SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

KANTA-ADAPTERIN PARA-METRISOINTITYÖKALU

TEKIJÄ/T: Laura Kemppainen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Laura Kemppainen	
Työn nimi Kanta-adapterin parametrisointityökalu	
Päiväys 15.12.2019	Sivumäärä/Liitteet 45
Ohjaaja(t) Lehtori, Jukka Kinnunen, Lehtori, Sami Lahti	
Toimeksiantaja/Yhteistyökumppani(t) Mediconsult Oy	

Tiivistelmä

Kanta-yhteys on olennainen osa nykyisiä potilastietojärjestelmiä, koska yhä useammat asiakirjat ja reseptit pyritään tallentamaan Potilastiedon arkistoon ja Reseptipalveluun. Omakanta-palvelun kautta potilas voi itse tarkastella omia terveystietojaan. Lisäksi tiedot ovat haettavissa palvelusta jokaiseen potilastietojärjestelmään, joihin on määritetty Kanta-yhteys. Kanta-adapterista johtuvat ongelmat ilmenevät useimmiten arkistointiongelmina. Arkistointi ongelmat johtuvat taas usein virheellisistä lähtötiedoista tai virheellisistä asetuksista. Tällä hetkellä Kanta-adapterin asetukset jakautuvat eri paikkoihin Mediatrin järjestelmäasetukissa ja siksi virheellisen asetuksen löytäminen on hankalaa.

Opinnäytetyön tavoitteena oli toteuttaa Kanta-adapterin parametrisointityökalu websovelluksena Mediconsult Oy:lle. Työkalun merkitys yritykselle on merkittävä. Se listaa ja validoi Mediatrin asetuksia, joita Kanta-adapteri tarvitsee Kanta-yhteyttä varten. Opinnäytetyössä kehitetty työkalu on tarkoitus saada käyttöön yrityksen tekniselle tuelle ja ohjelmistotuelle ongelmanratkaisun tueksi. Sovellus nopeuttaa heidän työskentelyään, kun virheellinen tiedon paikantaminen on nopeampaa.

Opinnäytetyössä tehtiin teoria osuus ja tekninen toteutus. Teoria osuus käsittelee käytettyjä työkaluja ja tekniikoita. Lisäksi teoria osuudessa esitellään lyhyesti tilaajana toiminut yritys ja opinnäytetyön osalta olennaiset yrityksen tuotteet. Näistä tuotteista Mediatri on järjestelmä, johon potilastieto kirjataan ja Kanta-adapteri huolehtii puolestaan Kanta-yhteydestä Kelalle ja tiedon tallentamisesta Kanta-palveluihin. Tuotteisiin ja niiden toimintoihin tutustuminen oli olennainen osa työtä, jotta työkalun toteuttaminen oli mahdollista. Ohjelmointikielinä käytettiin palvelinpuolella Javaa ja käyttöliittymä toteutettiin Angularilla.

Tekninen osuus toteutettiin Rest-pohjaisena websovelluksena. Työkalun tarkoitus on kerätä kaikki Kanta-adapterin kannalta tärkeä asetukset ja koodistot websovellukseen, josta pääkäyttäjät näkevät nopeasti virheelliset tai puutteelliset parametrit. Käyttöliittymä huomauttaa käyttäjälle virheellisistä tiedoista väreillä. Näin käyttäjän on mahdollista löytää virheellinen tieto nopealla silmäilyllä.

Avainsanat
Websovellus, Rest, Potilastiedon arkisto, sovelluskehitys

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Laura Kemppainen			
Title of Thesis Parametrization Tool for Kanta Adapter			
Date	15 December 2019	Pages/Appendices	45
Supervisor(s) Mr. Jukka Kinnunen, Senior Lecturer and Mr. Sami Lahti, Senior Lecturer			
Client Organisation /Partners Mediconsult Ltd			

Abstract

Kanta connection is an essential part of a modern medical report system, because more and more documents and prescriptions are aimed to be stored in MyKanta. Via MyKanta the patient can examine his/her medical reports. These reports can be retrieved to every medical report system that has established the Kanta connection. Problems with the Kanta adapter usually occur as archiving problems. Archiving problems are usually results from incorrect initial data or settings. At the moment the settings of the Kanta adapter locate in different places in Mediatri system settings and this makes finding errors difficult.

The purpose of this theses was to implement a parametrization tool for the Kanta adapter as a web application. This project was commissioned by Mediconsult Ltd and the importance of this theses was significant. The tool lists and validates Mediatri's settings which are needed for the Kanta adapter to connect to Kanta. The tool created in this thesis is meant to be used for problem solving in the company's technical and software support. The application makes working more efficient because locating the reason behind Kanta connection errors will be found more quickly.

Both the theory part and the technical implementation were implemented in the thesis. The theory part explains used tools and techniques. Mediconsult as a company and its relevant products are introduced in the theory part. Of these products, Mediatri is an application in which patient records are saved. The Kanta adapter takes care of the Kanta connection to Kela. Familiarizing with the products and their functions was important to successfully create this new tool. Java was used as a software language and user-interface was created using Angular.

As a result of this thesis, the new tool was implemented as a Rest web application. With this new tool it is possible to collect all the relevant settings to one web application where main users can easily identify incorrect parameters. In this thesis a simple web page was created to collect relevant settings, codes and which alerts the user of incorrect data by using colours. This enables the user to find incorrect data with a quick glance.

Keywords
Web application, Rest, Patient records archive, Application development

SISÄLTÖ

1 JOHDANTO

Opinnäytetyö toteutettiin Mediconsult Oy:lle, joka tuottaa ratkaisuja sosiaali- ja terveydenhuoltoon. Opinnäytetyön tavoitteena oli luoda Rest-pohjainen websovellus, joka listaa käyttäjälle Kanta-arkistoinnin kannalta olennaiset asetukset ja validoi ne. Puutteelliset ja virheelliset asetukset estävät arkistoinnin, minkä vuoksi virheellisen tiedon löytäminen on tärkeää. Työkalun tavoitteena on nopeuttaa virheellisen tiedon paikantamista arkistointiongelmien yhteydessä.

