SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

MINIMIKONTEKSTINHALLINNAN TEKNOLOGIAUUDISTUS

TEKIJÄ/T: Jenna Räsänen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Jenna Räsänen	
Työn nimi Minimikontekstinhallinnan teknologiauudistus	
Päiväys 17.5.2020	Sivumäärä/Liitteet 19
Ohjaaja(t) Lehtori Sami Lahti, lehtori Jussi Koistinen	
Toimeksiantaja/Yhteistyökumppani(t) Mediconsult Oy	

Tiivistelmä

Opinnäytetyön tarkoituksena oli toteuttaa minimikontekstinhallinnan teknologiauudistus. Minimikontekstinhallinta on työpöytäintegraatio, jonka tarkoituksena on helpottaa yhteisen kontekstin ja kertakirjautumisen avulla erillisten terveydenhuollon järjestelmien yhtäaikaista käyttöä. Minimikontekstinhallinnan toiminta perustuu CCOW (Clinical Context Object Workgroup) standardin pohjalta määriltyyn terveydenhuollon sovellusten yhteiseen kontekstiin, jossa on joukko samalla tavalla käsiteltäviä tietoja.

Työssä käytetyt teknologiat olivat Java, Wildfly 16, Lettuce, Redis ja Redis Sentinel. Työ toteutettiin Javan versiolla 11. Wildfly toimii palvelimena minimikontekstinhallinnalle. Lettuce on asiakasohjelma Redikselle, joka on tietorakenne varasto, jossa kontekstin tietoja säilötään istunnon aikana. Redis Sentinel mahdollistaa klusteroinnin ja korkean saatavuuden (high availability).

Lopputuloksena on asiakasyritykselle toteutettu toimiva ja uudistettu minimikontekstinhallinta. Aikaisempi toteutus käytti muistinvaraisia tietorekenteita ja konteksti oli yhden instanssin sisällä. Uudistuksen yhteydessä tietorakenteet siirrettiin muistissa olevaan tietovarastoon ja on siten klusteroitavissa.

Avainsanat
Minimikontekstinhallinta, teknologiauudistus, työpöytäintegraatio, yhteinen konteksti, Java, Wildfly, Lettuce, Redis

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Jenna Räsänen			
Title of Thesis Technology Reform of Minimum Context Management			
Date	May 17, 2020	Pages/Appendices	19
Supervisor(s)			
Client Organisation /Partners Mediconsult Ltd			

Abstract

The purpose of the thesis was to implement a technology reform of minimum context management. Minimum context management is a desktop integration designed to facilitate the simultaneous use of separate healthcare systems through a common context and single sign-on. The operation of minimum context management is based on the common context of healthcare applications defined on the basis of the CCOW standard (Clinical Context Object Workgroup), which contains a number of data processed in the same way.

The technologies used in the thesis were Java, Wildfly 16, Lettuce, Redis and Redis Sentinel. The thesis was implemented in Java version 11. Wildfly is a server for minimum context management. Lettuce is a client for Redis, which is a data structure repository where the context information is stored during a session. Redis Sentinel enables clustering and high availability.

The end result is a functional and renewed minimum context management implemented for the client company. The previous implementation used in-memory data structures and the context was within a single instance. In connection with the reform, the data structures were transferred to the in-memory data warehouse and thus can be clustered.

Keywords
Mimimum context management, technology reform, desktop integration, common context, Java, Wildfly, Lettuce, Redis

SISÄLTÖ

1 JOHDANTO

Opinnäytetyön tilaajana oli Mediconsult Oy, suomalainen sosiaali- ja tervedenhuoltoalan tietojärjestelmätoimittaja. Työn tavoitteena oli tehdä palvelinpohjaisen minimikontekstinhallinnan teknologiauudistus, ns. rewrite. Minimikontekstinhallinta on työpöytäintegraatio, jonka tehtävänä kertakirjautumisen avulla helpottaa erillisten järjestelmien yhtäaikaista käyttöä.

Opinnäytetyössä keskeisiä asioita olivat muun muassa asynkronisuus, muistinvarainen tietovarasto sekä klusteroitavuus.

Tässä opinnäytetyössä kuvaillaan HL7 Finland ry:n määritystä minimikontekstinhallinnasta. Määritys, joka tähän on kuvattu, on pelkistetty ja riisuttu versio. Tarkemman, kokonaisen määritelmän voi lukea HL7 Finland ry:n minimikontekstinhallinnan määrittelystä.

