mitmikpäringute aegade võrdlus SOAP päringute aegade summaga.

6.7 Suuruse mõju päringu ajale

„ATCKlassifikaatoridByIDArray“ päringut kasutati päringu suuruse ja aja vahelise seose leidmiseks. Eksperimente on selgitatud alampeatükis 5.12. Tulemused on toodud Joonistel 27 (C#) ja 28 (Java). Mõlemad graafikud on jälle väga sarnased. Java API on küll veidi aeglasem, kuid see võib sõltuda tehtud realisatsioonist ning selle alusel järeldusi teha ei saa.



Joonis 27. Päringu aja sõltuvus sõnumi suurusest (C#).



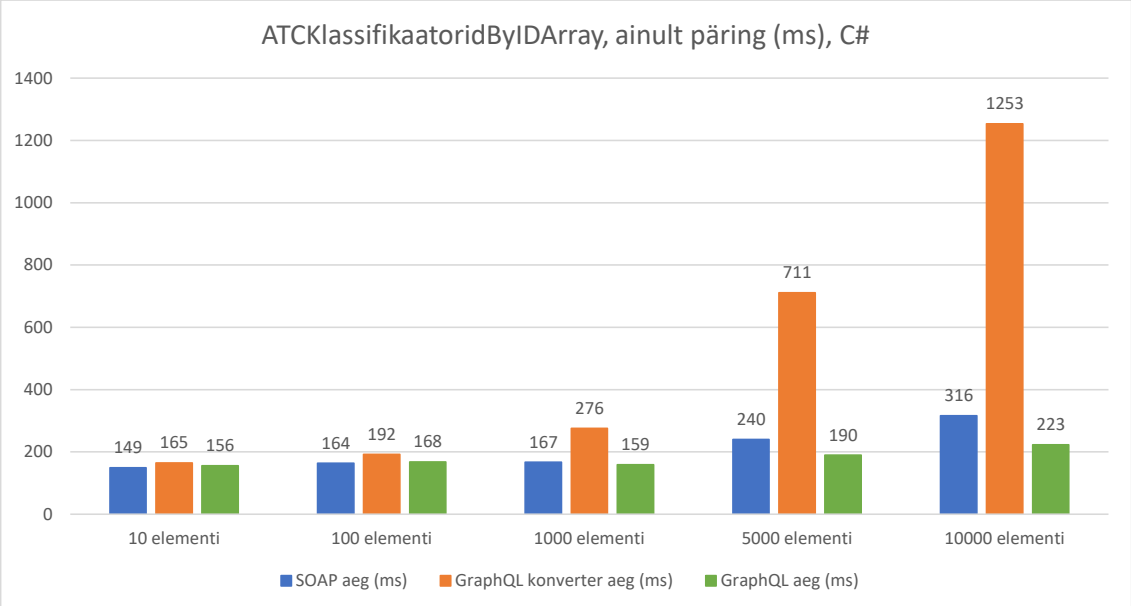
Joonis 28. Päringu aja sõltuvus sõnumi suurusest (Java).

Jättes konverterteenuse esialgu võrdlusest välja, siis tulemustest selgub, et kuni 100 elemendini on tulemused väga sarnased. SOAP on küll natukene kiirem, kuid mitte üle 10 protsendi. Arvudes väljendatuna on selles piirkonnas kõige suurem vahe ainult 20 ms. Tuhande elemendi juures on mõlemas keeles GraphQL juba kiirem, kuid vahe jääb samuti alla 10 protsendi. Viie tuhande elemendi juures on GraphQL juba C# puhul 16,6% ning Java puhul 9,24% kiirem. Üldiselt ei ole vahe siiski väga märkimisväärne, kuid saab oletada, et veel suuremate päringute puhul GraphQL-i edu suureneks.

Konverteri aeg on loogiliselt kõige aeglasem. Ka siin on kuni 100 elemendini vahe üsna väike ja jääb 25% raamidesse. Tuhande elemendi korral on C# keeles konverter juba 36% ning 5000 elemendi juures 58% aeglasem. Sõltuvalt olukorrast ja päringute arvust, võib selline lisakulu juba kasutamist segama hakata. Mõlemas keeles on arvudes väljendatuna vahed väga sarnased: konverter lisab ~115ms 1000 elementi korral ja ~510ms 5000 elemendi korral. Protsentides väljendatuna on Java API vahed väiksemad, kuna Java API realisatsioon on üldplaanis aeglasem.