Työn kannalta olennaiset yrityksen tuotteet ovat potilastietojärjestelmä Mediatri ja potilastiedon arkistoinnista vastaava Kanta-adapteri. Mediatriin lääkärit tai hoitajat kirjaavat potilastiedon ja Kanta-adapteri hoitaa tietojen lähettämiseen Kanta-palveluun. Kanta-adapteri muokkaa Mediatrissa tallennettua tietoa kansallisesti määritettyyn rakenteiseen muotoon, jotta arkistoitavat asiakirjat ovat ladattavissa myös toiseen potilastietojärjestelmään. Rakenteiseen muotoon on määritetty yhteisesti käsitteet ja luokitukset, joiden avulla kerran kirjattua tietoa voidaan hyödyntää laajemmin (Terveyden ja hyvinvoinnin laitos (THL), 2018).

Potilastiedon arkisto on osa Kanta-palveluita, joka sisältää Potilasasiakirjojen arkistointipalvelun ja Tiedonhallintapalvelun. Potilaan keskeiset terveystiedot, suostumukset, kieltojen sekä tahdonilmaisujen hallintapalvelut ovat osa Tiedonhallintapalvelua. Sähköiset lääkemääräysten tallentamisesta vastaa Resepti-palvelu. Kansalaisilla on mahdollisuus tarkastella kaikkia näitä tietoja Omakanta-palvelun kautta. Potilastiedon arkiston on tarkoitus toimia keskitettynä pitkäaikaisarkistona, jonka tiedot ovat terveydenhuollon ammattihenkilöiden käytettävissä. Palvelun tarkoitus on parantaa potilasturvallisuutta ja samalla estää päällekkäisten tutkimusten tarvetta, kun edellisen tutkimuksen tiedot ovat kaikkien ammattilaisten käytettävissä. Lisäksi hoitoketjun toimivuus parantuu julkisen ja yksityisen palveluntuottajan välillä. (Terveyden ja hyvinvoinnin laitos (THL), 2018)

Työn olennaiset vaiheet ovat Mediatriin ja Kanta-adapteriin tutustuminen, käytettäviin tekniikoihin perehtyminen ja websovelluksen toteutus Javaa ja Angularia hyödyntäen. Työ mahdollistaa websovelluksen kehittämisen alusta loppuun ja se tarjoaa hyvän kokonaiskuvan vaiheista, joita websovelluksen kehittäminen vaatii. Lisäksi se tarjoaa mahdollisuuden tutustua terveydenhuollon järjestelmiin, joille on määritetty tiukat vaatimukset, jotka tietojärjestelmätoimittajien on huomioitava.

2 LYHENTEET JA MÄÄRITELMÄT

Ammattioikeuspalvelu	Sisältää tiedot sosiaali- ja terveysalalla työskentelevien ihmisten ammattioikeuksista. Sen avulla voidaan tarkastaa, että ammattilaisella on oikeus tehdä kyseinen toimenpide (esimerkiksi reseptin allekirjoitus)
AvoHilmo	Perusterveydenhuollon avohoidon hoitoilmoitus
Backend-koodi	Backend-koodi ajetaan sovelluksen palvelimella. Backend-koodissa voidaan suorittaa esimerkiksi lomakkeiden käsittely tai salasanan tarkastus
CDA	XML-pohjainen merkintästandardi, joka määrittelee kliinisten asiakirjojen merkkaustavan, rakenteen ja semantiikan
CDA R2	HL7 ja Kelan hyväksymä asiakirjarakenne. Sisältää rakennemäärittelyt Kanta-arkistointia varten
CRUD	Lyhenne ohjelmoinnissa käytettävistä perusfunktioista: Luo (Create), lue (Read), päivitä (Update) ja poista (Delete)
Debug	Koodin suorittaminen rivi kerrallaan mahdollisen virheen paikantamiseksi
DOM-menetelmät	Dokumenttioliomalli, joka kuvaa rakenteisen dokumentin puuna
Duodecim	Lääkäriseura, jonka asiantuntijat ylläpitävät terveyskirjastoa
eArkisto	Kanta-palvelun Sähköinen potilastiedon arkisto
eResepti	Kanta-palveluiden sähköinen lääkemääräys
Frontend-koodi	Frontend-koodi suoritetaan verkkoselaimessa. Esimerkiksi HTML- ja ulkoasu ovat frontend-koodia
Hilmo	Sosiaali- ja terveydenhuollon hoitoilmoitusjärjestelmä
HL7	Yhdysvaltalainen organisaatio Health Level Seven Inc., joka kehittää tietojärjestelmien standardeja terveydenhuoltoon
HL7v3	HL7 määrittämä sanomarajapinta

ISO 8061	Päivämäärän ja ajan esittämistä varten kansainvälinen standardi
Java EE	Java Enterprise Edition on tarkoitettu palvelinsovellusten kehittämiseen. Se sisältää web-palvelintekniikoita
JavaScript	Yleensä webympäristöissä käytettävä ohjelmointikieli, jolla voidaan lisätä sivulle dynaamista toiminnallisuutta
Java SE	Java Standard Edition sisältää yleiset ominaisuudet, graafiset käyttöliittymät, tietokanta- ja perus-XML-rajapinnat
JDBC	Javan ohjelmointirajapinta, joka hallitsee yhteyksiä asiakkaan ja tietokannan välillä
Jira	Projektin hallintaan kehitetty ohjelmisto
Kanta	Kelan ylläpitämä Kansallinen terveysarkisto, jonne tallennetaan esimerkiksi potilastiedot ja lääkereseptit
Ketterä kehitys	Ohjelmistotuotannossa käytettävä menetelmä, jossa tuotekehityksen suunnittelu ja toteutus tapahtuu lyhyissä sykleissä
Klusteripalvelin	Tietokone, joka toimii palvelimena ja jakaa tehtäviä muiden tietokoneiden välillä
Kontekstipalvelin	Kansallisesti määritettyjä avain-arvo pareja välittävä työpöytäintegraatio
Koodistopalvelu	Terveyden ja hyvinvoinnin laitoksen (THL) koodistopalvelu, joka sisältää koodistoja terveys- ja sosiaalipalveluiden käyttöön. Sen tehtävä on varmistaa tietorakenteiden laatu yhteisillä koodistoilla
Medikanta	Hakee potilaan tiedot Potilastiedon arkistosta
NodeJS	Avoimeen lähdekoodiin perustuva JavaScript runtime-ympäristö, joka huolehtii JavaScript-koodin suorittamisesta
Object Pascal	Pascal-ohjelmointikielen objektiorientoitunut johdannainen
OID-yksilöintitunnus	Tunniste, joka yksilöi tietyn objektin (esimerkiksi koodiston tai organisaation)