2 LYHENTEET JA MÄÄRITELMÄT

ASYNKRONISUUS	Itsenäisesti, muista toiminnoista riippumattomasti, suoritettavia toimintoja
CALLBACK	Suoritettava koodi, joka passataan argumenttina toiselle suoritettavalle koodille, jonka odotetaan kutsuvan argumenttia
CCOW	Clinical Context Object Workgroup, HL7 -standardiprotokolla, joka mahdollistaa yhtenäisen tietojen esittämistavan työpöytä- tai portaalitasolla
ECLIPSE MICROPROFILE	Perustason alustamääritelmä
HL7	Health Level Seven International, kansainvälinen, voittoa tavoittelematon kehittäjäorganisaatio, jonka missiona on tuottaa sähköisen (terveydenhuollon) asioinnin standardit
JAVA	Yleiskäyttöinen, luokkaperusteinen olio-ohjelmointikieli
JAVA EE	Ohjelmistokehitysalusta Java -sovellusten kehittämiseen ja suorittamiseen
KLUSTEROITU SOVELLUS	Usean instanssin verkkomalli, jolla taataan korkea tavoitettavuus (high availability)
LETTUCE	Lettuce on skaalautuva, säieturvallinen (thread-safe) Redis client (asiakasohjelma) joka mahdollistaa myös asynkronisuuden
MEDICONTEXT	Mediconsult Oy:n minimikontekstinhallinta
NON-BLOCKING	Operaatio, joka ei estä suorittavan säikeen etenemistä
REDIS	Avoimen lähdekoodin NoSQL tietorakenne varasto
WILDFLY	Java EE 8 sertifioitu sovelluspalvelin

3 TYÖN TILAAJA

Työn tilaajana on suomalainen, vuonna 1975 perustettu Mediconsult Oy. Mediconsult on perheyritys, joka on keskittynyt sosiaali- ja terveydenhuollon tietojärjestelmiin ja on yksi alan johtavia toimijoita Suomessa. Toimipisteitä on Suomessa viisi, Helsingissä, Salossa, Kuopiossa, Joensuussa ja Oulussa. Kuudes toimipiste sijaitsee Colombossa, Sri Lankassa. Työntekijöitä on yhteensä noin 150.

Yrityksen vahvuutena on yli 40 vuoden kokemus sosiaali- ja terveydenhuollosta. Mediconsultin asiakkaita ovat sekä julkiset että yksityiset terveydenalan ammattilaiset ja sairaanhoitopiirit. Tuotteita kehitetään jatkuvasti ja ratkaisut räätälöidään asiakkaan tarpeiden mukaiseksi. (Mediconsult Oy)

Toiminnan lähtökohtana ovat korkea laatu ja jatkuva prosessien parantaminen. Mediconsultin kaikki tuotteet ja palvelut on sertifioitu, lisäksi yritys suosii hankinnoissaan ympäristöystävällisiä palveluita ja tuotteita. (Mediconsult Oy, ei pvm)

Mediconsultin tuotteet mahdollistavat terveydenalan ammattilaisille muun muassa helpon ja turvallisen asiakastietojen kirjaamisen, reseptin kirjoittamisen sekä asiakkuudenhallinnan. Mediconsult Oy:n tuoteratkaisut liittyvät asiakkuudenhallintaan, tiedolla johtamiseen, toiminnanohjaamiseen, sähköiseen asiointiin, omahoitoon, mobiilikirjaamiseen, sähköiseen reseptiin, koulutukseen sekä konsultointiin. (Mediconsult Oy, ei pvm)

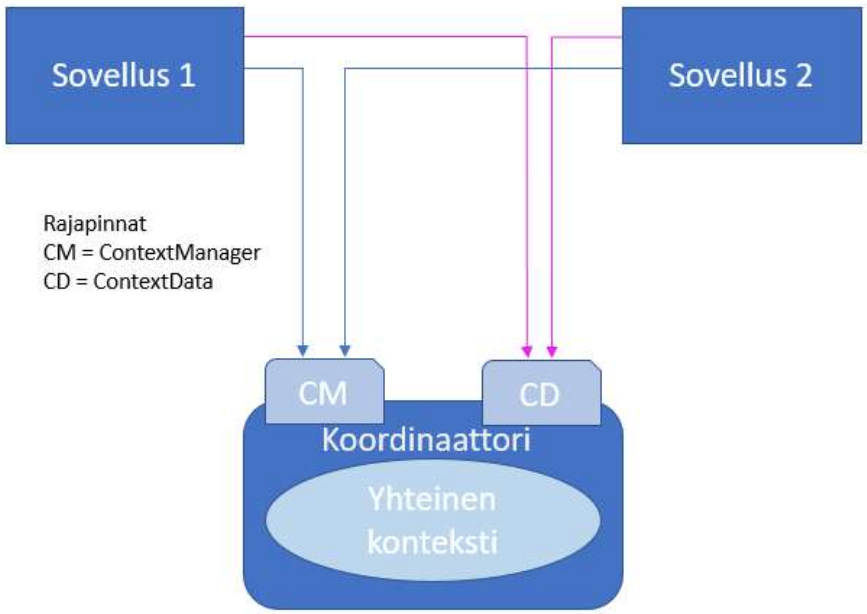
4 MINIMIKONTEKSTINHALLINTA

HL7 Finland ry on vuonna 1995 perustettu yhdistys, joka pyrkii edistämään järjestelmäintegraatio -periaatteella tapahtuvaa tietojärjestelmien kehittämistä ja terveydenhuollossa käytettäviä tietojärjestelmästandardeja. HL7 Finland ry:n tavoitteena on muun muassa ohjelmistojen saattaminen yhteensopiviksi, avoimet standardit, rajapintojen yhdenmukainen soveltaminen, tietojärjestelmien käyttäjien, toimittajien ja viranomaisten sekä standardoijien kanssa yhteistyössä toimiminen. HL7 Finland ry on jäsenenä kansainvälisessä HL7 International -järjestössä. (Porrasmaa & Kaskinen, ei pvm)

