Opinnäytetyö AMK

Tieto- ja viestintätekniikka

2022

Leevi Susila

INTEGRAATIOHÄLYTYKSIEN HALLINNAN AUTOMATISOINTI

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö AMK | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja Viestintätekniikka

2022 | 35 sivua

Leevi Susila

INTEGRAATIOHÄLYTYKSIEN HALLINNAN AUTOMATISOINTI

Suomen terveydenhuollossa on käytössä useita erilaisia tietojärjestelmiä. Modernin terveydenhuollon toiminnan takaamiseksi tulee näiden järjestelmien välille kehittää järjestelmäintegraatiota, jotka ovat kestäviä ja joiden häiriötilanteisiin voidaan reagoida nopeasti. Tämän opinnäytetyön tavoitteena oli kehittää automatisoitu ratkaisu useiden järjestelmäintegraatiotuotantojen erilaisten toiminnallisten moduulien tuottamien hälytysten yhtenäiseen käsittelyyn. Suomen terveydenhuollon IT-järjestelmien nykytilanne on monimutkainen ja tämän tilanteen hallitsemiseksi vaaditaan useita työkaluja ja protokollia saumattoman dataintegraation luomiseen. Työssä esitellään suunniteltua ja toteutettua hälytyksen käsittelyratkaisua jatkokehitysehdotuksineen.

Automaatioratkaisu tehtiin toimeksi antaneen yrityksen jo olemassa oleville IT-alustoille, Intersystems IRIS for Health -tietoalustalle ja Efecte IT Service Management -alustalle. Tiedonsiirto ja hälytysten käsittely on kehitetty Caché ObjectScript -ohjelmointikielellä.

Opinnäytetyön aikana luotu hälytyskäsittelijä saatiin valmiiksi suunnitellusti ja mukautettiin osaksi yrityksen aktiivista hälytystenkäsittelyratkaisua lisäämällä se toimimaan aikaisemman ratkaisun rinnalle. Tällä sovituksella kehitettyä ratkaisua voidaan seurata ja kehittää edelleen, kunnes tarvittavat toiminnallisuudet saadaan valmiiksi ja vanha ratkaisu voidaan korvata uudella.

Asiasanat:

integraatio, REST, ohjelmointirajapinta, Caché ObjectScript, Intersystems, tiedonsiirto

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2022 | 35 pages

Leevi Susila

AUTOMATION OF INTEGRATION ALERT HANDLING

There are several different information systems in use in Finnish healthcare. In order to guarantee the operation of modern healthcare, system integrations must be developed between these systems that are sustainable. The aim of this thesis was to develop an automated solution for the unified handling of alarms generated by different functional modules of several system integration productions. The current situation of Finnish healthcare IT systems is complex and several tools and protocols are needed to create a seamless data integration to manage this situation. The paper presents a planned and implemented alarm handling solution with further development proposals.

The automation solution was implemented for the commissioning company's existing IT platforms, Intersystems IRIS for Health and Efecte IT Service Management. Data transfer and alarm handling have been developed on the Caché ObjectScript programming language.

The alarm handler created during the thesis was completed as planned and adapted into the company's active alarm handling solution by adding it to work alongside the previous solution. The solution developed with this adaptation can be monitored and further developed until the necessary functionalities are completed and the old solution can be replaced with a new one.

Keywords:

integration, REST, programming interface, Caché ObjectScript, Intersystems, data transfer

Sisältö

Käytetty sanasto

API	Sovellusohjelmointirajapinta (Lane, 2019)
FHIR	HL7 standardi. Fast Healthcare Interoperability Resources.(The Office of the National Coordinator for Health , 2019)
HL7	Kokoelma kliinisen datan tiedonsiirtostandardeja. Health Level Seven. (HL7 International, 2019)
ITIL	Information Technology Infrastructure Library (IBM Cloud Education, 2019)
JSON	Standardoitu datanvaihtoprotokolla. JavaScript Object Notation.(Ecma International, 2017)
REST	Representational state transfer (Fielding R. T., 2000)
SOTE	Sosiaali- ja terveydenhuolto
TLS	Salausprotokolla salattujen tiedonsiirtolinkkien muodostamiseksi. Transport Layer Security.(SSL-tukitiimi, 2021)
XML	Standardoitu datanvaihtoprotokolla. Extensible Markup Language.(Safris, 2021)

1 Johdanto

Suomen terveydenhuollossa on käytössä monia eri järjestelmiä niiden käyttötarpeesta ja järjestelmää käyttävästä terveydenhuollon järjestäjästä riippuen. Näiden järjestelmien tuottaman datan yhteen keräämiseksi ja hyödyntämiseksi, tulee järjestelmien välille toteuttaa järjestelmäintegraatioita. (Lehto & Neittaanmäki, 2017) Jotta järjestelmäintegraatioita toteuttava taho pystyy varmistamaan toimintansa tehokkuuden, tulee integraatioissa ilmeneviin häiriöihin olla mahdollisuus reagoida nopeasti terveydenhuollon palveluiden toiminnan takaamiseksi.

Tämä opinnäytetyö on toteutettu 2M-IT:lle, joka toteuttaa ICT-palveluita 16 tulevan hyvinvointialueen sisällä järjestelmäintegraatiopalveluiden ollessa yksi keskeisitä palveluista (2M-IT Oy, 2022). Integraatioiden häiriötilanteisiin nopeasti reagoimisen mahdollistamiseksi, opinnäytetyössä toteutetaan integraatiohälytyksien käsittelyn ja luokittelun automaatiototeutus toimeksiantajan valmista integraatioalustaa hyödyntäen.

Nykyisessä toimeksiantajan hälytyksienkäsittelyprosessissa hälytykset siirtyvät sähköpostiviesteinä toimeksiantajan IT-pavelunhallintajärjestelmään, jossa viestit vaativat vielä manuaalista käsittelyä ennen kuin ne voidaan ratkaista tai siirtää eteenpäin häiriöstä vastaavalle taholle. Opinnäytetyössä tarkoituksena on toteuttaa ratkaisu, joka poistaa tai vähentää manuaalisen hälytyksien käsittelyn tarvetta ja vapauttaa integraatioista vastaavan henkilöstön resursseja tärkeämpiin tehtäviin. Toteutuksessa hyödynnetään toimeksiantajan valmiiksi käytössä olevia työkaluja ja järjestelmiä, kuten Intersystems IRIS datankäsittelyalustaa, Caché ObjectScript -ohjelmointikieltä ja Efecte IT-palvelunhallintajärjestelmää. Hälytystietojen siirtämisessä käsittelyprosessin osilta toisille hyödynnetään tarkoitusta varten kehitettyjä HTTP-rajapintoja.

Tämän opinnäytetyön teoriaosuudessa esitellään ensin Suomen nykyinen terveydenhuollon IT-infrastruktuurin rakenne, ja tätä kautta osoittamaan järjestelmäintegraatioiden tärkeys. Lisäksi keskitytään esittelemään varsinaisen toteu-

tuksen kannalta tärkeimmät datan välitys- ja käsittelyprotokollat sekä toteutuksessa hyödynnettävät toimeksiantajan käytössä olevat tiedonhallintajärjestelmät. Lopuksi esitellään lopullinen toteutettu ratkaisu sekä paneudutaan ratkaisun arviointiin ja kehitysmahdollisuuksiin.

2 Tietojärjestelmät terveydenhuollossa

Suomen sosiaali- ja terveydenhuollon järjestäminen on jaettu perusterveydenhuoltoon ja erikoissairaanhoitoon (Rissanen & Lammintakanen, 2017). Suomessa on 21 sairaanhoitopiiriä, jotka vastaavat jäsenkuntiensa erikoissairaanhoidon järjestämisestä (Terveysministeriö, 2022). Perusterveydenhuollosta ja sosiaalihuollosta vastaavat vuoden 2023 alusta eteenpäin sosiaali- ja terveydenhuollon uudet hyvinvointialueet. Tätä ennen vastuu järjestämisestä on ollut yksittäisillä kunnilla tai kuntayhtymillä. (Valtioneuvosto, 2021)

Tämä terveydenhuollon palveluiden järjestämisvastuiden jakaminen ja jatkuva hallinnollisten ratkaisujen kehittäminen on johtanut tilanteeseen, jossa Suomessa on syntynyt useita itsenäisiä palvelujärjestelmiä terveydenhuollon järjestämiseksi. Vaikka kehitys kohti yhtenäisempää palvelujärjestelmää, esimerkiksi sote-uudistuksen avulla, on viime vuosina ollut jatkuvaa, Suomessa ollaan edelleen tilanteessa, jossa informaation ja tiedon liikkuminen edellä mainittujen eri toimielimien välillä ei ole aina täysin mutkatonta. Kun tähän lisätään vielä yksityisen terveydenhuollon osuuden kasvaminen ja näiden omat toimintajärjestelmät, päädytään tilanteeseen, jossa informaation ja tiedon saumattoman liikkumisen tärkeys kasvaa. (Rissanen & Lammintakanen, 2017)