Otsikkorekisteri	Sisältää Mediatrissa käytettävät otsikot ja niiden parametrisoinnit
ORM	Object-Relational Mapping, joka mahdollistaa sovelluksen malliobjektien yhdistämisen relaatiotietokantaan ja päinvastoin
Palvelunjärjestäjä	Palvelunantaja, joka järjestää sosiaali- ja terveydenhuollon palveluita (asiakasrekisterinpitäjä)
Palveluntuottaja	Tuottaa sosiaali- ja terveyspalveluita
Rakenteinen muoto	CDA-asiakirjan muoto, jossa tieto esitetään sellaisessa muodossa, jonka tietojärjestelmä ymmärtää (esimerkiksi koodeina tai XML-muoto)
SAX-kirjasto	Käsittelyrajapinta XML-muotoiselle tiedolle
TypeScript	Avoimeen lähdekoodiin perustuva ohjelmistokieli, joka lisää ominaisuuksia JavaScriptiin
Varmennepalvelu	Mahdollistaa terveydenhuollossa toimivien henkilöiden luotettavan tunnistautumisen ja asiakirjojen allekirjoittamisen
Viestijonot	Kommunikaatiotapa eri prosessien välillä
Web-palvelut	Ohjelmistojärjestelmä, joka mahdollistaa vuorovaikutukset yhteensopiville tietokoneille tietoverkon yli
XML	Rakenteellinen kuvauskieli, jonka avulla voidaan esittää suuria tietomääriä selkeämmin

3 TOIMEKSIANTAJA

Opinnäytetyön toimeksiantaja on Mediconsult Oy, joka toimittaa tietojärjestelmiä sosiaali- ja terveydenhuoltoalan yrityksille ja toimijoille. Yritys on vuonna 1975 perustettu perheyritys. Tällä hetkellä yritys työllistää yli 150 työntekijää Suomessa ja Sri Lankassa. Toimipisteitä on viidellä eri paikkakunnalla Suomessa ja Sri Lankan Colombossa, jossa sijaitsee yrityksen tytäryhtiö MC Medisoft. Suomen toimipisteet sijaitsevat Helsingissä, Kuopiossa, Salossa, Joensuussa ja Oulussa. (Jaakkola, 2019)

Yritys tuottaa palveluita niin suurille sairaanhoitopiireille kuin pienemmille toimijoille yksityisellä ja julkisella sektorilla. Järjestelmän käyttäjiä ovat lääkärit, hoitajat sekä sosiaali- ja terveydenhuollon-asiakkaat, jotka ylläpitävät itse terveystietojaan. Palveluita on kehitetty suurien sairaanhoitopiirien ja pienten toimijoiden lisäksi sosiaalihuoltoon, kuntoutukseen, kotiin vietäviin palveluihin ja suunterveydenhuoltoon. Mediconsult Oy:ssa käytetään ketterää ohjelmistokehitystä ja kehitys tapahtuu yhdessä asiakkaiden kanssa. Yritys noudattaa Terveyden ja hyvinvoinnin laitoksen (THL) kansallisia määräyksiä pyrkien korkeaan laatuun ja prosessien jatkuvaan parantamiseen. (Mediconsult Oy)

Mediconsult Oy:n ydinosaaminen rakentuu asiakas- ja potilastiedon, toiminnanohjauksen, omahoidon ja sähköisen asioinnin ratkaisujen pohjalle. Asiakkaille pyritään tuottamaan kustannustehokkaita, helppokäyttöisiä sekä kilpailukykyä ja asiakastietoisuutta parantavia ratkaisuja. Yrityksen keskeisimmät ratkaisut ovat asiakas- ja potilastiedon hallinta, tiedolla johtaminen, toiminnanohjaus ja optimointi, sähköinen asiointi, omahoito, mobiilikirjaaminen ja sähköinen resepti. Näiden lisäksi Mediconsult Oy tarjoaa koulutus, konsultointi, projektinhallinta, käyttöpalveluita ja pilvipalveluita tukemaan toimijoita sosiaali- ja terveydenhuollossa. (Mediconsult Oy)

Asiakkuudenhallinta mahdollistaa asiakastiedonhallinnan niin terveydenhuollossa kuin sosiaalihuollossa. Ratkaisu mahdollistaa potilasturvallisen kirjaamisen, kun kirjaaminen voidaan tehdä tietokoneella tai mobiilisti paikasta riippumatta. Se sisältää muun muassa raportointityökaluja, graafisia seurantanäkymiä, sairauskokonaisuuksia, osastokarttoja ja rakenteiset potilaskertomukset. (Mediconsult Oy) Mobiilikirjaamisen myötä tiedot voidaan kirjata heti asiakkaan luona ja tiedot ovat näin aina ajantasaiset. Mobiiliin voidaan kirjata esimerkiksi mittauksia, potilaan lääkitys sekä muita kirjauksia, jotka tulevat käynnin aikana ilmi. (Mediconsult Oy)

Asiakkuuden hallintaan voidaan liittää myös sähköinen asiointi. Palvelun avulla esimerkiksi ajanvaraus voidaan hoitaa sähköisesti. Ajanvarauksen jälkeen voidaan asiakkaalle lähettää tekstiviesti vahvistukseksi. Potilas voi lisäksi täyttää esimerkiksi esitietolomakkeet ennen vastaanottoa tai vastata kyselyihin, joilla voidaan seuloa ne henkilöt, jotka tarvitsevat kiireellisintä hoitoa. Sähköisen asioinnin hyviä puolia on myös se, että asiakaspalvelua voidaan tarjota ympäri vuorokauden. (Mediconsult Oy)