CCOW-standardin ratkaisua työpöytäintegraation toteuttamiseen on käytetty pohjana määriteltäessä minimikontekstinhallintaa. HL7 Finland ry:n tekemän määrittelyn tarkoituksena on kuvata minimiratkaisua, jolla saavutetaan CCOW-standardin mukainen perustoiminnallisuus. (Porrasmaa & Kaskinen, ei pvm)

Minimikontekstinhallinta on työpöytäintegraatio ja sen määriteltyjen toiminnallisuuksien tavoitteena on käytettävyyden parantaminen. Työpöytäintegraatio helpottaa erillisten järjestelmien yhtäaikaista käyttöä samalla työasemalla kertakirjautumisen ja järjestelmien yhteisen kontekstin avulla. (HL7 ry, 2006)

Koordinaattori on sovellus tai ohjelmistokomponentti, joka ohjaa eri sovellusten välisiä toimintoja. Kontekstinhallintaan (KUVA 1), eli koordinaattoriin, liittyneet sovellukset eivät tunne toistensa protokollia ja kaikki kanssakäyminen kulkee koordinaattorin kautta. (HL7 ry, 2006) Opinnäytetyö on tehty minimikontekstinhallinnan määrittelyversion 3.0 mukaisesti.



KUVA 1 Kontekstinhallinta

4.1 Yhteinen konteksti

Yhteinen konteksti on työasemakohtainen ja siinä ylläpidetään tietoa viimeisimmäksi sisäänkirjautuneesta käyttäjästä ja viimeisimmäksi valitusta potilaasta. Kontekstissa on mahdollisuus ylläpitää myös muita tietoja, jotka ovat hyödyllisiä terveydenhuollolle. Kontekstin toiminnan pohjana on, että yhteinen konteksti muodostuu tietojoukosta, jota voidaan tarkastella samalla tavalla riippumatta sovelluksen sisäisestä toiminnasta.

Minimikontekstinhallinnan olennaisimmat tiedot koskevat käyttäjää ja potilasta. Käyttäjän ja potilaan tietoja hyödyntämällä voidaan toteuttaa toiminnallisuus, jota kontekstiin liittyvät sovellukset hyödyntävät toteuttamalla kertakirjautumisen ja seuraamalla saman potilaan tietoja säädellysti eri ohjelmissa. (HL7 ry, 2006)

HL7 on määritellyt CCOW -standardin mukaan kontekstidatan tietotyyppejä. Opinnäytetyössä toteutettu minimikontekstinhallinta kuitenkin käsittelee saapuvan datan merkkijonona, joka vastaanotetaan, käsitellään ja palautetaan vastaanotetussa muodossa.

4.2 Rajapinnat ja metodit

Minimikontekstinhallinnassa on kaksi rajapintaa, jotka ovat ContextManager ja ContextData. ContextManager sisältää kontekstin luomiseen, kontekstiin liittymiseen ja siitä poistumiseen tarvittavat metodit, *createSession*, *joinCommonContext* ja *leaveCommonContext.* ContextData sen sijaan pitää sisällään kontekstin sisällön käsittelemiseen tarvittavat *getItemValues* ja *setItemValues* -metodit.

4.3 Kontekstiin liittyminen

Minimikontekstinhallinnan kontekstiin liitytään ContextManager:n kautta. Pyynnössä välitetään parametrina liittyvän sovelluksen nimi. Ensimmäisenä kontekstiin liittynyt sovellus laukaisee session luonnin. Sessiolle generoidaan avain, jonka avulla sallitut sovellukset voivat liittyä saman session kontekstiin. Jokainen liittyvä sovellus saa lisäksi myös osallistujanumeron, jonka avulla yksilöidään liittynyt sovellus kyseisessä sessiossa.

HL7 ry:n määritysten mukaisesti, kontekstiin saa liittyä vain sallitut sovellukset. Luotetut ja sallitut sovellukset voidaan määritellä organisaatiokohtaisesti. Vain luotetut sovellukset voivat asettaa käyttäjätunnuksen kontekstiin. Kaikki luotetut sovellukset ovat sallittujen sovellusten listauksessa, mutta kaikki sallitut sovellukset eivät ole luotettuja.