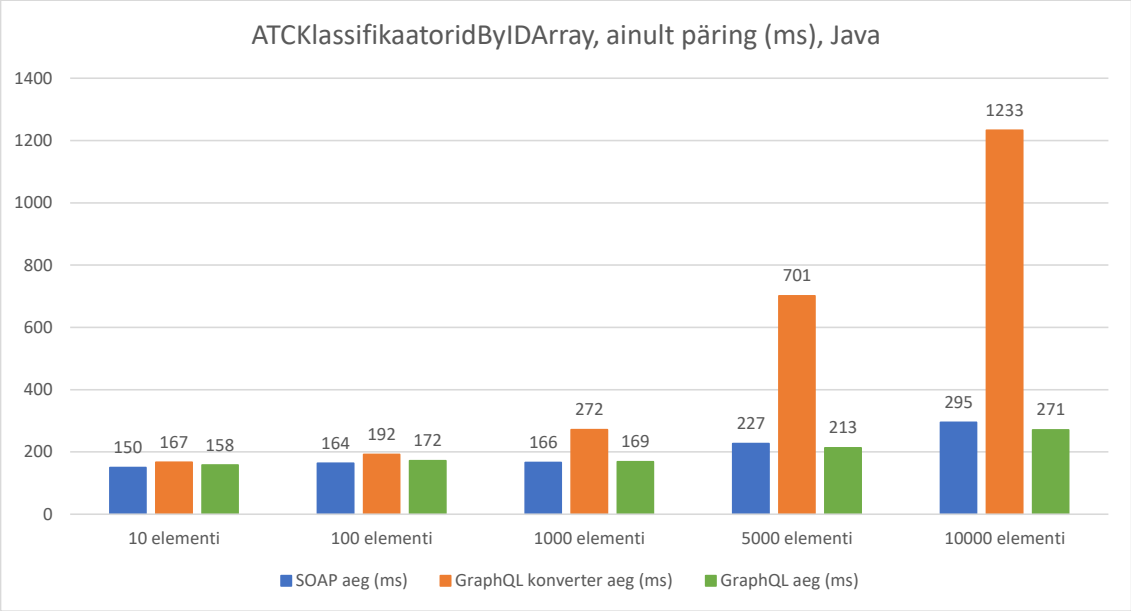
6.7.1 Päringu suuruse mõju

Joonisel 29 toodud tulemused on saadud muudetud API-ga, mis tagastab alati väikese vastuse sõltumata päringu suurusest. Tulemused näitavad seega päringu suuruse mõju ning ei ole olulisel määral vastusest mõjutatud. Kuni 100 elemendini on graafik üsna sarnane eelmistele: GraphQL ning SOAP API on üsna võrdsed ning konverter lisab vähe juurde. Mida rohkem elemente aga lisada, seda rohkem kaldub kasu GraphQL-i poole. 10 000 elemendi juures on GraphQL juba 29% kiirem.



Joonis 29. Ainult päringu suuruse mõju päringu ajale C# API-ga.

Java korral, Joonisel 30, on näha üsna sarnast tulemust, aga vahe GraphQL-i ning SOAP-i vahel on väiksem. Arvestades, et väiksemad päringud on C# ja Java puhul üsna võrdsed, siis siit saab järeldada, et Hot Chocolate raamistik on suuremate päringute puhul efektiivsem kui GraphQL Java.

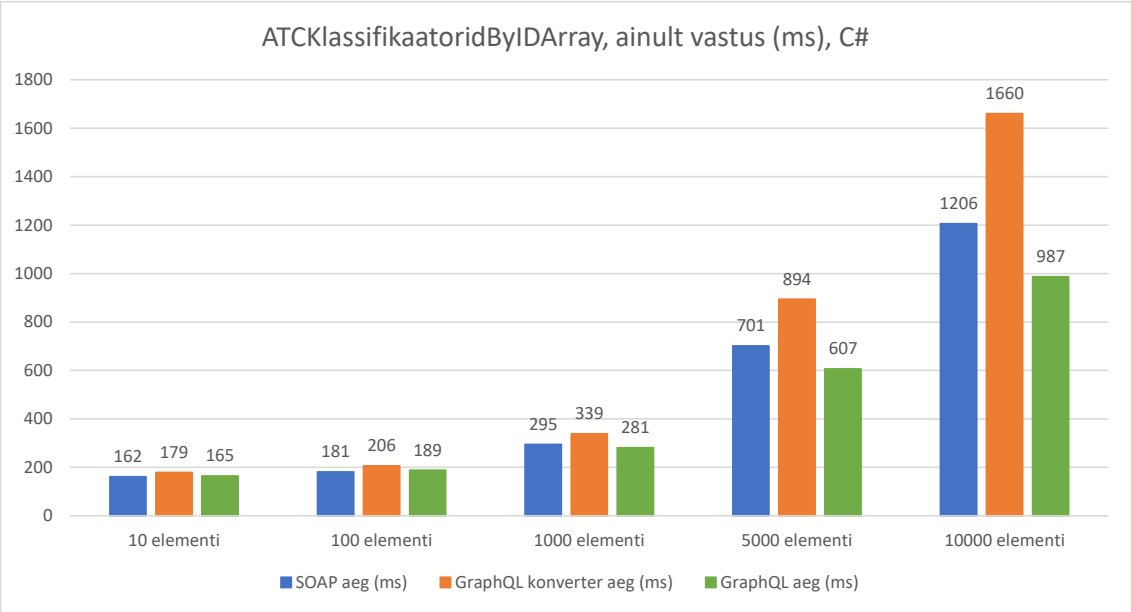


Joonis 30. Ainult päringu suuruse mõju päringu ajale Java API-ga.