Sähköinen resepti on nykyisin pakollinen ominaisuus ja se on korvannut paperiset reseptit vuodesta 2017 lähtien. Mediresepti voidaan liittää minkä tahansa potilastietojärjestelmän rinnalle. Se mahdollistaa reseptien luomisen, muokkaamisen, uusimisen ja tarkastelun yhdessä palvelussa. Palvelua voivat käyttää eri kokoiset organisaatiot ja itsenäiset ammatinharjoittajat. (Mediconsult Oy)

Toiminnanohjaus tarjoaa ratkaisuja, jotka auttavat hallinnoimaan resursseja ja tarjottavia palveluita. Se mahdollistaa reaaliaikaisen seurannan, jonka avulla resurssit ja palvelutarpeet voidaan määrittää helposti, uudelleen järjestellä ja kohdentaa haluttuun aikaan. Toiminnanohjausta voidaan hyödyntää esimerkiksi kotiin vietävissä palveluissa ja kuntoutuksen kurssienhallinnassa. Se mahdollistaa kattavan raportoinnin ja seurannan käytettävän palvelun osalta ja ratkaisut voidaan sovittaa jokaisen organisaation tarpeisiin. (Mediconsult Oy)

Omahoito mahdollistaa esimerkiksi kotimittausten, kyselyiden, testien ja riskikartoituksen hyödyntämisen luotettavasti ja tietoturvallisesti. Se mahdollistaa potilaalle pääsyn ammattilaisen merkitsemiin terveystietoihin, joita ovat muun muassa voimassa oleva lääkitys tai otetut laboratorio tutkimukset. Vastavuoroisesti ammattilaisen on mahdollista nähdä potilaan omat merkinnät potilastietojärjestelmän kautta. Omahoito mahdollistaa juuri ennaltaehkäisevän hoidon esimerkiksi riskiryhmiin kuuluville. Riskiryhmään kuuluvien potilaiden tekemä omahoitoseuranta mahdollistaa tarkastusten ajoituksen tai tarvittavan ajanvarauksen ammattilaiselle oikeaan aikaan. Omahoito palveluun voidaan liittää vertaistukiportaali, joka mahdollistaa keskustelun muiden samassa tilanteessa olevien kanssa. (Mediconsult Oy)

Tiedolla johtaminen käsittää kliinisen työn näkymiä ja raportointiin liittyviä työkaluja. Niitä hyödyntävät sekä palvelunjärjestäjät ja palveluntuottajat sosiaali- ja terveydenhuollon organisaatioissa. Kliinisen työn työkalut kasaavat kaikki potilaan tiedot visuaalisesti ja helposti tarkasteltavaan muotoon. Tämä mahdollistaa esimerkiksi nopean kokonaiskuvan muodostamisen asiakkaan terveydentilasta. Ammattilaisen on näin helppo rakentaa optimaalinen hoitopolku asiakkaalle ja poistaa esimerkiksi päällekkäisyyksiä käytetyistä lääkkeistä. Raportoinnin työkaluja hyödynnetään tuotetun tiedon jalostukseen organisaation ja viranomaisten hyödyksi. (Mediconsult Oy)

3.1 Mediatri

Mediatri on potilastietojärjestelmä, jonne kirjataan potilastietoja, minkä kautta käyttäjät käsittelevät kirjattua tietoa. Sitä voidaan hyödyntää esimerkiksi tavallisella lääkärinvastaanotolla, neuvoloissa, hammashuollossa ja laboratorioissa. Mediatri on Kanta-yhteensopiva, se mahdollistaa eResepti- ja eArkisto-toiminnallisuudet Kanta-adapteria hyödyntäen. (Mediconsult Oy) Ohjelmistoon kerran kirjattu tieto siirtyy suoraan sen eri osioihin kuten lähetteisiin tai lausuntoihin. Lisäksi siihen on pyritty rakentamaan automatiikkaa, joka nopeuttaa tietojen kirjaamista. Automatiikka mahdollistaa joidenkin kenttien täytön lähtötietojen tai juuri kirjatun tiedon perusteella. Lisäksi Mediatri kokoaa kaikki viranomaisraportit kuten AvoHilmo- ja Hilmo-raportit kirjaamisen yhteydessä annetuista tiedoista. (Mediconsult Oy) Raporteilla kerätään tilastotietoa terveydenhuollon toiminnasta (Rissanen, 2019).

Tiedon kirjaamisen lisäksi ohjelmisto tarjoaa tukea ammattilaisille hoitotyöhön. Mediatrissa on integraatio Duodecimin tietokantoihin, joka mahdollista muistutusten, lääkkeiden yhteisvaikutusvaroitus-

ten ja hoitosuositusten tuottamisen automaattisesti sidottuna kyseisiin hoitotilanteeseen. Ammattilaisen tuen lisäksi Mediatri tarjoaa myös potilaalle mahdollisuuden saada oleelliset potilastiedot Medinet-verkkosivustolla tarkasteltavaksi ajasta tai paikasta riippumatta. (Mediconsult Oy)

Ohjelmisto toimii Windows-työasemalla ja tarvitsee yhteyden tietokantaan ODBC-tietokannan kautta. (Kolmakow, 2016) ODBC-tietokannan (Eng. Open Database Connectivity) tehtävänä on toimia rajapintana tiedonvälityksessä sovellusten ja tietolähteiden välissä (Craig;ym., 2017). ODBC yhteyden lisäksi Mediatri vaatii yhteyden Kanta-adapteriin, Medikantaan sekä kontekstipalvelimelle http-protokollan avustuksella. Kaikki Mediatriin lisättävä tieto lisätään tietokantaan, josta myös muut edellä mainitut palvelut voivat hyödyntää niitä. Lisäksi tietokantaan tallennettuja tietoja voidaan hyödyntää, jos yhteys Kanta-palveluihin on poikki, mutta tällöin uusia tietoja ei arkistoida palveluun. Kaikki Mediatrin tieto tallennetaan SolidDB-tietokantaan, josta Kanta-adapteri hakee tarvittavat tiedot asiakirjan lähetystä varten Kanta-palveluihin. Ohjelmisto on kehitetty Delphi-ohjelmointia hyödyntäen (Esitelty tarkemmin luvussa 4.1). (Kolmakow, 2016)