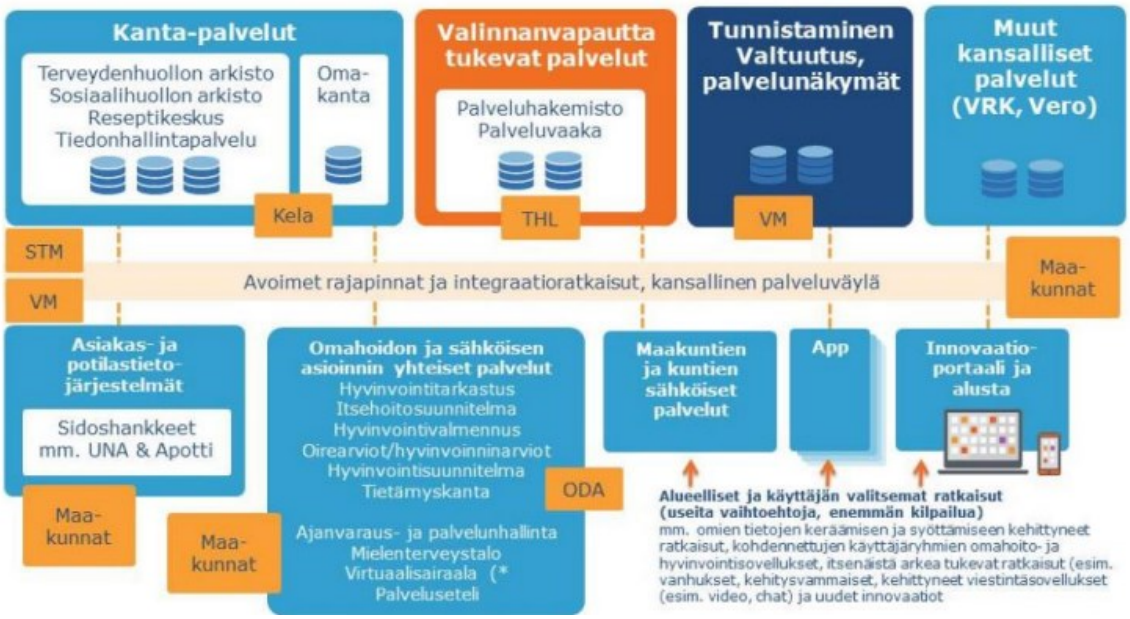
2.1 Terveydenalan järjestelmät ja menetelmät Suomessa

Valviran tietojärjestelmärekisterin mukaan Suomessa on rekisteröitynä noin 80 Sosiaali- ja terveydenhuollon tietojärjestelmää. Näillä tarkoitetaan asiakastietojen sähköisen käsittelyn järjestelmiä (Valvira, 2022). Järjestelmien lukumäärä kasvaa entisestään, kun huomioidaan, että samasta järjestelmästä voi olla käytössä useita eri versioita, joiden tietovarannot eivät aina ole täysin yhteensopivia keskenään. (Lehto & Neittaanmäki, 2017)

Todellinen sote-tietojärjestelmien lukumäärä Suomessa on noin 400–800 järjestelmää sairaanhoitopiiriä kohden. Tähän lukuun on tietojärjestelmien lisäksi lisätty eri erikoisalojen järjestelmät sekä muut ohjelmistot ja järjestelmät, jotka

tuottavat dataa mutta eivät säilö sitä, eivätkä siten täytä Valviran tietojärjestelmän määrityksiä. (Lehto & Neittaanmäki, 2017)

Uudistuksista ja yhtenäistämisestä huolimatta tulee Suomen terveydenhuollon järjestelmien määrä pysymään todennäköisesti noin 200 järjestelmässä per palvelukokonaisuus (sairaanhoitopiiri & hyvinvointialue). Erikoisalojen järjestelmien ja tietojärjestelmien sekä Suomen sote-tietojärjestelmäkokonaisuuden (Kuva 1) tiedonsiirron mahdollistamiseksi on järjestelmäintegraatioiden tuottaminen ja ylläpitäminen jatkuvasti tarpeellista. (Lehto & Neittaanmäki, 2017)



Kuva 1. sote-tietojärjestelmäratkaisukokonaisuudet Suomessa ja näiden tiedonvälityksen kuvaus. (Lehto & Neittaanmäki, 2017)

2.2 Järjestelmäintegraatiot terveydenhuollossa

Järjestelmäintegraatiolla tarkoitetaan datan siirron mahdollistamista kahden tai useamman eri tietojärjestelmän tai ohjelmiston välillä (Haglund, 2018). Tietojärjestelmän rakenteesta riippuen integraatio saattaa vaatia datan muokkausta

datan siirtymisen aikana, jotta kohdejärjestelmä pystyisi ymmärtämään lähdejärjestelmän toimittamaa dataa. Järjestelmäintegraatio voi pitää sisällään monta yksittäistä integraatiota, jossa tieto liikkuu järjestelmästä tai järjestelmän osasta toiseen, tai se voi tarkoittaa tiedon siirtymistä yhdestä lähteestä useampaan eri kohdejärjestelmään (Altexsoft, 2021.). Tiedon liikkuminen järjestelmien välillä voi olla myös kaksisuuntaista. Integraatioiden avulla luodaan siis kahdesta erillisestä järjestelmästä yksi yhteinen suurempi toimiva kokonaisuus (Haglund, 2018).

Tietojärjestelmien integroinnissa tärkeitä määriteltäviä osia ovat tiedonsiirron vaatimat toiminnallisuudet eli protokollat, formaatit ja rajapinnat sekä käytettävän integrointimenetelmän ja järjestelmän valinta (Umapathy, Purao, & Barton, 2008). Tarvitaan siis tieto siitä, mitä lähetetään, miten lähetetään, miten lähetys puretaan ja miten purettu tieto käsitellään taas ymmärrettävään muotoon.

Terveydenhuollossa tärkeimpiä integraatiota vaativia järjestelmiä ovat potilastietojärjestelmät. Jokaisella sairaanhoitopiirillä on omat potilastietojärjestelmät, jotka muodostavat pohjan sote-ammattilaisten päivittäiselle työskentelylle. Potilastietojärjestelmiin tallennetaan tietoa potilaista sekä heille suoritetuista toimenpiteistä. (Rissanen & Lammintakanen, 2017) Esimerkkejä Suomessa käytössä olevista eri valmistajien potilastietojärjestelmistä ovat DomaCare (Invian Oy), Acute (Vitec Acute Oy), Lifecare (Tieto Finland Oy) ja Uranus (CGI Suomi Oy) (Valvira, 2022).

Jotta potilastietojärjestelmiin saadaan kaikki tarvittava potilaiden tieto, täytyy näihin järjestelmiin integroida kaikki potilasdataa tuottavat laitteet ja järjestelmät. Tällaisia järjestelmiä ja laitteita voivat olla esimerkiksi röntgenlaitteet, laboratorioiden omat testianalyysijärjestelmät, elintoimintoja valvovat laitteet tai muiden sairaanhoitopiirien ja yksityisten toimijoiden potilastietojärjestelmät. (Lehto & Neittaanmäki, 2017)

3 Arkkitehtuuriratkaisut

3.1 Viestinnän Protokollat ja ratkaisut

Integraatiot ja varsinkin järjestelmäintegraatiot tapahtuvat verkon yli välitettävien viestien ja pyyntöjen avulla (Lane, 2019). Integraatioiden ja verkkoviestinnän suunnittelun helpottamiseksi maailmalla on käytössä yleisiä standardisoituja protokollia verkkoviestintään ja tietojen esittämiseen. Nämä standardit mahdollistavat nopeamman järjestelmien ja viestinnän kehittämisen sekä helpomman yhteistyön usean eri toimijan välillä.

3.1.1 SQL-tiedonhallintakieli

SQL (Structured Query Language) on strukturoidun datan hakemiseen, luomiseen ja muokkaamiseen tarkoitettu ohjelmoinnissa hyödynnettävä kieli. SQL toimii relaatiotietokantojen, eli dataa taulukkomaisesti säilövien tietokantojen kanssa (Codd, 1990). Sen toiminta perustuu lausekkeisiin, joilla osoitettua tietokantaa käsketään suorittamaan tietty toiminto tietokannassa ja palauttamaan tarvittaessa tietoa tietokannasta halutun mukaisesti. (Melton & Simon, 1993)

SQL on standardoitu kieli (ISO/IEC JTC 1, 2016). Eri ohjelmistotoimittajat ovat kehittäneet joitain kielen toimintoja omiin suuntiinsa. Jokainen SQL-versio on kuitenkin säilyttänyt kielen perustoiminnot lähes yhdenmukaisina.

SQL ja sen variaatiot ovat usein olennainen osa Integraatioita, koska monet järjestelmä- ja palvelinratkaisutoimittajat sisällyttävät tuotteisiinsa mahdollisuuden käyttää SQL:n kaltaista rajapintakieltä kyselyiden tekemiseen (Bui, 2008). Kyselyiden lisäksi rajapintakielellä voidaan myös toteuttaa järjestelmän datan muokkaaminen rajapinnan kautta. Tämä mahdollistaa järjestelmään tallennettujen tietojen osittaisen hyödyntämisen ilman, että ulkoisen järjestelmän tarvitsisi olla suorassa yhteydessä kohdejärjestelmän koodiin.

3.1.2 API-ohjelmointirajapinta

API (Application Programming Interface) tarkoittaa ohjelmointirajapintaa, joka tarkoittaa ulkoisen toimijan ja järjestelmän välisen kommunikoinnin mahdollistavaa osaa. Se antaa ulkoiselle ohjelmalle tai käyttäjälle mahdollisuuden hyödyntää ohjelmiston toimintoja välittävän osan eli rajapinnan kautta (Jin, Sahni, & Shevat, 2018).