4.4 Kontekstista poistuminen ja kontekstin tuhoaminen

Sovelluksen poistuessa kontekstista, se tunnistetaan osallistujanumeron avulla. Minimikontekstinhallinnan muistivarastossa on ajantasainen tieto sovelluksista, jotka ovat eri konteksteihin liittyneet. Jos

kontekstin viimeinenkin sovellus poistuu, kontekstin sisältö tuhotaan, sillä sitä on turha enää ylläpitää. Kontekstin sisältö tuhotaan myös silloin, jos käyttäjätunnuksen asettanut sovellus poistuu tai jos aikakatkaisun raja on tullut vastaan.

4.5 Subjekti

Yhteisen kontekstin sisältö on ryhmitelty tietokokonaisuuksiin, joita sanotaan subjekteiksi. Kaikissa subjekteissa on tietojoukko, joka muodostuu nimi-arvo-pareista. Tällainen nimi-arvo-pari voi olla esimerkiksi käyttäjästä oleva tieto (KUVA 2). (HL7 ry, 2006)

KUVA 2 Käyttäjäsubjektista löytyvä tieto, joka vastaa käyttäjätunnusta

4.5.1 Subjektin nimeäminen

Kaikilla subjekteilla on yhteinen rakenne (KUVA 3). Subjektin osat erotetaan toisistaan pisteillä. *Subject_label* osoittaa subjektin, jolle tieto kuuluu, esimerkiksi User. *Role* ilmaisee subjektin toimenkuvan. Rooleiksi on määritelty *Id,* tieto, jota käytetään tunnistamiseen, esimerkiksi uniikki käyttäjätunnus, *Co*, *Id.*tä vahvistavaa tietoa, joka on yhteydessä *Id.*n tietoon, esimerkiksi käyttäjän etunimi ja *An,* lisätieto, joka on riippuvainen jostakin id-tyypin tiedosta. *Name_prefix* on tiedon varsinainen nimi, esimerkiksi Logon, joka on kirjautumistieto. *Optional_name_suffix* on valinnainen tieto, joka toimii varsinaisen tiedon lisätarkentimena, mikä taas mahdollistaa sen, että useampi kuin yksi tieto voi kuvata samaa käsitettä. (HL7 ry, 2006)

KUVA 3 Subjektin syntaksi

4.5.2 Subjektien käyttö

HL7 ry:n minimikontekstinhallinnan uusimpien määrittelyjen mukaan jokaisella subjektilla täytyy olla pakollinen id-tieto. Medicontextissa vain *User-* ja *Patient* -subjekteilla id-tieto on pakollinen, *User.Id.Logon* ja *Patient.Id.NationalIdNumber*.

4.6 Kontekstinhallinnan tietoturva

Tietoturva on toteutettu osin subjektien ja sovellusten hallinnalla ja käsittelyillä, sessioavaimen yksilöllisyydellä ja erillisen välitysohjelman avulla. Välitysohjelma mahdollistaa yhteyden muodostamisen

kontekstinhallintaan ilman sovellussertifikaattien jakelua työasemille. Välitysohjelma käyttää kaksisuuntaista TLS-protokollaa (Transport Layer Security Protocol) https-protokollan (Hypertext Transfer Protocol Secure) alla, jonka avulla kontekstiin liittyvät sovellukset voivat varmistaa siirrettävän tiedon eheyden ja salata sen.

Https on yhteysprotokolla, joka suojaa käyttäjän tietokoneen ja sivuston välisten tietojen eheyttä ja luottamuksellisuutta. HTTPS:llä lähetetyt tiedot suojataan nykyisin TLS-protokollalla, joka tarjoaa kolmenlaista avaintason suojausta, salauksen, tietojen eheyden ja todennuksen. (Google, 2020)

5 TEKNOLOGIAUUDISTUS

Mediconsult Oy:n vanha toteutus minimikontekstinhallinnasta on toteutettu Java 5 ja palvelimena on toiminut Apache Tomcat 5.0.19. Uusi minikontekstinhallinta haluttiin klusteroitavaksi. Aikaisemmin konteksti on ollut muistinvaraisesti erilaisissa tietorakenteissa yhden instanssin sisällä, joten tiedon klusterointi ei ole ollut mahdollista. Uudesta minimikontekstinhallinnasta haluttiin myös tilaton (stateless).

Uuden minimikontekstinhallinta toteutettiin Java 11 ja palvelinratkaisuksi päädyttiin Wildfly 16, jonka etuina ovat Eclipsen microprofiilit ja hyödylliset Java EE (Enterprise Edition) -ominaisuudet, kuten JAX-RS, joka on yksi Java EE:n tarjoamista standardeista.