Mediatri mahdollistaa sähköisen lääkemääräyksen ja niiden allekirjoittamisen. Allekirjoittamista varten Mediatri vaatii ohjelmiston, joka lukee sirukortin. Tällaisia ohjelmistoja ovat esimerkiksi mPollux ja DigiSign Client. Lisäksi allekirjoittamista varten tarvitaan Javan 32-bittinen Runtime Environment (JRE), johon perehdytään tarkemmin opinnäytetyön myöhemmässä vaiheessa. (Kolmakow, 2016)

Viime vuosina yritys on alkanut kehittämään selainpohjaista sovellusta Medicloudia, jonka tarkoitus on myöhemmin korvata Delphillä kehitetty Mediatri. Medicloudissa hyödynnetään selainohjelmoinnissa käytettäviä tekniikoita kuten Angular, JavaScript ja TypeScript. (Kolmakow, 2019)

Omat haasteensa Mediatriin yli neljänsadan tietokantataulun tietokanta, jonka rakenne ei vastaa perinteisiä tietojärjestelmien tietokantoja. Tietokannassa ei ole perinteisiä kenttäkäsitettä, joissa esimerkiksi osoitetieto tallennetaan aina osoitekenttään. Mediatrin konfiguroitavuus mahdollistetaan juuri tietokantarakenteen avulla, jossa tietoja voidaan tallentaa tietokantaan juuri halutuilla parametrisoitavilla arvoilla. Tiedot yhdistetään toisiinsa tietokantarakenteeseen määritetyllä omistaja-, osa- ja numerotiedoilla, joiden avulla voidaan linkittää tietoa esimerkiksi eri tietokantataulujen välillä.

Terveydenhuoltoon liittyvät prosessit ja käytännöt eivät olet vielä tänäkään päivänä täysin standardoituja, mikä aiheuttaa potilastietojärjestelmien räätälöinnin tarpeen asiakkaan tarpeen mukaan. Osa syy standardien puuttumiseen on myös se, että potilastietojärjestelmiä on käytettävissä useita, joista jokainen toimii hieman eri tavalla ja jokaisen järjestelmä on rakennettu eritavoin. Tämä tekee jokaisesta järjestelmästä omanlaisensa monimutkaisen kokonaisuuden, joka on pystyttävä räätälöimään jokaisen asiakkaan tarpeisiin. (Rissanen, 2019)

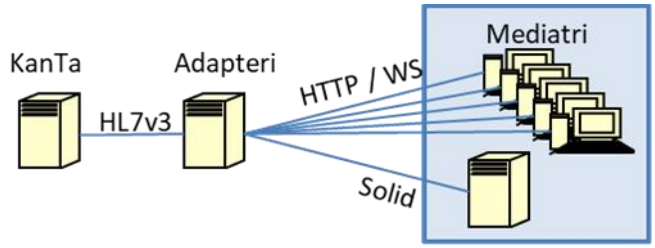
Mediatrissa konfiguroitavia osia ovat ympäristöön liittyvät osiot, kuten osoitteet ja niihin liittyvät parametrit. Lisäksi voidaan konfiguroida esimerkiksi otsikoiden asetukset ja se minkä tyyppisenä tietoa tallennetaan ja mitä lomakkeita käytetään. Usein näiden tietojen lisäksi konfiguroidaan samalla lomakkeiden sisällöt. (Rissanen, 2019)

3.2 Kanta-adapteri

Mediatri on liitetty Kansalliseen terveysarkistoon eli Kanta-palveluihin adapteripalvelun avulla. Kanta-palveluihin kuuluvat sähköinen lääkemääräys (eResepti) ja sähköinen potilaskertomusarkisto (eArkisto). Adapterin ensisijainen tehtävä on tallentaa Mediatrin potilaskertomusaineisto CDA R2-muodossa ja lähettää tiedot Kanta-palveluun. (Mediconsult Oy) Lisäksi adapteripalvelun välityksellä voidaan hakea tietoa Kanta-palveluista. Tällaisia tietoja ovat esimerkiksi erilaiset kielto- ja suostumus asiakirjat (Kolmakow, 2016). Näiden lisäksi adapteripalvelu vastaa sanomamuodostuksesta ja -vastaanotosta. Adapteripalvelu voi vastaanottaa palvelupyyntöjä sekä Kanta-palveluista ja Mediatrilta. Jotta adapteripalvelu pystyy keskustelemaan Kanta-liitännän kanssa, se tukee Kelan ja HL7:n määrityksiä ja HL7v3-protokollaa. Näiden lisäksi se tukee oheispalvelujen määritettyjä protokollia. Tällaisia oheispalveluja ovat muun muassa koodistopalvelu, varmennepalvelu ja ammattioikeuspalvelu. (Mediconsult Oy)

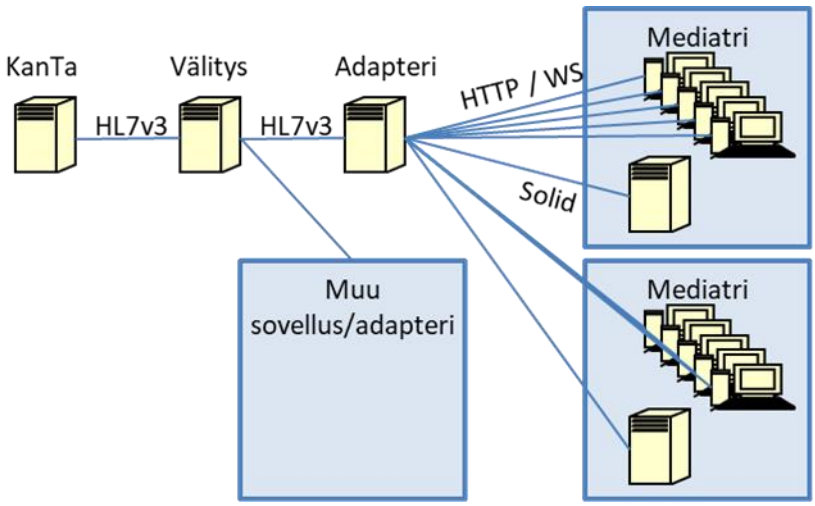
Kanta-adapterin pääasiallinen tehtävä on siis muodostaa XML-asiakirjoja Mediatrin antamista tiedoista ja lisäksi se muodostaa sanomat, joita tarvitaan Potilastiedon arkistoa varten. Kun adapteri on muodostanut kaiken tarvittavan tiedon Kanta-palveluita varten, se allekirjoittaa viestit sähköisesti hyödyntäen järjestelmäallekirjoitusta. (Kolmakow, 2016)