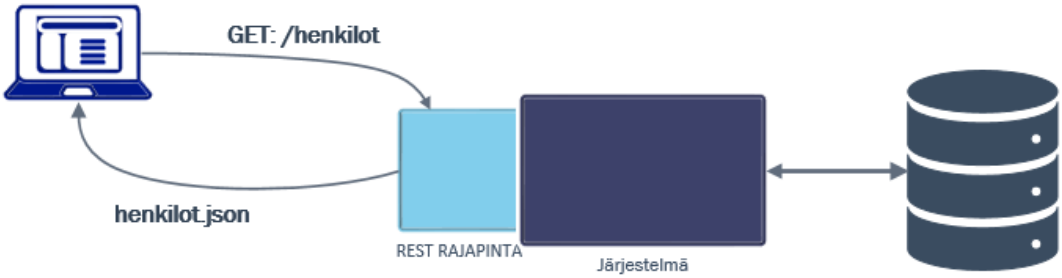
Käyttäjärajapintana voi olla esimerkiksi mobiilisovelluksen näytöllä näkyvät painikkeet. Käyttäjällä on todennäköisesti käsitys, mitä painikkeen painamisesta seuraa, vaikka hänellä ei ole mitään tietoa siitä, mitä järjestelmän sisällä tapahtuu. Näin toimii myös ohjelmistorajapinta. Ainoana erona on, että kaksi erillistä ohjelmistoa keskustelevat keskenään. Ohjelmistorajapinnan avulla voidaan tietoa vaihtaa ohjelmistojen välillä yhteisesti ymmärrettävällä tavalla.

Kun nykyään puhutaan API:sta, tarkoitetaan yleisesti juuri verkkorajapintaa eli HTTP-liikennettä hyödyntävää rajapintaa, joka tarjoaa verkkopyynnön esittävälle taholle vastauksena koneen sekä ihmisen ymmärtämää dataa JSON- tai XML-formaatissa (Lane, 2019)

3.1.3 REST-tyylikäsite

REST (Representational State Transfer) on arkkitehtuurityyli, joka on luotu verkkosovellusten suunnittelun ja toteutuksen ohjeistukseksi. Se ohjeistaa kehittämään hyvin skaalautuvia ja muun verkon kanssa kommunikoivia rajapintamenetelmiä. Etenkin verkkorajapintojen kehityksessä ”RESTful” -käsitteellä viitataan yleisesti hyväksyttyihin suunnitteluohjeistuksiin ja http-metodeihin, joiden avulla REST-rajapinnan, eli REST-API:n kanssa keskustellaan. (Fielding R. T., 2000)

Käytännössä REST-rajapinta tarkoittaa tiettyä verkkopalvelimelle määriteltyä osoitetta, joka ottaa vastaan neljää erilaista pyyntöä: GET, POST, PUT, DELETE. REST-osoitteeseen lähetetyn pyynnön sisältö tai avainsana ja pyynnön tyyli määrittävät minkä toiminnon REST-rajapinta (Kuva 2) pyytää palvelinta suorittamaan siihen tehtyjen sääntöjen mukaisesti. (Fieldning, 2014)



Kuva 2. Pelkistetty REST-rajapinnan prosessikuvaus. Rajapinta vastaanottaa pyynnöt ja välittää ne eteenpäin sisäiselle järjestelmälle.

3.1.4 JSON-tiedostomalli

JSON (JavaScript Object Notation) on tiedonsiirtoon tarkoitettu ohjelmointikielestä riippumaton standardisoitu strukturoidun datan esittämisen tiedostoformaatti (Ecma International, 2017). JSON on tekstitiedosto, jossa sen sisältämä tieto on jaoteltu attribuutti - arvo pareiksi. Nämä arvoparit kirjoitetaan yhtenäiseksi tekstijonoksi, jossa erottelu tapahtuu pilkuilla (Kuva 3). Arvo voi myös sisältää toisen JSON tekstin.

```
{
    "Henkilöt": {
        "Henkilö": [{
            "id":   123,
            "nimi": {
                "etunimi":  "Matti",
                "sukunimi": "Meikäläinen"
            },
            "sähköposti":   "matti.meikalainen@gmail.com",
            "maa":          "Suomi"
        },
        {
            "id":   123,
            "nimi": {
                "etunimi":  "Maija",
                "sukunimi": "Meikäläinen"
            },
            "sähköposti":   "maija.meikalainen@gmail.com",
            "maa":          "Ruotsi"
        }]
    }
}
```

Kuva 3. Esimerkki JSON-tiedoston rakenteesta. Henkilöt tietueen sisällä on kuvattuna kahden erillisen henkilön tiedot. Huomioitavaa on esimerkiksi ”nimi” -tietueen sisällä eriteltyinä olevat etunimi ja sukunimi.