Kõige rohkem aga hakkab nendelt joonistelt silma konverterteenuse aeg. Kümne ja saja elemendi korral on konverteri lisatud aeg üsna väike ja jääb 20% raamesse. Tuhande elemendi korral on lisandunud aeg juba ligikaudu 65%, kuid kui päringuid palju ei ole, siis on konverter ikkagi kasutatav. 5000 elemendi korral on vahe juba mitmekordne ning hakkab tugevalt kasutamist segama. Põhjuse leiab konverteri kasutatavas raamistikus (Membrane SOA), mis järelikult ei oska suurte hulkadega kiiresti toime tulla. Siinkohal tasub märkimist, et eelnimetatud raamistikus esines algul viga, mille tõttu vastav raamistik ei osanud üle kümne elemendi päringusse lisada. Magistritöö jaoks antud viga parandati.

6.7.2 Vastuse suuruse mõju

Lisapäringute hulgas oli ka vastuse suuruse mõju uurimine, mille korral sisend oli väike, kuid vastus sisaldas palju elemente. Need viidi läbi muudetud API-ga, kus koodide asemel anti sisse soovitud vastuste arv. Tulemused on Joonisel 31. Kuni tuhande elemendini on SOAP ja GraphQL üsna sarnased, vahe jääb alla 5%. 5000 elemendi juures on GraphQL juba kiirem, kuid kõigest 13,4%. 10 000 elemendi juures on vahe suurenenud 18,2% peale. Saab eeldada, et veel rohkemate elementide korral suureneks nende vahe veelgi.



Joonis 31. Ainult vastuse suuruse mõju päringu ajale C# API-ga.

Võrreldes eelneva eksperimendiga, on konverterteenus vastuse suurusest vähem mõjutatud ja lisab ajale 5000 elemendi juures ainult 27,5% ning 10 000 elemendi juures 37,6%. Oluline on mainida, et eelnevas eksperimendis oli 10 000 elemendiga SOAP päringu suurus 342,5 kB, kuid siin on 10 000 elemendiga vastuse suurus 3,9 MB. Seega saab järeldada, et konverteri vastuse teisendus on oluliselt efektiivsem kui päringu teisendus.

6.8 Tulemused X-tee keskkonna suhtes

X-tee keskkonna kohta on osa statistikast kättesaadav avaandmetena[1]. Analüüsides avalikke andmeid ühe nädala kohta (10.04.2021 - 16.04.2021) selgub, et keskmine päringu suurus oli 7,2 kB ning keskmine vastuse suurus oli 24,0 kB. Keskmine päringule kuluv aeg oli 838,5 ms.

Eelnevatest eksperimentidest selgus, et veel tuhande elemendi juures oli konverteri lisaaeg väike. Nendes sõnumites oli päringu suuruseks 4,9 kB ning vastuse suuruseks 27,6 kB, mis on mõlemad lähedal X-tee keskmisele sõnumi suurusele. Konverter lisas tuhande elemendi korral ligikaudu 110 ms, mis X-tee keskmisele päringuajale lisaks ~11%. Järelikult oleks antud magistritöö käigus valminud konverter enamikes päringutes täiesti kasutatav.

[1] `https://logs.x-tee.ee/EE/gui/`

6.9 Väljade valimine

Kõik läbiviidud eksperimendid GraphQL-iga valisid vastusess alati kõik pakutud väljad. Seega jäi kasutamata GraphQL-i üks põhilisi lisaväärtusi ehk väljade valimine. Loogiliselt võttes ei ole võimalik, et mõne välja mitteküsimine päringut kuidagi aeglasemaks teeks ja seega ei saaks nendes katsetes GraphQL kuidagi SOAP-ile alla jääda.

Sellised eksperimente ei koostatud, kuna niimoodi saavutatav ajaline ja mahuline võit sõltub väga palju konkreetsest olukorrast ja kasutusviisist. Selle tõttu on raske moodustada sobilikku katset, mis oleks mõlemapoolselt aus. Puudusid ka reaalsete kasutuste näited, mille järgi sobivaid eksperimente koostada.