Java EE sovellusalusta on rakennettu Java SE (Standard Edition) alustan päälle. Java EE tarjoaa rajapinnan ja ajonaikaisen ympäristön suurten, monikerroksisten, skaalautuvien, luotettavien ja suojattujen verkko sovellusten kehittämiseen ja käyttämiseen. (Oracle, 2012)

Eclipsen mikroprofiilit perustuvat Java EE standardeihin ja niiden suunnitteluperiaatteisiin. Mikroprofiilit laajentavat standardeja toiminnallisuuksilla, jotka eivät kuulu Java EE:hen. Tällaisia toiminnallisuuksia ovat esimerkiksi joustavuus, seuranta ja konfigurointi. (Daschner, 2019)

Uudistetussa minimikontekstinhallinnassa erilaisten tietorakenteiden sijasta konteksti säilötään Redikseen, joka tarjoaa yksinkertaisen ja nopean rajapinnan, helposti konfiguroitavan korkean saatavuuden (high availability) sen klusterointimahdollisuudella sekä viansiedolla. Varsinaista tietokantaa ei tarvittu, sillä tiedon pysyvyys (persistenssi) ei ole olennaista. Redis kuitenkin tukee persistenssiä ja klusteroitua minimikontekstinhallintaa käytettäessä, persistenssi on olennainen.

Asiakassovellus ottaa yhteyden minimikontekstinhallintaan HTTP kutsun avulla. Minimikontekstinhallinta on konfiguroitavissa jokaisen asiakkaan tarpeiden mukaisesti. Esimerkiksi eri organisaatioissa minimikontekstinhallinnan määrittelyt luotetuista ja sallituista sovelluksista sekä subjektien riippuvuudet voivat poiketa.

Tietojen eheyttä vaativat toiminnot, kuten kontekstiin liittyminen, tietojen asettaminen tai hakeminen suoritetaan transaktioissa, joten ne käyttävät samaa tietokantayhteyttä, joka transaktion suorituksen jälkeen palautetaan connection pool:iin.

Connection pooling parantaa sovelluksen suorituskykyä, kun yhteyttä ei tarvitse muodostaa jokaista transaktiota varten uudestaan. Lettucen yhteydet on suunniteltu säieturvallisiksi, joten yksi yhteys voidaan jakaa useiden säikeiden kesken. (Connection Pooling 5.1, 2018)

5.1 Kontekstin luominen

Kun minimikontekstinhallinta saa pyynnön luoda konteksti (KUVA 4), pyynnössä on parametreina URL, jota pyynnössä käytettiin ja ip-osoite, josta pyyntö tuli. Kontekstia luodessa generoidaan jokaiselle kontekstille yksilöllinen kontekstiavain. Tietovarastoyhteys aukeaa ja konfiguroinnista riippuen, minimikontekstinhallinta klusteroidaan. Tietovarastosta tarkistetaan, ettei samanlaista kontekstiavainta ole jo valmiina. Kontekstiavaimen tietoihin tallennetaan ip-osoite sekä aikaleima (timestamp).

```
http://localhost:8088/Medicontext/cm?method=CreateSession&interface=ContextManager
```

KUVA 4 Pyyntö kontekstin luomiseen

5.2 Liittyvän sovelluksen yksilöinti

Sen jälkeen, kun konteksti on luotu, muut sovellukset voivat liittyä samaan kontekstiin kontekstiavaimella. Liittymiskutsussa (KUVA 5) parametereina on sovelluksen nimi, ip-osoite sekä avain kontekstiin, johon halutaan liittyä. Jokainen liittyvä sovellus tulee olla listattuna sallittuihin sovelluksiin.

Liittyvälle sovellukselle luodaan yksilöllinen kuponkitunnus sekä tarkistetaan, että onhan konteksti, johon yritetään liittyä, olemassa. Kuponkitunnuksen luomisen epäonnistuessa yhteentörmäykseen, yritetään kuponkitunnuksen luomista uudestaan korkeintaan viisi kertaa. Yhteentörmäyksellä tarkoitetaan tässä yhteydessä sitä, että liittyvälle sovellukselle luotu kuponkitunnus on jo olemassa.

Ennen kuponkitunnuksen lisäämistä tietovarastoon, tarkistetaan ettei samanniminen sovellus ole jo liittynyt ja että tunnusta ei vielä ole tietovarastossa. Kun sovellus on saanut kuponkitunnuksen, sitä käytetään jokaisessa kutsussa, jotta lähettäjä voidaan tunnistaa.

```
http://localhost:8088/Medicontext/cm?method=JoinCommonContext&interface=ContextManager&applicationName=
Mediatri&SessionKey=MCM:874887921DB0613F9A60D8ABE31F037EC1W81AC@K67UZEAS
```

KUVA 5 Pyyntö liittyä kontekstiin

5.3 Tietojen asettaminen ja hakeminen

Kun tietoja asetetaan kontekstiin, pyynnössä on parametreina kuponkitunnuksen lisäksi lisättävät nimi-arvo-parit. Ennen kuin tiedot lisätään tietovarastoon, niistä tarkistetaan muun muassa, että ne ovat oikean muotoisia (KUVA 6) ja että subjektista riippuvia tietoja ei aseteta ilman subjektin id:tä. Vain luotetut sovellukset voivat asettaa käyttäjätunnuksen.