3.1.5 XML-tiedostomalli

XML eli Extensible Markup Language on JSON:n tapaan strukturoidun tiedon esittämiseen ja järjestelmien välillä kommunikoimiseen tarkoitettu tiedostoformaatti. JSON-formaatista poiketen XML hyödyntää rakenteessaan tunnisteita (avaus tunniste, sulku tunniste), joiden väliin tunnisteen mukaisen attribuutin arvo eli data sijoitetaan (Kuva 4). Tunnisteiden toimintaperiaatteen ansiosta XML tukee tiedoston kommentointia ja datan tarkempaa jaottelua sekä kategorisointia tunnisteisiin lisättävien attribuuttien avulla. Tästä syystä XML-tiedostot ovat kooltaan hieman JSON-tiedostoja suurempia. Siksi JSON on yleensä parempi valinta yksinkertaisissa nopeissa ratkaisuissa, kun taas XML on parempi ratkaisu vaativampaan tiedonsiirtoon järjestelmien välillä. (Safris, 2021)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Henkilöt>
  <Henkilö>
    <id>123</id>
    <nimi>
      <etunimi>Matti</etunimi>
      <sukunimi>Meikäläinen</sukunimi>
    </nimi>
    <sähköposti>matti.meikalainen@gmail.com</sähköposti>
    <maa>Suomi</maa>
  </Henkilö>
  <Henkilö>
    <id>123</id>
    <nimi>
      <etunimi>Maija</etunimi>
      <sukunimi>Meikäläinen</sukunimi>
    </nimi>
    <sähköposti>maija.meikalainen@gmail.com</sähköposti>
    <maa>Ruotsi</maa>
  </Henkilö>
</Henkilöt>
```

Kuva 4. Esimerkki XML-tiedoston rakenteesta. Kuvattuna sama tieto kuin kuvassa 3.

3.2 Intersystems IRIS for Health

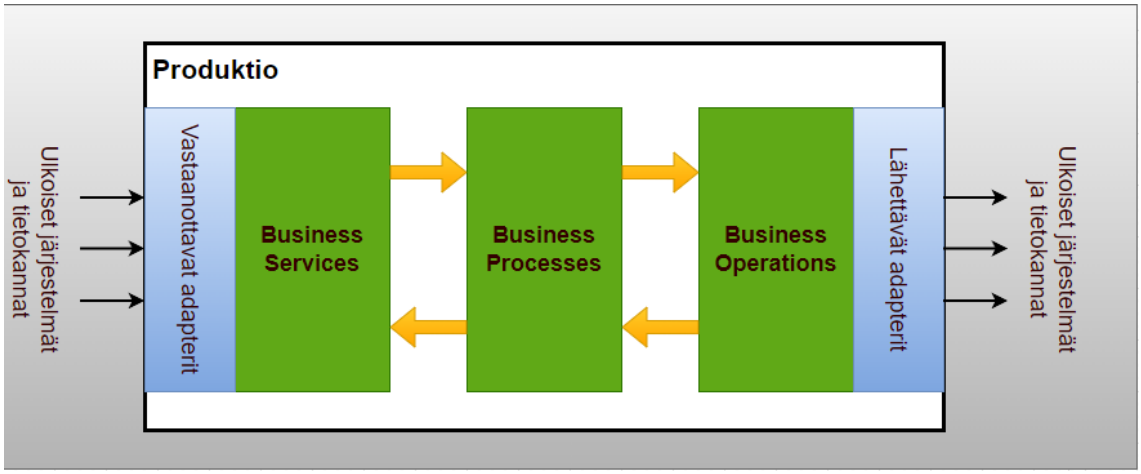
Intersystems (Intersystems, Cambridge, Massachusetts, Yhdysvallat) on tiedonhallintaratkaisuja toimittava globaali yritys, jonka erityisosaamisalueisiin kuuluvat integraatiot sekä terveystietojärjestelmät (Intersytems, 2022). Intersystems IRIS for Health on kokonaisvaltainen terveydenhuoltoon tarkoitettu alusta ja tietokantaratkaistu, jolla voi kehittää sekä ottaa käyttöön terveysdatan käsittelyyn soveltuvia applikaatioita. Toimintoihin kuuluvat datan muokkaamisen ja siirtämisen lisäksi laajat analytiikka- ja hallintaominaisuudet (Intersystems, 2021).

Järjestelmäintegraatioiden kannalta IRIS for Health -ratkaisun tärkein toiminnallisuus on siihen sisältyvä HealthShare-palvelun Health Connect -integraatiomoottori. Health Connect on integraatioiden toteuttamisen, suunnit-

telun ja valvonnan alusta, jossa yksinkertaisia integraatioita voidaan toteuttaa ja konfiguroida suoraan Intersystemsin tarjoamilla valmiilla ohjelmointiluokilla, jolloin datan siirtymistä ei tarvitse ohjelmoida erikseen. Terveysdatan siirtämiseksi Health Connect tukee FHIR-standardia HL7v2 ja HL7v3 muotoisen datan hallitsemiseksi. (Intersytems, 2021)

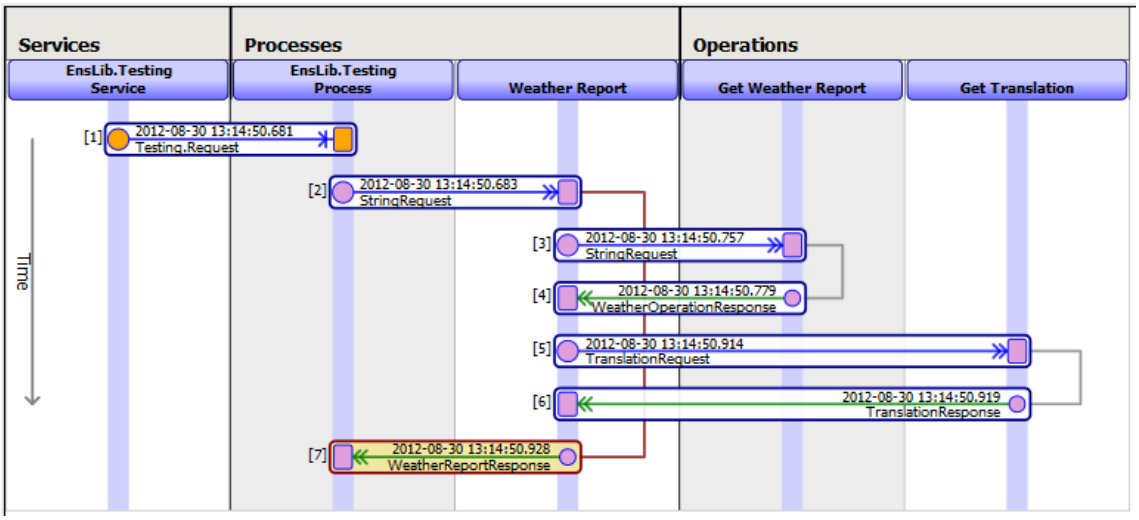
IRIS for Health -integraatioratkaisut perustuvat produktioiden, (engl. Interoperability Productions) luomiseen. Produktio on valmis graafisen käyttöliittymän omaava kehys integraatioiden toteuttamiseen ja se tarjoaa useita valmiita eri yhteysprotokollia ja metodeja tiedon siirtämiseen IRIS-palveluun ja sen ulkopuolelle. Produktio koostuu yhdestä tai useista kolmeen kategoriaan luokiteltavista Business Host -osista: Business Services, Business Processes ja Business Operations (Kuva 5). (Intersystems, 2021)

Kuvan 5 mukaisesti, Business Service ja Business Operation vastaavat yhteyksistä produktion ulkopuolelle, kun taas Business Process toimii nimensä mukaisesti prosessorina. Business Processin sisällä tapahtuu siis kaikki mahdollinen sanoman datan muokkaaminen ja tietojen hakeminen sekä tallentaminen, muiden osien toimiessa vain yhteysväylinä. Mikäli siirrettävä data tai sanoma ei vaadi ollenkaan muokkausta tai tallentamista, voidaan Business Process jättää produktiosta kokonaan pois.



Kuva 5. Pelkistetty Intersystems IRIS -produktio. Kuvattuna produktion sisällä tapahtuva datan siirtyminen ja Business Hostit. (mukailtu Intersystems, 2021)

Visual Trace (Kuva 6) tallentaa jokaisen viestin, jonka produktio vastaanottaa ja esittää saapuneen viestin vaiheittaisen prosessoinnin produktion sisällä (Intersystems, 2021). Kuvassa 6 Enslib.Testing Service vastaanottaa viestin ulkopuolisesta järjestelmästä, josta se siirtyy käsiteltäväksi säätietoja keräävään prosessiin. Sääprosessi kerää Business Operaation -moduulin avulla tarvittavat säätiedot ja käännökset ulkoisista järjestelmistä, jonka jälkeen se palauttaa koostetun tiedon.



Kuva 6. IRIS Visual Trace esimerkki. (Intersystems, 2021)

3.3 Caché ObjectScript -ohjelmointikieli

ObjectScript tai Caché ObjectScript on MUMPS-ohjelmointikielen mallia noudatteleva ohjelmointikieli, joka on tarkoitettu liiketoimintalogiikan, integraatioiden ja dataprosessoinnin ohjelmistojen nopeaan ja helppoon kehittämiseen Intersystems IRIS -data-alustoilla. (Intersystems, 2018)

MUMPS-kieli on tehokas datan prosessointikieli, joka toimii suoraan tietokannan sisällä (O'Kane, 2017). Tämä mahdollistaa tiedon nopean tallentamisen ja siirtämisen järjestelmän sisällä tietoa tarvitseville suorittaville ohjelmiston osille. (Trask, 1998)

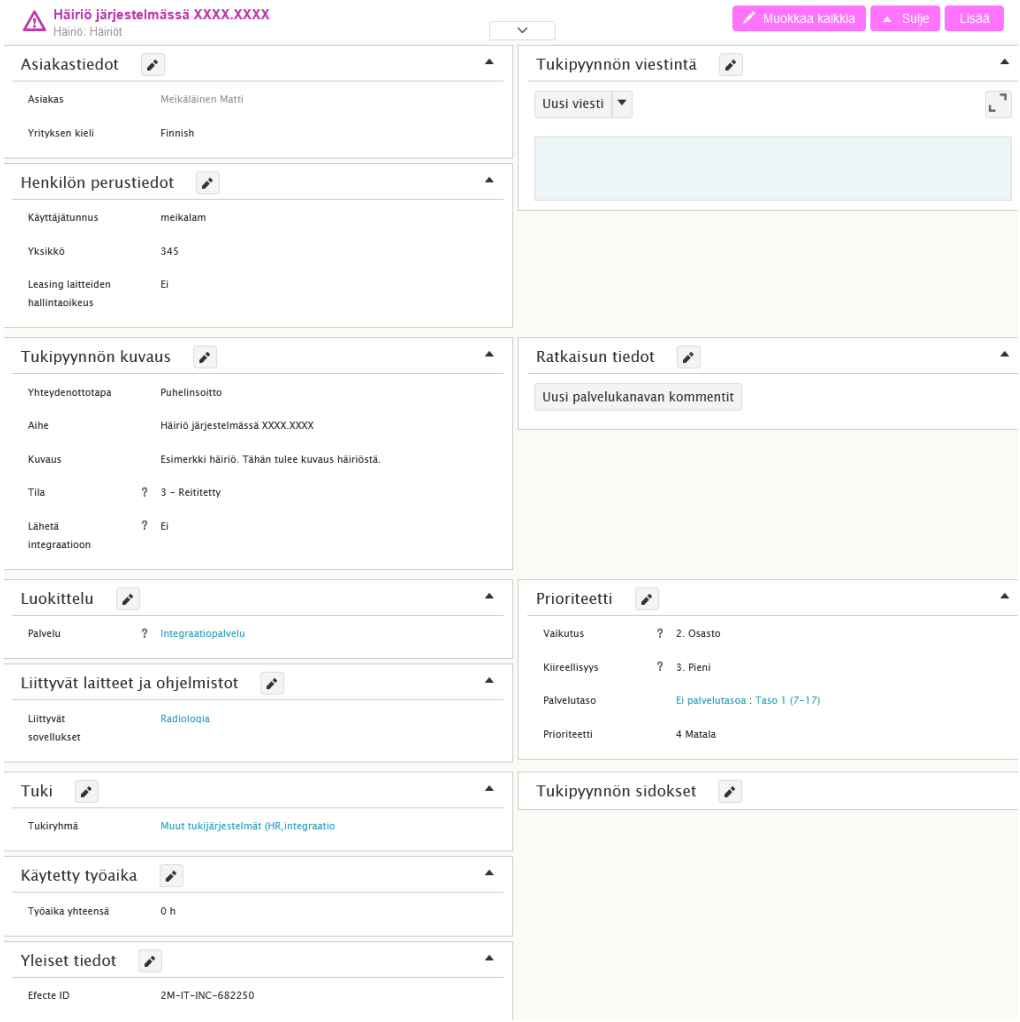
ObjectScriptin etuna on myös sen yhteensopivuus HealthConnect-hallintaportaalin näkymän kanssa. Tämä tarkoittaa sitä, että Healthconnect ymmärtää ObjectScriptissä tehtyjä muuttujien määrityksiä ja kommentteja implementoiden nämä koodissa määritellyt moduulit ja ominaisuudet suoraan hallintaportaalin graafiseen näkymään, mikäli näin halutaan (Intersystems, 2018). Tämän ansiosta Intersystemsin tarjoamaa integrointialustaa ja integraatioita pystyy lukemaan, muokkaamaan ja käyttämään helposti, niin ohjelmistokehittäjät, kuin ohjelmointikieltä osaamattomatkin järjestelmäasiantuntijat ja muut ammattilaiset.

3.4 Efecte ja IT-palvelunhallintajärjestelmät

IT-palvelunhallinnalla tarkoitetaan määritettyjä käytäntöjä ja prosesseja, joita käytetään yrityksen sisäisten prosessien, laitteiston, henkilöstön ja tehtävien kirjaamiseen, säilyttämiseen ja järjestelyyn (ManageEngine, 2020). IT-palvelunhallintajärjestelmä on palvelunhallinnan toteuttamista varten tuotettu järjestelmä, jonka tarkoituksena on esittää ja tuottaa palvelunhallinnan määritysten mukaista dataa helposti prosessoitavassa ja hallittavassa muodossa. IT-palvelunhallintajärjestelmässä käsitellään esimerkiksi muutoksenhallintaa, resurssienhallintaa, projektihallintaa, tietohallintoa ja häiriöhallintaa. (Bigelow, 2018)