6.10 Järeldused

Kiiruse poolest olid GraphQL ja SOAP väga lähedased. Ei olnud ühtegi katset, kus SOAP oleks olnud märgatavalt kiirem. Seega leidis kinnitust hüpotees, et GraphQL ei ole SOAP-ist aeglasem. Samas kehtib see mõlemat pidi ja ei saa väita, et GraphQL kiirem oleks. Eriti just X-tee kontekstis, kus keskmine sõnum ei ole eriti suuremahuline.

Suuremate päringute korral oli seevastu näha GraphQL-i ülekaalu, mis on arvatavasti tingitud JSON-i kiiremast parsimisest SOAP-iga võrreldes. Kuna ülekaal oli sarnane mõlema programmeerimiskeele juures, siis ei saa see olla juhuslik.

Võrreldes sõnumite suurusi, on GraphQL ja JSON tehtud näiterakendustega alati väiksemad, enamasti vähemalt kaks korda väiksemad. Üksikutel juhtudel võib GraphQL päring suuremaks minna, kuid koos vastuse suurusega on siiski GraphQL väiksem. Lisaks on SOAP sõnumites kohustuslik X-Road päis tunduvalt mahukam, kui sama info GraphQL sõnumites.

Mitmikpäringute hüpotees leidis samuti kinnitust. Korraga pärimisel on võimalus saada ka konverterteenusega isegi mitmekordne kokkuhoid, mis suureneb iga lisa ressursiga. Seega pakub konverter lihtsa viisi olemasolevate SOAP päringute paralleliseerimiseks. Lisaks saab nii veelgi rohkem mahtudelt kokku hoida, kuna ei pea mitu korda saatma SOAP sõnumite X-Road päiseelemente.

7. Kokkuvõte

Antud töö eesmärgiks oli võrrelda SOAP baasil API ning GraphQL-i kasutava API võrdlust X-tee keskkonnas.

Eesmärgi saavutamiseks tuli arendada X-Road keskkonda GraphQL-i tugi. Selle jaoks loodi eraldi turvaserveri versioon, mida eksperimentides kasutati. GraphQL-ile ülemineku soodustamiseks loodi automaatne SOAP-i ja GraphQL-i vaheline konverter olemasolevate teenuste kasutamiseks. Konverter paikneb teenusepakkuja poolses turvaserveris, kuna nii on võimalik väiksemate sõnumite tõttu võrguliikluses säästa.

Eksperimentide läbiviimiseks püstitati eraldi X-Road keskkond, millesse püstitati keskserver, kolm turvaserverit ning teised vajalikud komponendid. Testimiseks koostati näiteteenus reaalselt kasutatava X-tee Ravimiregistri teenuse alusel. Sellest teenusest valiti kuus operatsiooni, mis realiseeriti. Näiteandmed laaditi alla Ravimiregistri kodulehelt.

Näiterakendused koostati kahes erinevas programmeerimisekeeles (C# ja Java), et vähendada realisatsioonist tingitud mõjusid. Iga eksperiment viidi mõlemas keeles läbi kolme erineva API-ga: SOAP, GraphQL ja GraphQL läbi konverterteenuse. Tulemustest selgus, et GraphQL ei olnud ka selle jaoks halvimates tingimustes märgatavalt aeglasem ning suuremate sõnumitega oli GraphQL kiirem.

Selgus, et konverterteenuse lisatud aeg oleks X-tee keskkonnas pigem aksepteeritav ning seega täiesti kasutatav. Lisaks võimaldab konverter vähendada päringute mahtusid. GraphQL ja seega ka konverterteenus võimaldab küsida mitut ressurssi samaaegselt ja seega kokku hoida ajas ning mahus võrreldes järjestikku tehtavate päringutega.

Kokkuvõtvalt selgus, et võrreldes SOAP teenustega ei kaota GraphQL-i kasutuselevõtuga midagi. Lisaväärtused nagu väljade valimine ning mitmikpäringud oleksid aga kindlasti tuleviku mõttes kasulikud.

8. Tulevik

Magistritöö tulemused näitavad, et GraphQL on reaalne alternatiiv SOAP-ile konkreetselt X-tee kontekstis. Siiski leidub veel kohti, mida saaks edasi uurida.

Eksperimendid viidi läbi küll reaalselt kasutusel oleva API alusel, kuid alati küsiti vastusesse kõiki võimalikke väljasid. Kui oleks võimalik saada infot reaalsete kasutusolukordade kohta, siis nende alusel saaks uurida ka väljade valimisest saadavat kasu.

Tehtud konverterteenus katsetati läbi Ravimiregistri teenusega. Lisaks prooviti veel X-tee kataloogist mingi hulk teisi teenuseid. Vajalik oleks aga põhjalikum konverterteenuse valideerimine, mille käigus võiks automaatselt läbi kontrollida kõik kättesaadavad X-tee SOAP teenused.