Tietoja haettaessa, pyynnössä on kuponkitunnus sekä haettavien tietojen nimet (itemNames). Jos haettavaa tietoa ei löydy, myöskään tiedon nimi ei palaudu vastauksessa.

```
private static final Pattern ITEMNAMEPATTERN = Pattern.compile(
    "^(\\[\\w+(\\.\\w+)*\\])?" + // optional Subject descriptor, ex. "[hl7.org]"
        "(\\w+)" + // Subject, ex. "User"
        "\\.([iI][dD]|[cC][oO]|[aA][nN]|[iI][nN]|[oO][uU]|[tT][oO])" + // dot and Role, ex. ".Id"
        "\\.(\\[\\w+(\\.\\w+)*\\])?" + // dot and optional Name_prefix descriptor, ex. ".[hl7.org
        "(\\w+)" + // then Name_prefix, ex. "Logon"
        "(\\.(\\w+))?$"); // optional Name_suffix, ex. "Mediatri"
```

KUVA 6 Nimi-arvo-parin muodon tarkistus

5.4 Kontekstista poistuminen

Kun sovellus poistuu kontekstista, tarkistetaan kontekstin aktiiviset sovellukset ja onko poistuja kontekstin viimeinen osallistuja. Sovellus voi poistua kontekstista myös aikakatkaisun myötä.

5.5 Optimistinen lukitus

Minimikontekstinhallinnassa käytetään optimistista lukitusta, jolla estetään transaktion aikana tietovaraston tiedon tai avaimen muuttuminen. Rediksessä 'MULTI' -käsky aloittaa transaktio -ketjun. Kun 'MULTI' on kutsuttu, 'EXEC' -käsky suorittaa jonossa olevat komennot.

Optimistisen lukituksen avulla merkitään ne tietokannan avaimet ja kentät, jotka ovat kunkin transaktion kannalta olennaisimmat. Esimerkiksi kontekstin tietoja hakiessa merkitään kuponkitunnuksen tietokanta-avain tietoja hakevalta sovellukselta. Tällä tavoin voidaan varmistaa, ettei tietoja kysyvän sovelluksen kuponkinumeron tiedot tietovarastossa vaihdu tai poistu kesken transaktion. 'EXEC' -käskyn yhteydessä tarkistetaan, ettei merkittyjä tietoja ole muutettu merkitsemisen ja 'EXEC' -käskyn välissä.

5.6 Ajastimet

Uudistetussa minimikontekstinhallinnassa käytetään kahta ajastinta, jotka etsivät mahdollisesti poistettavia tietokannan avaimia ja tietoja. Ensimmäinen ajastin käy tietokantaa läpi joka kolmas minuutti etsien sovelluksia, jotka eivät ole olleet yhteydessä minimikontekstinhallintaan tietyn ajan kuluessa. Toinen ajastin käy tietokannan läpi kerran vuorokaudessa vapauttaen käyttämättömiä kuponkinumeroita.

```
@Schedule(hour = "2") //käynnistyy kerran päivässä klo.0200
public void removeUnusedKeys() {

  Range<Number> olderThan24h = Range.<Number> unbounded().lt(Instant.now().minus(24, ChronoUnit.DAYS).toEpochMilli());
  //hakee lotterysetin kupongit
  Set<String> lotterySet =
      new HashSet<>(redisManager.getCommands().zrangebyscore("lotterySet", olderThan24h).toCompletableFuture().join());
  //hakee coupons. -alkuiset avaimet
  List<String> couponsKeys = RedisOperations.scanKeys(redisManager.getCommands(), "coupons.*").toCompletableFuture().join();

  for (String couponKey : couponsKeys) {
    //tarkistetaan onko kuponki lotterySet:ssä, jos on, poistetaan couponsLististä
    if (lotterySet.contains(MedicontextUtils.extractParticipantCoupon(couponKey))) {
      couponsKeys.remove(couponKey);
    }
  }
  //couponListiin jäljelle jääneet poistetaan tietokannasta (=jo poistuneiden sovellusten kupongit)
  redisManager.getCommands().srem("lotterySet", (String[]) couponsKeys.toArray());

}
```

KUVA 7 Ajastin, joka vapauttaa käyttämättömät kuponkinumerot

5.7 CompletionStage

Koska Lettuce on oletuksena asynkroninen, non-blocking sekä hyödyntää CompletionStagea, oli työssä luonnollista käyttää Javan CompletableFuture -luokkaa. CompletionStage on Javan vastine JavaScriptin Promiselle. Promise on objekti, joka edustaa asynkronisen toiminnan mahdollista onnistumista tai epäonnistumista. Jotta asynkronisuus säilyisi ja välittyisi myös Wildflyhin, on tärkeää, että koko sovellus on tehty asynkroniseksi.