Toimivat IT-palvelunhallintajärjestelmät on myös usein rakennettu noudattamaan ITIL-käytäntöjä, tai mukailemaan niitä (Bigelow, 2018). ITIL:llä (Information Technology Infrastructure Library) tarkoitetaan alun perin Yhdistyneen Kuningaskunnan tietokone ja telekommunikaatio viraston kehittämää palveluprosessirakennetta, jonka tarkoituksena on helpottaa, nopeuttaa ja standardisoida IT-alan palveluiden tuotannon elinkaarta. ITIL-rakenne käsittelee palveluiden tuottamisen strategiaa, suunnittelua, muuntautumista, operaatioita ja kehittämistä. Nykyään ITIL on globaalisti suurin ja käytetyin IT-palvelutuotannon malli. (Montgomery & Bigelow, 2020)

Efecte on suomalaisen samannimisen yhtiön Efecte Oy:n tuottama IT-palvelunhallintajärjestelmä. Efecten yleinen toimintaperiaate, monien muiden palvelunhallintajärjestelmäntarjoajien tavoin, perustuu tietokortteihin (engl. Data Card). Yksi tietokortti edustaa yhtä muutosta, resurssia, häiriötä tai muuta tallennettavaa informaatiota. Esimerkiksi häiriön tietokortti eli tiketti (Kuva 7) edustaa tietoa yhdestä integraatiojärjestelmässä havaitusta häiriöstä. Tikettiä muokataan tarpeen mukaan ITIL:n mukaisen häiriökäsittelyn aikana, ja se voidaan siirtää eteenpäin häiriöstä vastaavalle vastuuryhmälle, eli ongelman selvityksestä vastaaville työntekijöille järjestelmän sisällä (Palilingan & Batmetan, 2018). Tiketti toimii siis häiriön yleisenä dokumentointialustana aina ratkaisuun ja tiketin sulkemiseen, tai arkistointiin saakka.



Kuva 7. Efecten häiriötietokortti. Kuvassa esitettynä yleisiä tietokortissa esiintyviä tietuekenttiä ja mahdollisia näissä kentissä olevia arvoja.

Efectessä on jokaiselle erilaista tietoa edustavalle tietokortille oma mallinsa (engl. template). Siten jokainen samanlaista resurssia edustava tietokortti on rakenteeltaan yhtenäinen. Tekstimuodossa olevan tiedon lisäksi voidaan tietuekenttiin määrittää arvoksi toisia tietokortteja. Linkitykset toisiin tietokortteihin näkyvät kuvassa 7 sinisinä teksteinä. Tämän avulla tietokortteja tulkitsevan henkilön tai järjestelmän on helppo ymmärtää tietokorttia vastaavan resurssin relaatiot muihin yrityksen resursseihin verrattuna. (Efecte, 2020)

Efecteä käytetään yleisesti selaimen käyttöportaalin kautta, josta tietokortteja voidaan hakea, luoda ja muokata. Käyttöportaali mahdollistaa myös helpon datan visualisoinnin sekä raporttien tekemisen käyttäjän tarpeiden mukaisesti. Jokaiselle käyttäjälle voidaan määritellä oikeudet, jolloin he pääsevät käsiksi vain työnsä kannalta vaadittaviin tietokortteihin. (Efecte, 2020)

Efecten pilvipohjaiseen ratkaisuun kuuluu myös Efecte Web API -ratkaisu. Efecte Web API mahdollistaa Efecten tietokorttien muokkauksen ja luomisen sekä tiedonhakemisen HTTP-rajapinnan kautta. Tietojen luominen ja muokkaaminen tapahtuu lähettämällä Efecteen XML-muotoisia tietoja. Tietojen hakeminen onnistuu käyttämällä SQL:n kaltaista Efecten omaa EQL-kieltä hakulausekkeiden suorittamiseen. (Efecte, 2020)

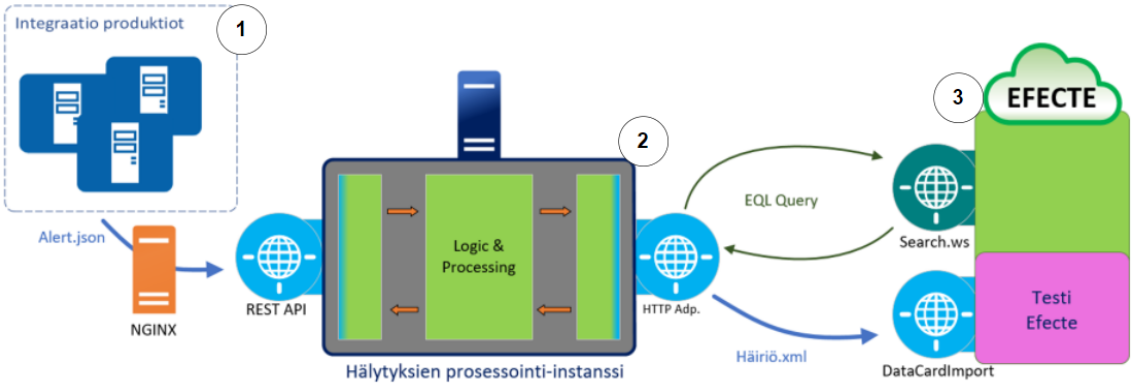
4 Toteutettava ratkaisu

Opinnäytetyössä toteutettiin järjestelmäintegraatioiden tuottamien hälytyksien luonnin ja luokittelun automatisointi toimeksiantajan IT-palvelunhallintajärjestelmään. Toteutusta voidaan myös ajatella yksisuuntaisena integraationa hälyttävän integraatioliittymän (lähdejärjestelmä) ja palvelunhallintajärjestelmän välillä (kohdejärjestelmä). Tämän ratkaisun toteuttamisessa hyödynnettiin toimeksiantajan jo olemassa olevia ratkaisumalleja ja työkaluja, kuten Intersystems IRIS for Health -integraatioalustaa ja Efecte-IT-palvelunhallintajärjestelmää. Näiden avulla myös ratkaisun jatkokehittäminen ja käyttäminen helpottuivat.

Ratkaisu kehitettiin projektiluontoisena toteutuksena, jossa suunnittelu ja katselmointi tapahtuivat kaksi kertaa viikossa järjestettävissä kokouksissa. Ohjelmoitavien moduulien toteuttaminen, tiedonvälityksen toteutus ja prosessilogiikan suunnittelun päävastuu jäi opiskelijalle, kun taas työn etenemisen ohjaaminen sekä vaadittavien verkkoliikenteen ohjauksien ja palvelimille tehtävien muutoksien päävastuu oli projektiin osallistuneilla ratkaisuarkkitehdeillä.

4.1 Toteutus

Toteutuksen toimintamalli on esitetty kuvassa 8. Jokainen yksittäinen integraatiohälytys käynnistää oman hälytysprosessinsa hälyttävän järjestelmän sisällä. Generoitu hälytys siirtyy lähettävälle adapterille, joka välittää sen erillisellä palvelimella sijaitsevalle käsittely- ja luokitteluprosessille. Tämä käsittelyprosessi siirtää valmiin hälytyksen luontisanoman Efecteen, jossa siitä generoituu sanoman mukainen häiriötiketti. Kaikki käsittelyprosessin aikainen hälytyssanoman käsittely, lähetys ja vastaanotto toteutettiin spesifeillä projektia varten ohjelmoiduilla ObjectScript -luokilla.



Kuva 8. Toteutuksen prosessikuvaus.

Häiriön luomisen prosessi käynnistyy kuvan 8 kohdassa 1, kun jokin Intersystem IRIS -järjestelmäintegraation osa hälyttää virheestä. Tästä hälytyksestä muodostuu automaattisesti Objectscript Ens.AlertRequest -objekti, joka otetaan käsittelyyn. Objektista kerätään hälytystekstin lisäksi joitakin palvelimen ja produktion tietoja. Tämän jälkeen ne kootaan JSON-tiedostoksi helpon ja nopean tiedonsiirron mahdollistamiseksi, ja lähetetään eteenpäin HTTP-viestinä hälytyksen käsittelyn keskitetylle prosessointi-instanssille (Kuva 8, kohta 2).

Hälytyksiä varten on prosessointi-instanssille rakennettu REST-API, joka vastaanottaa ja välittää viestit hälytykset prosessoivalle produktiolle. Kun hälytyksen prosessointi ja luokittelu ovat valmiita, muodostetaan käsittelyn aikana tuotettujen tietojen avulla Efecten XML-vaatimusten mukainen XML-tiedosto (Kuva 9). Tämä tiedosto välitetään uutena HTTP-viestinä Efecten Web API:n DataCardImport palvelulle, joka luo viestin sisällön perusteella kuvan 7 kaltaisen häiriötiketin Efecteen (kuva 8, kohta 3).