Jõudluse koha pealt leiti, et konverterteenus on piisavalt kiire enamike päringute jaoks. Samas oli päringu teisendamine palju aeglasem kui vastuse teisendamine. Sellest tulenevalt on kindlasti võimalik vähemalt see osa ümber kirjutada ning paremaks teha.

Kasutatud kirjandus

[1] *X-TEE FACTSHEET EE*. [Online]. Loetud aadressil: https://www.x-tee.ee/factsheets/EE/ Kasutatud: 10.03.2021.

[2] *X-Tee Alamsüsteemide Ja Teenuste Kataloog*. [Online]. Loetud aadressil: https://x-tee.ee/catalogue/EE Kasutatud: 27.04.2021.

[3] Anto Veldre. *Sissejuhatus X-Teesse (Osa 1)*. Riigi Infosüsteemi Ameti blogi. 28. august 2015. [Online]. Loetud aadressil: https://blog.ria.ee/sissejuhatus-x-teesse-osa-1/ Kasutatud: 10.03.2021.

[4] *Andmevahetuskiht X-tee*. Riigi Infosüsteemi Amet. [Online]. Loetud aadressil: https://www.ria.ee/et/riigi-infosusteem/andmevahetuskiht-x-tee.html Kasutatud: 15.03.2021.

[5] *X-Road® History*. X-Road® Data Exchange Layer. [Online]. Loetud aadressil: https://x-road.global/xroad-history Kasutatud: 15.03.2021.

[6] A. Kalja, T. Robal ja U. Vallner. „New Generations of Estonian eGovernment Components“. Teoses: *2015 Portland International Conference on Management of Engineering and Technology (PICMET)*. 2015 Portland International Conference on Management of Engineering and Technology (PICMET). August 2015, lk. 625–631. DOI: 10.1109/PICMET.2015.7273002.

[7] Festim Halili ja Erenis Ramadani. „Web Services: A Comparison of Soap and Rest Services“. *Modern Applied Science* 12.3 (veebruar 2018), lk. 9. DOI: 10.5539/mas.v12n3p175.

[8] *X-Road Architecture Technical Specification*. Nordic Institute for Interoperability Solutions. [Online]. Loetud aadressil: https://github.com/nordic-institute/X-Road/blob/develop/doc/Architecture/arc-g_x-road_arhitecture.md Kasutatud: 15.03.2021.

[9] *Web Service Definition Language (WSDL)*. W3C. 15. märts 2001. [Online]. Loetud aadressil: https://www.w3.org/TR/2001/NOTE-wsdl-20010315 Kasutatud: 15.03.2021.

[10] *X-Road: Message Protocol v4.0*. [Online]. Loetud aadressil: https://www.x-tee.ee/docs/live/xroad/pr-mess_x-road_message_protocol.html Kasutatud: 11.03.2021.

[11] Jürgen Šuvalov. *X-Road REST Tugi*. Riigi Infosüsteemi Ameti blogi. 13. aprill 2020. [Online]. Loetud aadressil: https://blog.ria.ee/x-road-rest-tugi/ Kasutatud: 27.04.2021.

[12] Saurabh Zunke ja Veronica D'Souza. „JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats". *IJCSN International Journal of Computer Science and Network* 3.4 (august 2014). ISSN: 2277-5420. [Online]. Loetud aadressil: http://ijcsn.org/IJCSN-2014/3-4/JSON-vs-XML-A-Comparative-Performance-Analysis-of-Data-Exchange-Formats.pdf.

[13] *The GraphQL Foundation | An Open and Neutral Home for the GraphQL Community*. [Online]. Loetud aadressil: https://foundation.graphql.org/ Kasutatud: 15.03.2021.

[14] Jaime Sayago Heredia, Evelin Flores-García ja Andres Recalde Solano. „Comparative Analysis Between Standards Oriented to Web Services: SOAP, REST and GRAPHQL". Teoses: *Applied Technologies*. Toim. Miguel Botto-Tobar *et al.* Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, lk. 286–300. ISBN: 978-3-030-42517-3. DOI: 10.1007/978-3-030-42517-3_22.

[15] G. Brito, T. Mombach ja M. T. Valente. „Migrating to GraphQL: A Practical Assessment". Teoses: *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). Veebruar 2019, lk. 140–150. DOI: 10.1109/SANER.2019.8667986.

[16] Jayden Seric. *Jaydenseric/Graphql-Multipart-Request-Spec*. 4. mai 2021. [Online]. Loetud aadressil: https://github.com/jaydenseric/graphql-multipart-request-spec Kasutatud: 09.05.2021.

[17] *GraphQL Specification*. Juuni 2018. [Online]. Loetud aadressil: http://spec.graphql.org/June2018/ Kasutatud: 18.03.2021.

[18] *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C. 16. august 2006. [Online]. Loetud aadressil: https://www.w3.org/TR/xml11/#sec-documents Kasutatud: 03.05.2021.