Java 8:ssa on CompletableFuture -luokka, joka toteuttaa Future ja CompletionStage -rajapinnat. CompletionStage rajapinnan avulla asynkronisia operaatioita voidaan ketjuttaa muiden asynkronisten käsittelyjen kanssa. CompletableFuture -luokassa on noin 50 eri metodia asynkronisten operaatioiden suorittamiseen, yhdistämiseen ja virheiden käsittelyyn. (Baeldung, 2020)

CompletionStage on lupaus siitä, että operaatio suoritetaan loppuun. Callbackien, esimerkiksi *thenAccept()*, avulla voidaan määritellä mitä tapahtuu, kun asynkronisesti toimiva koodi on suoritettu ilman, että vastausta odotetaan valmiiksi. Exceptionally -haara sisältää vaihtoehtoisen käsittelyn epäonnistuneelle operaatiolle. (Dead Code Rising, 2018)

Alla olevassa kuvassa (KUVA 8), on esimerkki siitä, kuinka CompletionStage -rajapintaa hyödynnetään asynkronisessa minimikontekstinhallinnassa. Ensimmäiseksi luodaan uusi instanssi CompeletableFuture -luokasta, sen jälkeen kutsutaan *contextManager* instanssin *leaveCommonContext* -metodia, jonka vastauksena palautuu *participantCoupon*. Jos ketjutetut operaatiot onnistuvat, futuuri saa vastauksen valmiiksi *completableFuture.complete* -metodilla ja kuponkitunnus poistetaan kontekstista. Jos taas operaatiot epäonnistuvat, futuurin vastaus saatetaan virheellisesti valmiiksi *completableFuture.completeExceptionally* -metodilla arvolla *error*. *Error* -vastaus sisältää tiedon siitä, mikä epäonnistui.

```
private CompletionStage<String> processLeaveCommonContext(String coupon) {

   CompletableFuture<String> completableFuture = new CompletableFuture<>();
   contextManager.leaveCommonContext(coupon).thenAccept(participantCoupon -> {
     completableFuture.complete("kuponki poistettu: " + participantCoupon);
   })
       .exceptionally(error -> {
         log.error("Tietojen poistaminen tietokannasta epäonnistui: " + error);
         completableFuture.completeExceptionally(error);
         return null;
       });

   return completableFuture;
 }
```

KUVA 8 Kontekstista poistuminen

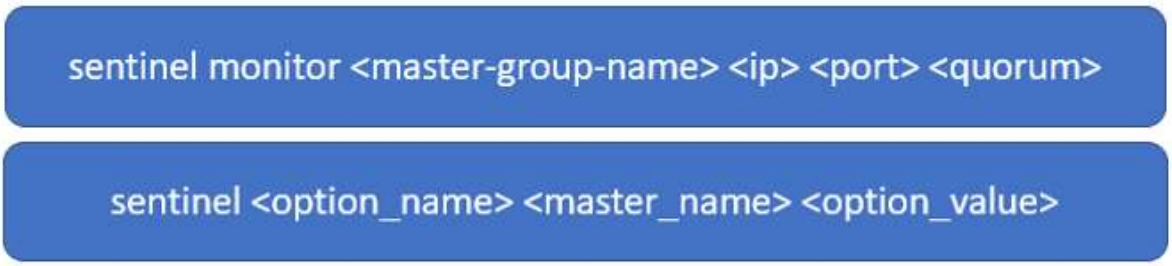
5.8 Redis Sentinel

Redis Sentinel helpottaa Redis -instanssien hallinnoimista. Sentinel muun muassa tarkkailee master ja replika -instansseja ja ilmoittaa jos jossakin tarkkailussa olevista instansseista on vikaa. Jos master ei toimi odotetusti, Sentinel aloittaa vikasieto prosessin, jossa replika ylennetään masteriksi ja muut replikat ohjautuvat käyttämään uutta masteria. Kun asiakasohjelma (client) ottaa yhteyden Redikseen, Sentinelin tehtävä on ilmoittaa olemassa olevan masterin osoite. (RedisLabs, 2020)

Sentinel käyttää käynnistyessään asetustiedostoa (configuration file) johon se tallentaa järjestelmän sen hetkisen tilan. Uudelleen käynnistymisen yhteydessä asetustiedostossa olevat tiedot noudetaan uudelleen. Sentinel ei käynnisty, jos asetustiedostoa ei ole määritelty. Sentinelit käyttävät oletuksena TCP porttia 26379 vastaanottamaan yhteyksiä muiden Sentineleiden IP-osoitteista. Jos oletusportti ei ole auki, mahdollinen vikasieto prosessi ei käynnisty. (RedisLabs, 2020)

Sentinel tarvitsee toimiakseen asetustiedoston lisäksi vähintään kolme Sentinel instanssia. Rediksen ja Sentinelin hajautettu järjestelmä ei takaa, Rediksen asynkronisesta replikoinnista johtuen, että kirjoitetut tiedot pystyttäisiin säilyttämään mahdollisen vian aikana. (RedisLabs, 2020)