Kanta-liityntä voidaan tehdä Mediatrin ja adapteripalvelun välillä useilla eri tavoilla. Tapoihin voi vaikuttaa esimerkiksi organisaation koko tai se, jos alueella on useita Mediatri-järjestelmän käyttäjiä. Yksi adapteripalvelu voi huolehtia yhden tai useamman Mediatrin liittämisestä Kanta-palveluun. Liityntä voidaan tehdä suoraan Kanta-palveluun tai vaihtoehtoisesti adapteripalvelun ja Kanta-palvelun välille voidaan asettaa välityspalvelu. Kuvassa 1 on esitetty yhden toimijan Mediatrin liitäntä Kanta-palveluun. Mediatri lähettävät tietoa adapteripalvelulle ja adapteri palvelu lähettää tiedot Kanta-palveluun HL7v3-protokollaa noudattaen. (Mediconsult Oy)



Kuva 1 Yhden toimijan suora Kanta-liityntä (Mediconsult Oy)

Kuvassa 2 on esitetty usean Mediatrin liitäntä Kanta-palveluun adapteripalvelun välityksellä. Tässä esimerkissä adapteripalvelun ja Kanta-palvelun välillä esiintyy vielä välityspalvelu, joka voi olla esimerkiksi muu sovellus tai adapteri. Esimerkissä kahdella toimijalla on yhteinen adapteripalvelu, joka välittää tietoja adapterille. Adapteripalvelu välittää tiedot välityspalvelulle ja välityspalvelu lopulta tiedot Kanta-palveluun. Tieto laitetaan jo adapteripalvelussa HL7v3-protokollan mukaiseen muotoon ja välityspalvelun tehtävä on vain siirtää tieto Kanta-palveluun. Adapteripalvelun lisäksi välityspalvelulle tulee tietoa toisesta adapterista tai sovelluksesta, joista tietoa välitetään myös Kanta-palveluun. Välityspalvelun tehtävä on koota tiedot kaikista sovelluksista ja/tai adaptereista, jolloin toimija tai alue ei tarvitse useaa Kanta-liitäntäpistettä. (Mediconsult Oy)

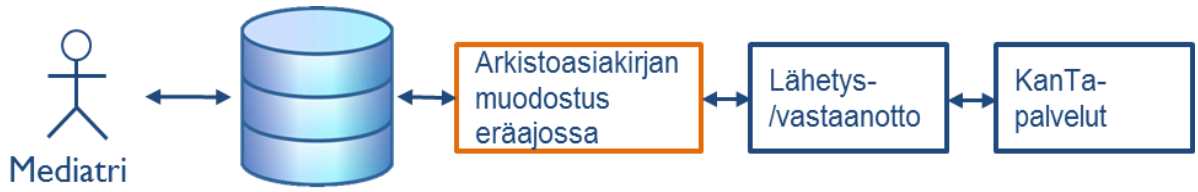


Kuva 2 Kanta-liitäntä välityspalvelun avulla, jossa usealla Mediatri-järjestelmällä on yhteinen adapteripalvelu

Se, mikä liitäntätapa adapteripalvelulle valitaan, täytyy aina katsoa jokaisen toimijan tai organisaation mukaan. Pienet toimijat, joilla on käytössä esimerkiksi vain Mediatri-järjestelmä, on helpointa liittää Kanta-palveluun kuvan yksi mukaan. Tällöin toimijalla ei ole tarvetta esimerkiksi liittää muita sovelluksia Kanta-palveluun. Jos toimijoita on useita ja heillä on esimerkiksi jokin muu sovellus, joka tarvitsee Kanta-palveluun liitännän olisi helpointa liittää kuvan kaksi mukaan. Yhteinen adapteripalvelu vähentää ensinäkin ylläpidon tarvetta ja välityspalvelun avulla toimijoiden ei tarvitse pitää yllä useaa eri Kanta-liityntäpistettä. (Mediconsult Oy)

Kanta-adapteri hyödyntää omaa MySQL-tietokantaa, jonne talletetaan adapterin tarvitsemat asetustiedot, ammattioikeustiedot ja raportit adapterin toiminnasta. Lisäksi se pitää yllä lokitietoa Kanta-adapterista lähetyistä ja vastaanotetuista viesteistä. MySQL-tietokannan lisäksi se tarvitsee toimiakseen WildFly-sovelluspalvelimen, johon perehdytään opinnäytetyössä myöhemmin eri tekniikoiden kohdalla. (Kolmakow, 2016)

Yksinkertaisuudessaan Kanta-adapteri arkistoi tietoa pääsääntöisesti eräajona. Eräajossa tietokoneelle annetaan usean eri tehtävän tiedot, jotka se sitten suorittaa annetussa järjestyksessä. Kun käyttäjä kirjaa tiedot Mediatriin, hän lukitsee tiedot ja tietokantaan tehdään merkintä siitä, että tiedot ovat valmiita arkistoitavaksi. Adapterin tehtäväksi jää varmistaa, että tiedot voidaan arkistoida, muodostaa tarvittavat HL7-sanomat ja lähettää ne Potilastiedon arkistoon. Lisäksi Kanta-adapteri hoitaa tietojen allekirjoituksen omalla järjestelmäallekirjoituksellaan. Samalla tallennetaan tietoja lähetyistä sanomista Kanta-adapterin MySQL-tietokantaan ja Mediatrin SolidDB tietoihin talletetaan arkistoinnin yhteydessä syntyviä tietoja (Kuva 3). (Kolmakow, 2016)