[19] *TIOBE Index*. TIOBE - The Software Quality Company. [Online]. Loetud aadressil: https://www.tiobe.com/tiobe-index/ Kasutatud: 26.04.2021.

[20] Michael Staib. *Welcome Hot Chocolate 11*. ChilliCream. 23. november 2020. [Online]. Loetud aadressil: https://chillicream.com/blog/2020/11/23/hot-chocolate-11/#execution-engine Kasutatud: 21.04.2021.

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

Mina, Siim Düüna

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "SOAP-i ja GraphQL-i võrdlus X-tee näitel", mille juhendaja on Tarvo Treier
   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.

10.05.2021

[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.

Lisa 2 - Eksperimentide tulemused

Tabel 1. Eksperimentide tulemused.

Sõnum	Keel	Variant	Mediaan-aeg (ms)	Päringu suurus (B)	Vastuse suurus (B)
ATCKlassifikaatorid	C#	Konverter	204	449	55459
ATCKlassifikaatorid	C#	GraphQL	179	417	55449
ATCKlassifikaatorid	C#	Soap	195	1520	110756
ATCKlassifikaatorid	Java	Konverter	202	449	55459
ATCKlassifikaatorid	Java	GraphQL	179	435	55528
ATCKlassifikaatorid	Java	Soap	196	1520	110700
ATCKlassifikaatoridByIDArray	C#	Konverter	190	1171	8824
ATCKlassifikaatoridByIDArray	C#	GraphQL	171	1059	8884
ATCKlassifikaatoridByIDArray	C#	Soap	159	2898	13490
ATCKlassifikaatoridByIDArray	Java	Konverter	216	1171	8824
ATCKlassifikaatoridByIDArray	Java	GraphQL	201	1157	8843
ATCKlassifikaatoridByIDArray	Java	Soap	185	2898	13434
Pakendid	C#	Konverter	197	591	11290
Pakendid	C#	GraphQL	182	559	11280
Pakendid	C#	Soap	170	1653	23389
Pakendid	Java	Konverter	198	591	11290
Pakendid	Java	GraphQL	184	577	11359
Pakendid	Java	Soap	170	1653	23333
PakendidByIDArray	C#	Konverter	1394	16174	1530901
PakendidByIDArray	C#	GraphQL	1087	15739	1506004
PakendidByIDArray	C#	Soap	1103	27155	1875863
PakendidByIDArray	Java	Konverter	1581	16174	1519809
PakendidByIDArray	Java	GraphQL	1265	16160	1509059
PakendidByIDArray	Java	Soap	1266	27155	1864707
PakendidByIDArrayOne	C#	Konverter	218	2276	1815
PakendidByIDArrayOne	C#	GraphQL	204	1841	1763
PakendidByIDArrayOne	C#	Soap	184	1503	3507
PakendidByIDArrayOne	Java	Konverter	237	2276	1803
PakendidByIDArrayOne	Java	GraphQL	226	2262	1864
PakendidByIDArrayOne	Java	Soap	200	1503	3439
Soodustused	C#	Konverter	269	441	136566
Soodustused	C#	GraphQL	223	409	136556
Soodustused	C#	Soap	255	1496	273953
Soodustused	Java	Konverter	266	441	136566
Soodustused	Java	GraphQL	214	427	136635

Tabel 1. Eksperimentide tulemused.