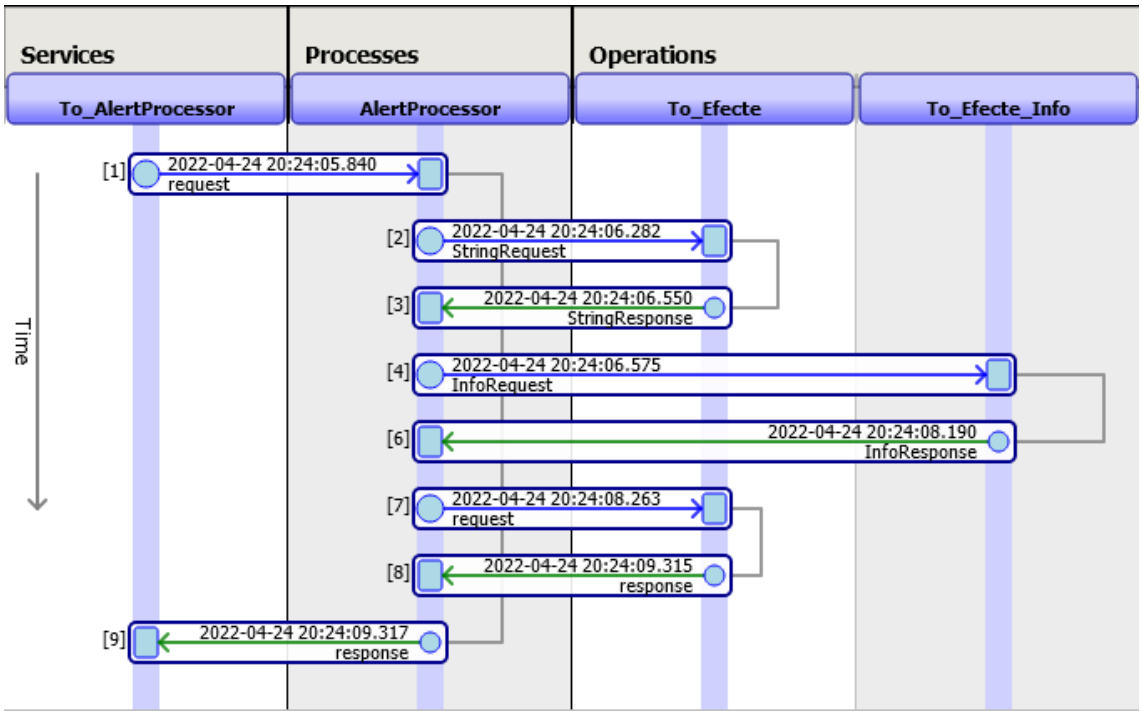
```
<?xml version="1.0" encoding="UTF-8" ?>
<entityset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="Test_Import.xsd">
    <entity name="Testi integraatio_hälytys 2">
        <template code="incident"/>
        <attribute code="subject">
            <value>Testi integraatio_hälytys 2</value>
        </attribute>
        <attribute code="description">
            <value>Tähän kuvaus tapahtuneesta häiriöstä, eli järjestelmän tuottama hälytysteksti.
                Lisäksi tietoja hälytyksen aiheuttaneesta konfiguraatiosta sekä palvelimesta.</value>
        </attribute>
        <attribute code="inc_category">
            <value>palvelu johon kyseinen hälytyksen aiheuttava integraatio kuuluu</value>
        </attribute>
        <attribute code="liityvat_sovellukset">
            <value>Minkä sovelluksen/järjestelmän integraatio</value>
        </attribute>
        <attribute code="impact">
            <value>Esim: 2. Osasto</value>
        </attribute>
        <attribute code="urgency">
            <value>Esim: 3. Pieni</value>
        </attribute>
        <attribute code="support_group">
            <value>Mille efecten sisäiselle tukiryhmälle tiketti osoitetaan</value>
        </attribute>
        <attribute code="perustelu_pp">
            <value>Vaadittava lokitieto, Esim: Luodaan testihäiriö integraatiohälytyksiä varten</value>
        </attribute>
    </entity>
</entityset>
```

Kuva 9. Esimerkki Efecteen lähetettävästä XML-tiedostosta. Täydennetty kuvaavilla esimerkkiarvoilla.

Kuvan 8 prosessikaaviossa on esitetty myös Efecte Web API:n Search.ws -palvelulle menevät EQL-kyselyt. Nämä kyselyt ovat osa hälytysprosessoinnin rutiineja. IT-palvelunhallintajärjestelmänä Efecte pitää sisällään tietoa esimerkiksi kaikista organisaation toteuttamista integraatioliittymistä, niiden palvelimista ja vastuualueista sekä näiden relaatioista. Hakemalla ja hyödyntämällä näitä tietoja hälytyksen prosessoinnin aikana voidaan hälytyksestä muodostuva häiriö luokitella tarkasti ja ohjata suoraan oikealle vastuuryhmälle. EQL-kyselyiden hyödyntäminen myös helpottaa toteutuksen skaalautuvuutta sekä parantaa sen jatkuvuutta, koska prosessoinnin toimivuus ei ole täysin riippuvainen yksittäisistä koodin sisälle määritetyistä käsittelysäännöistä.

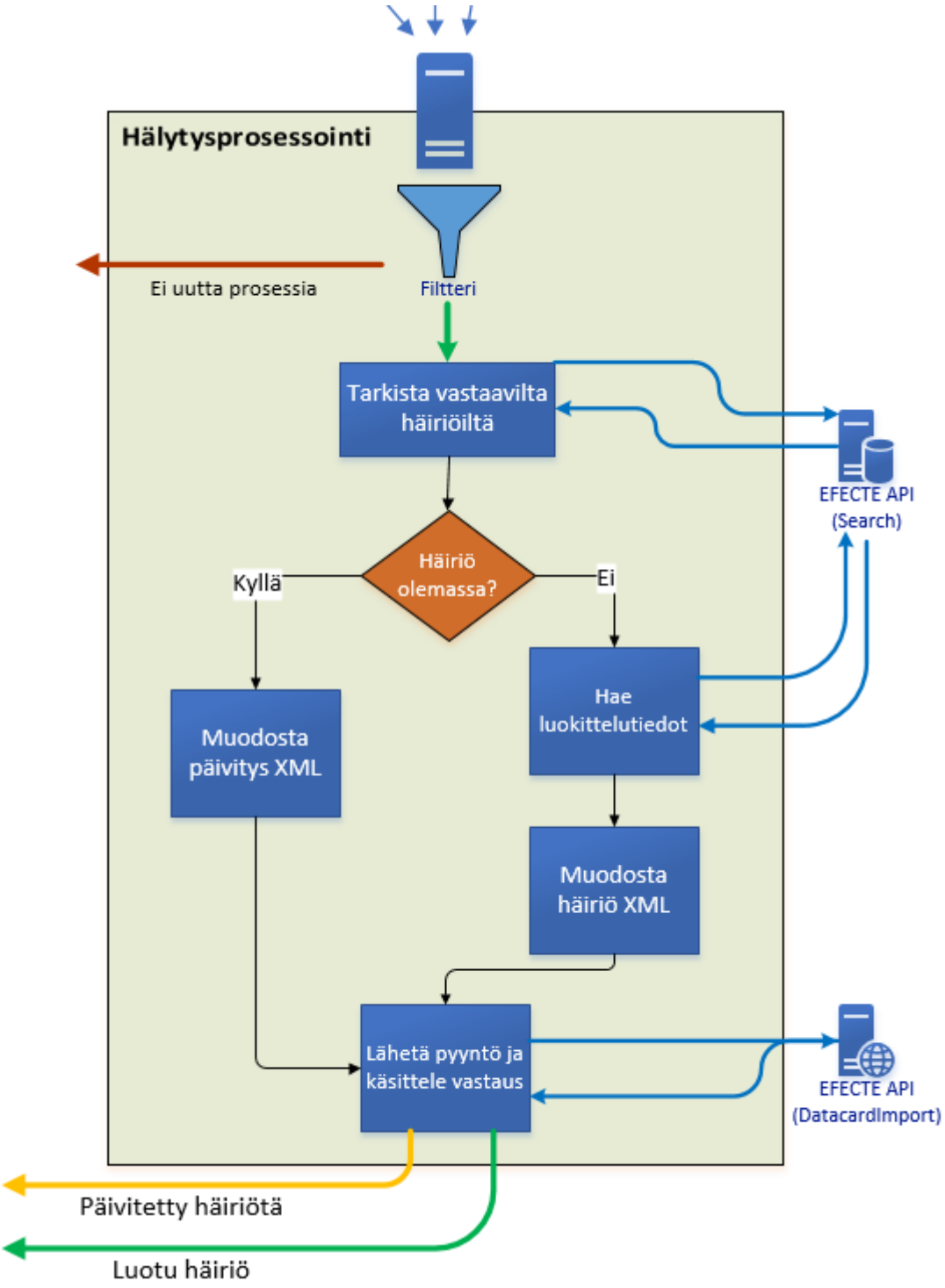
Objectscriptillä koodatuista luokista ja metodeista koostuva hälytyksien prosessointi (Kuva 8, kohta 2) vastaa esimerkiksi turhien tai toistuvien hälytyksien pois

suodattamisesta, hälytyksen aiheuttavan järjestelmän tarkemmasta tunnistamisesta, hälytyksen luokittelusta sekä häiriön ohjaamisesta suoraan oikealle Efecten tukiryhmälle hälytyksen sisällön perusteella. Kuvassa 10 on esitetty hälytyksien prosessoinnista muodostuva HealthConnectin Visual Trace -kaavio tilanteesta, jossa prosessointi tunnistaa uuden häiriön luonnin tarpeelliseksi, hakee tarvittavat luokittelutiedot Efectestä ja lähettää muodostetun XML-muotoisen sanoman Efecten DataCardImport-palvelulle.



Kuva 10. Kuvankaappaus IRIS Visual Trace -hälytyksenkäsittelyprosessista, jossa Efecteen lähetetään uusi häiriötiketin luomispyyntö.

Kuvassa 11 on esitetty prosessoinnin merkittävimmät käsittelyvaiheet sekä osoitettu miten eri hälytyksien kanssa toimitaan niiden sisällöstä riippuen. Mikäli samasta lähteestä tulee useampi samanlainen hälytys määritellyn aikaikkunan sisällä, suodatetaan nämä hälytykset pois prosessoinnista suodattamisen kohdalla. Tämän jälkeen prosessi vielä tarkistaa, onko hälytyksen tietoja vastaavalla tiedoilla jo olemassa olevaa aktiivista häiriötä. Mikäli tällainen häiriö löytyy Efectestä, suoritetaan vain häiriötiketin päivitys, eikä luoda täysin uutta häiriötikettiä.



Kuva 11. Prosessointi-instanssin logiikan kuvaus.

Kaikki lopullisessa toteutuksessa tapahtuva järjestelmien välinen viestintä toteutettiin HTTP-protokollan jatketulla HTTPS-viestintäprotokollalla. Tämän avulla voitiin varmistaa tiedon turvallinen siirtyminen internetissä ja intraneteissä. HTTPS-viestinnän toteuttamista varten välityspalvelimelle, keskitetylle prosessointi-instanssille ja kaikille integraatioproduktioille asetettiin käyttöön TLSv1.2-konfiguraatiot.

4.2 Testaus

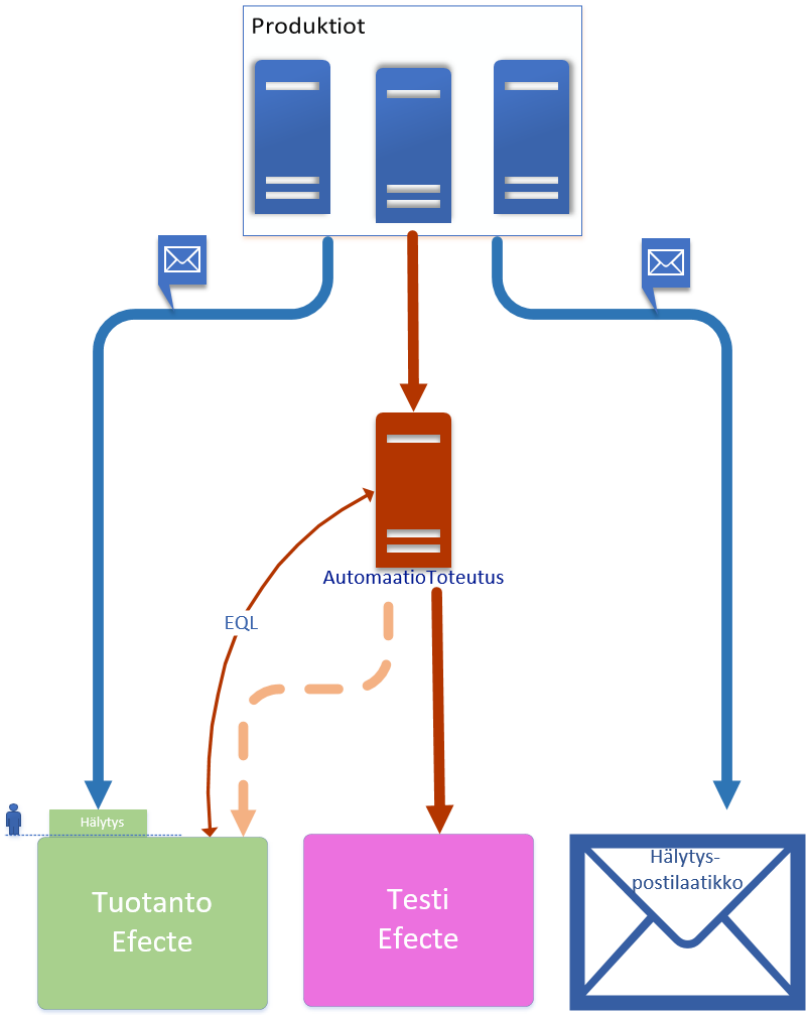
Sovellus- ja järjestelmäkehityksen tärkeimpiä ja aikaa vievimpiä vaiheita on testaus. Testauksen tärkeys korostuu etenkin järjestelmissä, joiden luotettavuus on oltava todella korkealla tasolla. Testauksen tarkoituksena on varmistaa sovelluksen suunnitellun, suunnitelman mukainen toimivuus ja vähentää jatkokehityksessä ilmeneviä ongelmia. Nyrkkisääntönä voidaankin pitää, että testaus vie sovelluskehityksen ajasta noin 50 prosenttia. (Maneela & Gaurav, 2012)

Testauksen tärkeys korostui myös opinnäytetyön toteutuksessa. Testausta suoritettiin vaiheittain järjestelmäkehityksen aikana, ja se keskittyi verkkoyhteyksien, datankäsittelyn ja prosessoinnin logiikan testauksiin. Vaiheittaisen testauksen avulla pystyttiin varmistumaan kehitettyjen toiminnallisuuksien oikeanlaisesta toiminnasta, ennen kuin kehityksessä siirryttiin seuraavaan vaiheeseen ja testatun toteutuksen päälle ruvettiin lisäämään uusia toiminnallisuuksia.

Kehityksen aikana tehty testaus aloitettiin varmistamalla viestien kulkeminenjärjestelmien välillä JSON- ja XML-formaateissa (Kuva 8). Kun järjestelmien välisestä kommunikoinnista saatiin varmuus, siirryttiin kehittämään ja testaamaan keskitetyn instanssin logiikkaa ja prosessointia. Tämän jälkeen toimivaksi todettuun kokonaisuuteen voitiin lisätä EQL-kyselyt Efecteen, ja testata niiden toimivuutta kehitettyä hälytysintegraatiota vastaan.

Koko kehitys- ja testausprosessi toteutettiin omassa testiympäristössään ja lisättiin vaiheittain tuotantoympäristöön osaksi tuotannon varsinaisia prosesseja.