Kuva 3 Tietojen arkistointi Kanta-palveluihin (Kolmakow, 2016)

3.3 Toimeksiantajan tarve työkalulle

Toimeksiantajan näkökulmasta katsottuna opinnäytetyössä toteutetun parametrisointi työkalun tarve oli merkittävä. Tällä hetkellä Mediatri kerää ohjaustietoa useaan eri tietokantaan (SolidDB ja MySQL). Kanta-adapteri tarvitsee tiedon välitykseen ja käsittelyyn ohjaustietoa, joten työkalun tarkoitus on kerätä nämä tiedot samaan paikkaan. Näin asetetut parametrit ja niiden arvo on helpompi selvittää, kun tieto on kerätty samaan paikkaan. (Kolmakow, 2019)

Sen lisäksi että Kanta-adapteri kerää ohjaustietoa useasta eri paikasta, on myös Mediatrin sisällä ohjaustieto jakautunut loogisesti useisiin lomakkeisiin. Ohjaustieto on toisin sanoen parametrisoitua tietoa ja tämä tieto voidaan sijoittaa usealla eri tavalla järjestelmään konfiguraation avulla. Järjestelmän kasvaessa konfiguroinnista on muodostunut ongelmia esimerkiksi virheellisten parametrien löytämiseen, jos asiakirjan arkistointi on epäonnistunut. (Rissanen, 2019)

Henkilöstön ja lomakkeiden määrän kasvaessa tämän ongelman ratkaisulle on tullut entistä merkittävämpi tarve. Konfiguroitava ja jatkuvasti laajentuva järjestelmä on aiheuttanut sen, että uusien työntekijöiden on yhä hankalampi löytää tarvittavat parametrit. Ohjaustiedon kokoaminen yhteen paikkaan helpottaa täten järjestelmäkonfiguraation hallintaa. (Rissanen, 2019)

4 KÄYTETYT TEKNIIKAT

Mediconsult Oy on tehnyt käytettävien tekniikoiden osalta valintoja niin, että ohjelmistokehityksessä voidaan keskittyä bisneslogiikkaan. Tämän vuoksi yritys käyttää polygot-ohjelmointia (Eng. Polygot Programming), jota hyödyntämällä voidaan valita parhaat työkalut jokaiseen ongelmaan. (Kolmakow, 2019) Polygot-ohjelmoinnissa hyödynnetään useita eri ohjelmistokieliä, jotta voidaan mahdollistaa lisätoimintojen käyttö ja koodia voidaan toteuttaa tehokkaammin. (Rouse, 2015) Eri tekniikat yhdistyvät toisiinsa vain tietokannan ja rajapintojen kautta. Yhteiset rajapinnat yhdistävät erilliset komponentit ja sovellukset toisiinsa, joiden myötä syntyy yksi kokonaisuus, jonka kehitys on tapahtunut bisneslogiikkaan panostaen. (Kolmakow, 2019)

4.1 Delphi

Alkujaan Mediatri on kehitetty hyödyntäen Delphi-ympäristöä. Mediatrin alkukehityksen aikaan ympäristössä oli helppo kehittää sovelluksia Windowsin käyttöjärjestelmälle ja sillä voitiin tehdä nopeasti koodia graafisen ohjelmankehitysympäristön avulla. Delphiä käytetään ohjelmistokehityksessä edelleen joissain määrin, koska vanha Mediatri pohjautuu siihen. (Kolmakow, 2019)

Delphi perustuu Object Pascal-ohjelmointikieleen, joka tukee olio-ohjelmointia. Sen vahvuutena on nopea kääntyminen, vaikka koodia olisi paljon. Tämä mahdollistaa käytön myös koneella, jolla on rajoitetut resurssit esimerkiksi muistin suhteen. Delphi on alun perin julkaistu vuonna 1995 Windows-sovelluksia varten. (Penland, 2018)

Delphin avulla voi luoda graafisia- tai konsolisovelluksia. Se perustuu valmiisiin komponentteihin, joita ohjelmoija siirtää hiiren avulla käyttöliittymään ja rakentaa näin sovelluksen. Hiirellä voi määrittää myös komponenttien koon ja sijainnin. Elementit voivat olla passiivisia, eli ne voivat näyttää esimerkiksi tekstiä tai ne voivat olla aktiivisia nappeja, jotka käyttäjä aktivoi hiirtä tai näppäimistöä hyödyntäen. Näin Delphi mahdollistaa nopean koodin tuottamisen ja sen avulla voidaan vähentää virheitä integroidun virheenkorjauksen avulla. (Khatri)

4.2 Java

Java on ohjelmointikieli, jota käytetään moneen kehittämiseen. Se on yleinen mobiilisovellusten kehittämisessä, web-sovellusten parissa ja lisäksi myös työpöytäsovelluksissa sekä pelien ohjelmoinnissa (w3schools). Javan on kehittänyt James Gosling vuonna 1995. Hän työskenteli kehittäessään Sun Microsystems yhtiössä, joka on nykyisin Oraclen omistama. (Leahy, 2019)

Java-ohjelmien suorittamien ja kehittäminen tarvitsee taustalle Java Runtime Environment (JRE)- ja Java Development Kit (JDK)-palvelimet. Java Runtime Environmentin käyttäjiä ovat järjestelmien loppukäyttäjät. Se on paketti, joka mahdollistaa Java-ohjelmien suorittamisen tietokoneella. Java Development Kit on tarkoitettu ohjelmistojen kehittäjille. Sen mukana ovat kehitystyökalut, jotka mahdollistavat Java-ohjelmien kehittämisen ja testauksen. Lisäksi siinä on mukana myös Java Runtime Environment, jotta kehittäjällä on mahdollisuus testata ohjelmistoa kehitysvaiheessa. Näiden