Asetustiedostossa määritellään masterit, joita valvotaan ja jokaiselle masterille annetaan eri nimi. Replikat löydetään automaattisesti, joten niitä ei tarvitse asetustiedostossa määritellä. Sentinel täydentää asetustiedoston replikan tiedoilla, jotta tieto säilyisi uudelleen käynnistymisen yhteydessä. Tiedosto kirjoitetaan uudelleen joka kerta, kun uusi replika ylennetään masteriksi tai kun uusi Sentinel on havaittu. Mastereiden lisäksi asetustiedostossa voidaan määrittää myös muita asetuksia, esimerkiksi down-after-milliseconds ja failover-timeout. (RedisLabs, 2020)



KUVA 9 Sentinelin konfiguraation syntaksi

6 POHDINTAA

Aiheena minimikontekstinhallinta oli minulle aivan uusi ja pääsin lopputyössäni käyttämään itselleni uusia tekniikoita. Koska minimikontekstinhallinnan teknologiauudistuksessa harppaus uusiin tekniikoihin ja ohjelmointikielen uuteen versioon oli sen verran iso, ettei vanhan toteutuksen koodia voinut juurikaan enää hyödyntää, käytetyt ratkaisut sovellettiin uusiksi.

Opiskelujen aikana asynkronisuus ja NoSQL -tietokanta jäivät etäisiksi termeiksi. Opinnäytetyössäni pääsin kuitenkin näihin tutustumaan käytännössä. Opinnäytetyöni eteni alkuvaiheessa aikataulun mukaisesti, mutta loppua kohden aikataulu venyi itsestäni riippumattomista syistä. Tällä hetkellä uudistettu minimikontekstinhallinta odottaa testaamista.

Työskentelyssä ehdottomasti mielenkiintoisinta oli päästä toteuttamaan oikealle yritykselle oikea, käytettävä sovellus. Opinnäytetyön lopputulos tulee olemaan osana jokaista Mediconsultin tuotetta helpottamassa kertakirjautumisen avulla eri sovellusten kommunikointia ja tiedon jakamista kontekstissa.

Ammatillisesti työn aihe ja toteuttaminen lisäsivät valmiuksiani työelämään. Työn edetessä sain ohjausta ja tukea asiantuntevalta ohjelmistoarkkitehdiltä.

7 LÄHDELUETTELO

Baeldung. (13. 12. 2019). *Building Microservices with Eclipse MicroProfile*. Haettu 16. 3. 2020 osoitteesta Baeldung: https://www.baeldung.com/eclipse-microprofile

Baeldung. (12. 2. 2020). *Guide To CompletableFuture*. Haettu 26. 2. 2020 osoitteesta Baeldung: https://www.baeldung.com/java-completablefuture

Connection Pooling 5.1. (17. 9. 2018). Haettu 21. 5. 2020 osoitteesta github.com: https://github.com/lettuce-io/lettuce-core/wiki/Connection-Pooling-5.1

Daschner, S. (2019). *Java EE, Jakarta EE, MicroProfile, or Maybe All of Them*. Haettu 21. 5. 2020 osoitteesta eclipse.org: https://www.eclipse.org/community/eclipse_newsletter/2019/february/Jakarta_Micro_All.php

Dead Code Rising. (29. 5. 2018). *Java 8: Writing asynchronous code with CompletableFuture*. Haettu 26. 2. 2020 osoitteesta Dead Code Rising: https://www.deadcoderising.com/java8-writing-asynchronous-code-with-completablefuture/

Google. (2020). *Secure your site with HTTPS*. Haettu 21. 5. 2020 osoitteesta https://support.google.com/webmasters/answer/6073543?hl=en

HL7 ry. (31. 1. 2006). *Minimikontekstinhallinnan määrittely.* Haettu 15. 1. 2020 osoitteesta HL7 FINLAND: http://www.hl7.fi/hl7-rajapintakartta/minimikontekstinhallinnan-maarittely/

Mediconsult Oy. (ei pvm). Esittely. Haettu 25. 2. 2020

Mediconsult Oy. (ei pvm). *Ratkaisut*. Haettu 25. 2. 2020 osoitteesta Mediconsult Oy: https://www.mediconsult.fi/ratkaisut

Mediconsult Oy. (ei pvm). *Toimintatapamme*. Haettu 25. 2. 2020 osoitteesta Mediconsult Oy: https://www.mediconsult.fi/yritys/toimintatapamme

Oracle. (4. 2012). *Your First Cup: An Introduction to the Java EE Platform*. Haettu 21. 5. 2020 osoitteesta docs.oracle.com: https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html

Porrasmaa, J.;& Kaskinen, T. (ei pvm). *HL7 Finland ry*. (HL7 ry) Haettu 13. 1. 2020 osoitteesta HL7 Finland: http://www.hl7.fi/esittely/

RedisLabs. (28. 1. 2020). *Redis Sentinel Documentation*. Haettu 28. 1. 2020 osoitteesta Redis: https://redis.io/topics/sentinel