Toimeksiantajalla on olemassa omasta Efectestään ja joistakin Integraatiotuotannoistaan testiversiot, jotka ovat rinnastettavissa tuotannon vastaaviin ratkaisuihin. Hyödyntämällä näitä testiympäristöjä, testaus voitiin suorittaa turvallisesti vaarantamatta nykyistä tuotantoa. Samalla voitiin kuitenkin luoda täysin tuotannon kaltaista testidataa toiminnan varmistamista varten. Tämän testitoteutuksen avulla toteutettava ratkaisu voitiin lopulta lisätä myös osaksi tuotannon prosesseja, kuitenkin siten, että nykyinen käytössä oleva sähköposti-ilmoituksilla toimiva hälytysprosessi ei häiriintynyt (Kuva 12). Tämä mahdollisti myös uuden hälytyskäsittelyprosessin tuottaman datan suoran vertailun nykyiseen.



Kuva 12. Hälytysprosessit rinnakkain. Sinisillä nuolilla esitetty nykyinen tuotannossa oleva ratkaisu. Punaisella esitettynä rinnalla toimiva testitoteutus.

5 Pohdinta

5.1 Toteutuksen arviointi

Koska projektin toteutus käsittää useita eri lähde- sekä kohdejärjestelmiä, ja koska sen toimintasäde kattaa useita organisaation sisäisiä työryhmiä, kului kehityksen aikaresursseista paljon eri osa-alueiden osaajien konsultointiin. Tämä aiheutti ennalta-arvaamattomia aikakustannuksia. Alkuperäiseen suunnitelmaan verrattuna toteutetusta työstä jätettiin pois käsittelysääntöjen hallitsemista ja muokkaamista varten suunnitellun käyttöliittymän kehittäminen. Kehitykseen varatun ajan puitteissa ei joidenkin suunniteltujen tekstinkäsittelysääntöjen implementointi ollut mahdollista, joten ne siirrettiin jatkokehitykseen. REST-rajapintaa hyödyntämällä tarjottiin mahdollisuus helpolle jatkokehittämiselle, koska rajapintaan voi helposti lisätä uusia toiminnallisuuksia vaikuttamatta tämänhetkisen toteutuksen moduulien toimintaan.

Toteutuksessa käytettyjen ohjelma- ja protokollaratkaisujen takana oli hyvin vahvasti suurempien järjestelmien vaatimukset sekä näiden järjestelmien tarjoamien tiedonsiirtotyökalujen toimintatehokkuudet. Tiedonsiirtoratkaisuihin valittiin XML- ja JSON-merkintäkielet niiden yleisyyden johdosta. Lisäksi Intersystemsin integraatioratkaisut tukevat molempien tiedostomuotojen käsittelyä. Tiedonsiirto olisi voitu toteuttaa myös kokonaan XML-muotoisena kahden eri protokollan sijasta. Kuitenkin XML-formaatin raskaammasta muodosta johtuen optimaalisen prosessointitehon säilyttämiseksi sekä ohjelmoinnin selkeyttämiseksi hyödynnettiin JSON -formaattia, kun se oli mahdollista.

Tehokas ja mahdollinen toinen ratkaisuvaihtoehto häiriöiden luokittelulle olisi hälytysprosessointimallin korvaaminen koneoppimismallilla. Tämä kehitysmahdollisuus nousi esille jo opinnäytetyöprojektin aikana, ja sen mahdollisesta kehittämisestä ja resurssitarpeista keskusteltiin projektipalaverien aikana. Koneoppimisratkaisun tarkoituksena on kouluttaa Efectestä saatavien ratkaistujen häiriöiden tietojen avulla ennustava malli, jolle syötetään hälytyksestä syntyneitä tietoja. Näiden tietojen avulla malli palauttaa sen arvioimat oikeat IT-

palvelunhallintajärjestelmän vaatimat tiketin luokittelut. (Kishore, 2019) Tämän ratkaisun avulla voidaan luopua yksittäisten käsittelysääntöjen luomisesta ja hallitsemisesta lähes täysin. Mikäli mallin koulutusta jatketaan sen tuotantoon siirtämisen jälkeenkin, myöskään uusien integraatioiden implementointi osaksi hälytysjärjestelmää ei vaadi jatkuvaa hälytysprosessoinnin muokkaamista.

5.2 Jatkokehitysmahdollisuudet

Toteutettu ratkaisu tarjoaa paljon jatkokehitysmahdollisuuksia, ja kehitysmahdollisuuksia nousikin esiin keskusteluissa jo nykyisen ratkaisun toteutusprojektin aikana. Aikarajoitteista johtuen, näitä ominaisuuksia tai ratkaisuja ei kuitenkaan ehditty lisäämään nykyiseen toteutukseen, mutta niiden lisäämisen mahdollisuus otettiin huomioon kehityksessä. Jatkokehitysprojekteja voisivat olla esimerkiksi käsittelysääntökäyttöliittymän kehitys, luokittelun toteuttaminen tekoälymallilla ja hälytysstatistiikkakäyttöliittymän kehittäminen.

5.2.1 Käsittelysääntöjen hallintaliittymä

Opinnäytetyössätoteutettu ratkaisu sisältää EQL-hakujen lisäksi myös hälytyksen käsittelysääntöjä, jotka perustuvat suurimmaksi osaksi hälyttävän palvelimen, hälyttävän produktion osan ja hälytystekstin tunnistamiseen. Toimeksiantajan tuottamien integraatioiden määrästä johtuen on kehitysvaiheessa ollut mahdotonta huomioida kaikkia mahdollisia erikoistapauksia, joihin yleiset käsittelysäännöt eivät välttämättä päde. Tästä syystä, ja koska uusia integraatioita kehitetään jatkuvasti, olisi perusteltua kehittää hälytysautomaatioratkaisulle jonkinlainen käyttöportaali, joka esittäisi olemassa olevat käsittelysäännöt helposti luettavassa muodossa, ja tarjoaisi mahdollisuuden lisätä uusia sääntöjä ilman tarvetta muokata hälytyskäsittelyn lähdekoodia. Jo toteutettua REST-rajapintaa voitaisiin hyödyntää tässä jatkokehityksessä lisäämällä sille tarvittavia sääntö-

jenmuokkausoperaatioita, joita voisi käyttää tarvetta varten kehitettävän verkkoapplikaation kautta.

5.2.2 Hälytysstatistiikan käyttöliittymä

Hälytyksien seurannan kannalta käytännöllinen toiminnallisuus olisi hälytyksien statistiikan seurannan mahdollistava graafinen ratkaisu, jota käytettäisiin tarkoitusta varten kehitetystä verkkoaplikaatiosta. Vaikka Efecten toimintoihin kuuluu hyvät statistiikka- ja seurantamahdollisuudet, pystyy se seuraamaan vain sille asti hälytysinstanssilta saapuneita häiriöpyyntöjä. Koska hälytyksen prosessoinnissa suodatetaan pois turhia hälytyksiä sekä suoritetaan olemassa olevien häiriöiden muokkaamista, olisi paremman seurannan kannalta tehokkaampaa luoda sisäinen tietokantaratkaisu. Tämä tietokanta tallentaisi jokaisen suoritetun hälytysprosessin sen lopullisesta statuksesta huolimatta. Näin hälytysprosessoinnin tehokkuuden ja integraatioiden virheherkkyyden seuranta tehokkaampaa.

5.2.3 Tiketin luokittelu koneoppimisella

Myös huomioitava kehitysmahdollisuus olisi hälytysprosessointimallin uudelleensuunnittelu ja osittainen korvaaminen edellä mainitulla koneoppimismallilla. Tämä malli voisi hyödyntää nykyisen toteutuksen tiedonsiirtokanavia ja arkkitehtuuria, mutta muutoksessa EQL-kyselyt korvattaisiin ennustavalla tekoälymallilla.

6 Yhteenveto