Sõnum	Keel	Variant	Mediaan-aeg (ms)	Päringu suurus (B)	Vastuse suurus (B)
Soodustused	Java	Soap	253	1496	273897
SoodustusedByIDArray	C#	Konverter	212	2036	36343
SoodustusedByIDArray	C#	GraphQL	174	1902	34218
SoodustusedByIDArray	C#	Soap	183	4599	43615
SoodustusedByIDArray	Java	Konverter	300	2036	35055
SoodustusedByIDArray	Java	GraphQL	281	2022	34684
SoodustusedByIDArray	Java	Soap	273	4599	42271
ATCKlassifikaatoridByIDArray1	C#	Konverter	177	651	678
ATCKlassifikaatoridByIDArray1	C#	GraphQL	167	539	648
ATCKlassifikaatoridByIDArray1	C#	Soap	159	1538	2112
ATCKlassifikaatoridByIDArray1	Java	Konverter	177	651	678
ATCKlassifikaatoridByIDArray1	Java	GraphQL	170	637	745
ATCKlassifikaatoridByIDArray1	Java	Soap	160	1538	2056
ATCKlassifikaatoridByIDArray10	C#	Konverter	180	768	1975
ATCKlassifikaatoridByIDArray10	C#	GraphQL	170	656	1862
ATCKlassifikaatoridByIDArray10	C#	Soap	162	1844	3849
ATCKlassifikaatoridByIDArray10	Java	Konverter	186	768	1975
ATCKlassifikaatoridByIDArray10	Java	GraphQL	172	754	2024
ATCKlassifikaatoridByIDArray10	Java	Soap	163	1844	3793
ATCKlassifikaatoridByIDArray100	C#	Konverter	242	1939	19197
ATCKlassifikaatoridByIDArray100	C#	GraphQL	215	1827	19042
ATCKlassifikaatoridByIDArray100	C#	Soap	199	4904	27680
ATCKlassifikaatoridByIDArray100	Java	Konverter	261	1939	19197
ATCKlassifikaatoridByIDArray100	Java	GraphQL	243	1924	19116
ATCKlassifikaatoridByIDArray100	Java	Soap	223	4904	27624
ATCKlassifikaatoridByIDArray1000	C#	Konverter	435	13640	178376
ATCKlassifikaatoridByIDArray1000	C#	GraphQL	293	13528	175655
ATCKlassifikaatoridByIDArray1000	C#	Soap	320	35505	245444
ATCKlassifikaatoridByIDArray1000	Java	Konverter	521	13640	178376
ATCKlassifikaatoridByIDArray1000	Java	GraphQL	385	13626	176757
ATCKlassifikaatoridByIDArray1000	Java	Soap	403	35505	245388
ATCKlassifikaatoridByIDArray5000	C#	Konverter	1380	66639	990442
ATCKlassifikaatoridByIDArray5000	C#	GraphQL	726	66527	1001304
ATCKlassifikaatoridByIDArray5000	C#	Soap	871	172505	1366335
ATCKlassifikaatoridByIDArray5000	Java	Konverter	1846	66639	990442
ATCKlassifikaatoridByIDArray5000	Java	GraphQL	1208	66625	982588
ATCKlassifikaatoridByIDArray5000	Java	Soap	1331	172505	1366279
ATCKlassifikaatorid+Soodustused	C#	Konverter	326	571	191549
ATCKlassifikaatorid+Soodustused	C#	GraphQL	248	518	191530
ATCKlassifikaatorid+Soodustused	Java	Konverter	325	571	191549
ATCKlassifikaatorid+Soodustused	Java	GraphQL	244	557	191618
ATCKlassifikaatorid+Soodustused+Pakendid	C#	Konverter	373	843	202389

Tabel 1. Eksperimentide tulemused.

Sõnum	Keel	Variant	Mediaan-aeg (ms)	Päringu suurus (B)	Vastuse suurus (B)
ATCKlassifikaatorid+Soodustused+Pakendid	C#	GraphQL	279	769	202361
ATCKlassifikaatorid+Soodustused+Pakendid	Java	Konverter	381	843	202389
ATCKlassifikaatorid+Soodustused+Pakendid	Java	GraphQL	278	829	202458
Tühi sõnum			19	115	560

Tabel 2. Eksperimentide tulemused muudetud API-ga, mis sõltuvad vastuse suurusest.

Sõnum	Keel	Variant	Mediaan-aeg (ms)	Päringu suurus (B)	Vastuse suurus (B)
ATCKlassifikaatoridByIDArray1	C#	Konverter	178	645	793
ATCKlassifikaatoridByIDArray1	C#	GraphQL	167	533	792
ATCKlassifikaatoridByIDArray1	C#	Soap	157	1532	2302
ATCKlassifikaatoridByIDArray1	Java	Konverter	180	645	793
ATCKlassifikaatoridByIDArray1	Java	GraphQL	168	631	862
ATCKlassifikaatoridByIDArray1	Java	Soap	161	1532	2246
ATCKlassifikaatoridByIDArray10	C#	Konverter	179	646	3188
ATCKlassifikaatoridByIDArray10	C#	GraphQL	165	534	3349
ATCKlassifikaatoridByIDArray10	C#	Soap	162	1533	5812
ATCKlassifikaatoridByIDArray10	Java	Konverter	182	646	3188
ATCKlassifikaatoridByIDArray10	Java	GraphQL	169	632	3257
ATCKlassifikaatoridByIDArray10	Java	Soap	159	1533	5756
ATCKlassifikaatoridByIDArray100	C#	Konverter	206	647	27129
ATCKlassifikaatoridByIDArray100	C#	GraphQL	189	535	28910
ATCKlassifikaatoridByIDArray100	C#	Soap	181	1534	40939
ATCKlassifikaatoridByIDArray100	Java	Konverter	210	647	27129
ATCKlassifikaatoridByIDArray100	Java	GraphQL	193	633	27198
ATCKlassifikaatoridByIDArray100	Java	Soap	183	1534	40883
ATCKlassifikaatoridByIDArray1000	C#	Konverter	339	648	266610
ATCKlassifikaatoridByIDArray1000	C#	GraphQL	281	536	284592
ATCKlassifikaatoridByIDArray1000	C#	Soap	295	1535	392019
ATCKlassifikaatoridByIDArray1000	Java	Konverter	347	648	266610
ATCKlassifikaatoridByIDArray1000	Java	GraphQL	286	634	266679
ATCKlassifikaatoridByIDArray1000	Java	Soap	307	1535	391963
ATCKlassifikaatoridByIDArray5000	C#	Konverter	894	648	1330867
ATCKlassifikaatoridByIDArray5000	C#	GraphQL	607	536	1420872
ATCKlassifikaatoridByIDArray5000	C#	Soap	701	1535	1952403
ATCKlassifikaatoridByIDArray5000	Java	Konverter	922	648	1330867
ATCKlassifikaatoridByIDArray5000	Java	GraphQL	597	634	1330936
ATCKlassifikaatoridByIDArray5000	Java	Soap	743	1535	1952347
ATCKlassifikaatoridByIDArray10000	C#	Konverter	1660	649	2661195
ATCKlassifikaatoridByIDArray10000	C#	GraphQL	987	537	2841216
ATCKlassifikaatoridByIDArray10000	C#	Soap	1206	1536	3902882
ATCKlassifikaatoridByIDArray10000	Java	Konverter	1743	649	2661195
ATCKlassifikaatoridByIDArray10000	Java	GraphQL	994	635	2661264
ATCKlassifikaatoridByIDArray10000	Java	Soap	1292	1536	3902826