Opinnäytetyön tavoitteena oli kehittää nykyisen toimintamallin korvaava integraatiohälytyksien käsittelyjärjestelmä, jonka avulla voidaan vähentää henkilöstöresurssien määrän tarvetta häiriötilanteiden käsittelyn ja seuraamisen osalta. Tämä tavoite pyrittiin toteuttamaan kehittämällä automatisoitu filtteröintiä, käsittelysääntöjä ja palvelunhallintajärjestelmän tietojen haravointia toteuttava järjestelmä. Suunnitellut perustoiminnallisuudet saatiin toteutettua ja testattua suunnitelman mukaisesti, mutta ratkaisun tuotantoon viemiseksi ja aiemman ratkaisun korvaamiseksi vaatii toteutus edelleen toiminnallisuuksien jatkokehitystä sekä keskustelua organisaation yhteisistä prosesseista.

Lähteet

Altexsoft. (2021, 3 10). *System Integration: Types Approaches, and Implementation Steps*. Retrieved 4 5, 2022, from Altexsoft: https://www.altexsoft.com/blog/system-integration/

Bigelow, S. J. (2018). ITSM (IT Service Management). TechTarget.

Bui, H.-L. (2008). *Survey and Comparison of Event Query.* München: Ludwig-Maximilians Universität.

Cloudflare. (2022, 04). *What Is HTTPS?* Retrieved from Cloudflare: https://www.cloudflare.com/learning/ssl/what-is-https/

Ecma International. (2017). *The JSON Data Interchange Syntax.* Retrieved from https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf

Efecte. (2020). Efecte Platform Description 2020.4 .

Efecte. (2020). Efecte Web API documentation.

Fielding, R. T. (2000). Representational State Transfer (REST). In R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures.*

Fielding, R., Irvine, U., Gettys, J., Mogul, J., Compaq, Frystyk, H., . . . Berners-Lee, T. (1999). *Hypertext Transfer Protocol -- HTTP/1.1.* The Internet Society.

Fieldning, R. T. (2014). Section 4: Request Methods. In R. T. Fieldning, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* Internet Engineering Task Force (IETF) .

Haglund, J. (2018, 1 9). *Järjestelmäintegraatio, mitä se on selkokielellä?* (Alfame) Retrieved 4 5, 2022, from Alfame:

https://www.alfame.com/blog/jarjestelmaintegraatio-mita-se-on-selkokielella

HL7 International. (2019, 09). *About HL7.* Retrieved from HL7 International: https://www.hl7.org/about/index.cfm?ref=nav

IBM Cloud Education. (2019, 05). *IT Infrastructure Library (ITIL)*. Retrieved from IBM Cloud Learn Hub: https://www.ibm.com/cloud/learn/it-infrastructure-library

Intersystems. (2018). *Using Caché ObjectScript.* Retrieved from https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GCOS

Intersystems. (2021). *Intersystems IRIS for Health 2021.2.* Retrieved from https://docs.intersystems.com/irisforhealthlatest/csp/docbook/DocBook.UI.Page.cls

Intersystems. (2021). *Introduction to Interoperability Productions.* Retrieved from Intersystems Documentation: https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=EGIN_intro

Intersytems. (2021). *HealthShare Health Connect 2021.2.* Retrieved from https://docs.intersystems.com/healthconnectlatest/csp/docbook/DocBook.UI.Page.cls

ISO/IEC JTC 1. (2016). Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).

Jin, B., Sahni, S., & Shevat, A. (2018). *Designing Web APIs.* O'Reilly Media.

Kishore, P. (2019). *IT Support Ticket Classification and deployment using Machine Learning and AWS Lambda.* Retrieved from https://towardsdatascience.com/it-support-ticket-classification-and-deployment-using-machine-learning-and-aws-lambda-8ef8b82643b6

Lane, K. (2019, 10 10). *Intro to APIs: History of APIs*. Retrieved from blog.Postman: https://blog.postman.com/intro-to-apis-history-of-apis/

Lehto, M., & Neittaanmäki, P. (2017). Suomen terveysdataympäristö. *Informaation tiedekunnan julkaisuja*(35/2017).

ManageEngine. (2020). *What is ITSM?* Retrieved from ManageEngnine ServiceDesk Plus: https://www.manageengine.com/products/service-desk/itsm/what-is-itsm.html

Maneela, T., & Gaurav, D. (2012). A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models. *2*(3).

Melton, J., & Simon, A. R. (1993). *Understanding the New SQL: A Complete Guide.* San Francisco: Morgan Kaufmann Publisher, Inc.

Montgomery, J., & Bigelow, S. J. (2020). *ITIL (Information Technology Infrastructure Library)*. (TechTarget) Retrieved from Techtarget: https://www.techtarget.com/searchdatacenter/definition/ITIL

O'Kane, K. C. (2017). *A Quick Introduction to the Mumps Programming Language.* Retrieved from http://www.cs.uni.edu/~okane/source/MUMPS-MDH/MumpsTutorial.pdf

Palilingan, V., & Batmetan, J. (2018). Incident Management in Academic Information System using ITIL Framework. *IOP Conference Series: Materials Science and Engineering*(306).

Rissanen, S., & Lammintakanen, J. (2017). *Sosiaali- Ja Terveysjohtaminen.* Helsinki: Sanoma Pro Oy.

Safris, S. (2021). A Deep Look at JSON vs. XML, Part 1: The History of Each. Retrieved from Toptal: https://www.toptal.com/web/json-vs-xml-part-1#:~:text=JSON%20is%20a%20data%20interchange,of%20any%20XML%20sub%2Dlanguage.

SSL-tukitiimi. (2021, 09). *Mikä on SSL?* Retrieved from SSL.com: https://www.ssl.com/fi/FAQ/faq-mik%C3%A4-on-SSL/

Terveysministeriö, S. j. (2022). *Sairaanhoitopiirit ja erityisvastuualueet*. (Sosiaali- ja Terveysministeriö) Retrieved from https://stm.fi/sairaanhoitopiirit-erityisvastuualueet

The Office of the National Coordinator for Health . (2019, 08). *What Is FHIR?* Retrieved from healthit.gov: https://www.healthit.gov/sites/default/files/2019-08/ONCFHIRFSWhatIsFHIR.pdf

Trask, G. S. (1998). *M Technology and MUMPS Language FAQ*. Retrieved from http://71.174.62.16/MDC/faq.htm

Umapathy, K., Purao, S., & Barton, R. R. (2008). Designing enterprise integration solutions: Effectively. *European Journal of Information Systems*(17. 518-527. 10.1057).

Valtioneuvosto. (2021). *Hyvinvointialueiden perustaminen*. (Valtioneuvosto) Retrieved from https://soteuudistus.fi/hyvinvointialueiden-perustaminen

Valtioneuvosto. (2021). *Mikä on hyvinvointialue?* (Valtioneuvosto) Retrieved from https://soteuudistus.fi/mika-on-hyvinvointialue

Valvira. (2022). *Sosiaali- ja terveydenhuollon tietojärjestelmät.* Retrieved from https://www.valvira.fi/terveydenhuolto/sosiaali-ja-terveydenhuollon-tietojarjestelmat