Tabel 3. Eksperimentide tulemused muudetud API-ga, mis sõltuvad päringute suurusest.

Sõnum	Keel	Variant	Mediaan-aeg (ms)	Päringu suurus (B)	Vastuse suurus (B)
ATCKlassifikaatoridByIDArray1	C#	Konverter	166	651	678
ATCKlassifikaatoridByIDArray1	C#	GraphQL	154	539	648
ATCKlassifikaatoridByIDArray1	C#	Soap	149	1538	2112
ATCKlassifikaatoridByIDArray1	Java	Konverter	166	651	678
ATCKlassifikaatoridByIDArray1	Java	GraphQL	157	637	745
ATCKlassifikaatoridByIDArray1	Java	Soap	151	1538	2056
ATCKlassifikaatoridByIDArray10	C#	Konverter	165	768	678
ATCKlassifikaatoridByIDArray10	C#	GraphQL	156	656	648
ATCKlassifikaatoridByIDArray10	C#	Soap	149	1844	2112
ATCKlassifikaatoridByIDArray10	Java	Konverter	167	768	678
ATCKlassifikaatoridByIDArray10	Java	GraphQL	158	754	745
ATCKlassifikaatoridByIDArray10	Java	Soap	150	1844	2056
ATCKlassifikaatoridByIDArray100	C#	Konverter	192	1939	678
ATCKlassifikaatoridByIDArray100	C#	GraphQL	168	1827	648
ATCKlassifikaatoridByIDArray100	C#	Soap	164	4904	2112
ATCKlassifikaatoridByIDArray100	Java	Konverter	192	1939	678
ATCKlassifikaatoridByIDArray100	Java	GraphQL	172	1924	745
ATCKlassifikaatoridByIDArray100	Java	Soap	164	4904	2056
ATCKlassifikaatoridByIDArray1000	C#	Konverter	276	13640	678
ATCKlassifikaatoridByIDArray1000	C#	GraphQL	159	13528	648
ATCKlassifikaatoridByIDArray1000	C#	Soap	167	35505	2112
ATCKlassifikaatoridByIDArray1000	Java	Konverter	272	13640	678
ATCKlassifikaatoridByIDArray1000	Java	GraphQL	169	13626	745
ATCKlassifikaatoridByIDArray1000	Java	Soap	166	35505	2056
ATCKlassifikaatoridByIDArray5000	C#	Konverter	711	66639	678
ATCKlassifikaatoridByIDArray5000	C#	GraphQL	190	66527	648
ATCKlassifikaatoridByIDArray5000	C#	Soap	240	172505	2112
ATCKlassifikaatoridByIDArray5000	Java	Konverter	701	66639	678
ATCKlassifikaatoridByIDArray5000	Java	GraphQL	213	66625	745
ATCKlassifikaatoridByIDArray5000	Java	Soap	227	172505	2056
ATCKlassifikaatoridByIDArray10000	C#	Konverter	1253	131640	678
ATCKlassifikaatoridByIDArray10000	C#	GraphQL	223	131528	648
ATCKlassifikaatoridByIDArray10000	C#	Soap	316	342505	2112
ATCKlassifikaatoridByIDArray10000	Java	Konverter	1233	131640	678
ATCKlassifikaatoridByIDArray10000	Java	GraphQL	271	131626	745
ATCKlassifikaatoridByIDArray10000	Java	Soap	295	342505	2056