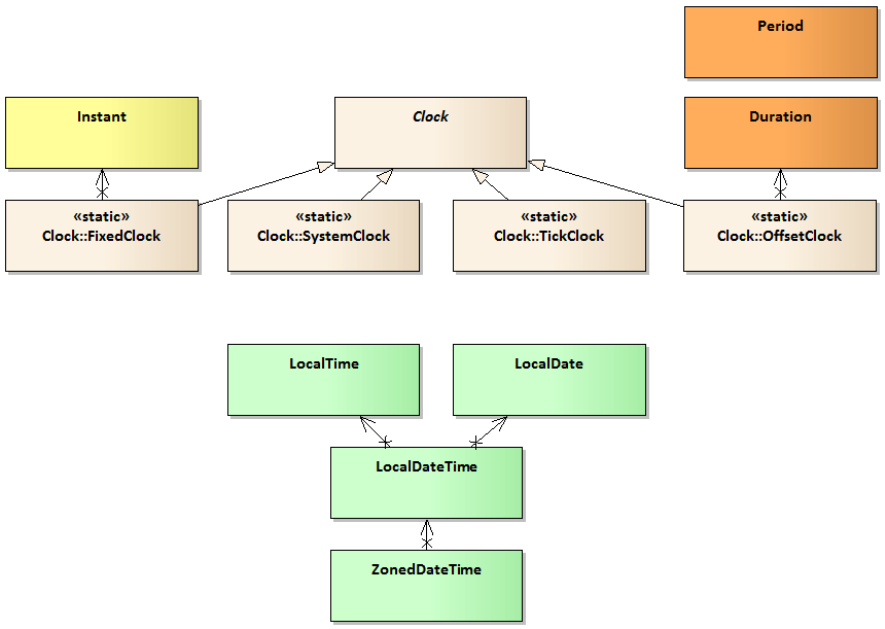
lisäksi on olemassa Java Virtual Machine (JVM), joka on rakennettu edellisten pakettien sisään. Sen tehtävänä on suorittaa ohjelma rivi kerrallaan, kun ohjelmistoa ajetaan Java Runtime Environmentin tai Java Development Kitin kautta. (Bhatia)

Java on valikoitunut myöhemmässä vaiheessa Mediconsult Oy:n ohjelmistokehitykseen backend-koodin kehittämiseen. Muutos tuli vuonna 2007, jolloin Java oli yksi johtavista kielistä web-sovelluskehityksessä. Java tarjosi jo silloin paljon kirjastoja sekä valmiita sovelluspalvelimia. (Kolmakow, 2019) Valmiit kirjastot mahdollistavat sen, että ohjelmointiin käytetty aika kulutetaan tehokkaasti. Jokaista koodiriviä ei tarvitse aina kirjoittaa itse, kun valmis kirjasto tekee sen ohjelmoijan puolesta. Lisäksi kirjastojen käyttö on ilmaista, minkä vuoksi niiden hyödyntäminen on helppoa. (Patsov, 2017)

Mediconsult Oy hyödyntää kirjastoista muun muassa Maven-, Mockito- ja JUnit-kirjastoja. JUnit- ja Mockito-kirjastoja hyödynnetään esimerkiksi yksikkötestien tekemiseen. Maven-kirjaston avulla voidaan taas hallita ohjelmistoprojektien riippuvaisuuksia ja kokoonpanoja. (Kolmakow, 2019) JUnit-kirjaston hyödyntäminen on yksi tapa tehdä yksikkötestejä. Sen avulla voidaan testata yksittäisiä metodeja tai luokkia niin että ulkoiset riippuvuudet poistetaan korvaamalla ne esimerkiksi mockatuilla objekteilla. JUnit-testit eivät kuitenkaan sovellu monimutkaisten metodien ja käyttöliittymien testaamiseen. (Vogella, 2007) Mockito-kirjaston avulla, voidaan testata jotakin luokkaa ja varmistaa, että luokka tai metodi toimii halutulla tavalla poistamalla esimerkiksi riippuvuuksia yksikkötesteistä. Mockito-objektit luodaan testikoodissa manuaalisesti ja sitä hyödyntämällä voidaan täyttää esimerkiksi metodin parametri luettelo. (Vogel;ym., 2012)

Näiden lisäksi yritys hyödyntää esimerkiksi XML-parsintaan käytettäviä kirjastoja sekä ajan ja päivän muodostukseen liittyviä rajapintoja (Kolmakow, 2019). XML-parsija mahdollistaa XML-asiakirjojen käytön tai tietojen muokkaamisen. XML-parsintaan on hyödynnettävissä useita eri kirjastoja. Esimerkiksi DOM-kirjasto lataa koko XML-tiedoston muistiinsa ja sen kanssa voidaan hyödyntää DOM-menetelmiä, kun taas SAX-kirjasto on tarkoitettu vain XML-asiakirjojen lukemiseen. (Tutorialspoint) DOM-menetelmät mahdollistavat asiakirjojen sisällön ja rakenteen tutkimisen (Tutorialspoint).

Javan versioon kahdeksan on kehitetty uusi kirjasto päivämäärien ja kellonaikojen käsittelyyn, jonka tarkoitus on korvata muut päivämäärä- ja aikakirjastot kuten Joda-Time-kirjasto tai Java Util-kirjastot. Uuden päivämäärä- ja aikakirjastojen tarkoitus on mahdollistaa paremmat sovellusliitännät tukemalla ISO-8061-aikaformaattia ja lisäksi se tukee myös esimerkiksi japanilaisten käyttämää kalenteria. Java kahdeksaan on luotu päivämäärä- ja aikaformaatti hyödyntäen ZonedDateTime- ja LocalDateTime-paketteja, joista osa sisältävät aikavyöhykkeet ja osa niistä jättävät aikavyöhykkeen määrittämättä (Kuva 4). Uudet päivämäärä- ja aikaformaatit hyödyntävät Java Time-paketteja, joiden välillä myös päivämäärät ja ajat on helppoa formatoida eri muotoihin yksinkertaisella konversiolla. (Evans;ym., 2014)



Kuva 4 Java kahdeksan aika- ja päivämääräkirjastojen käyttömahdollisuudet (Javin, 2018)

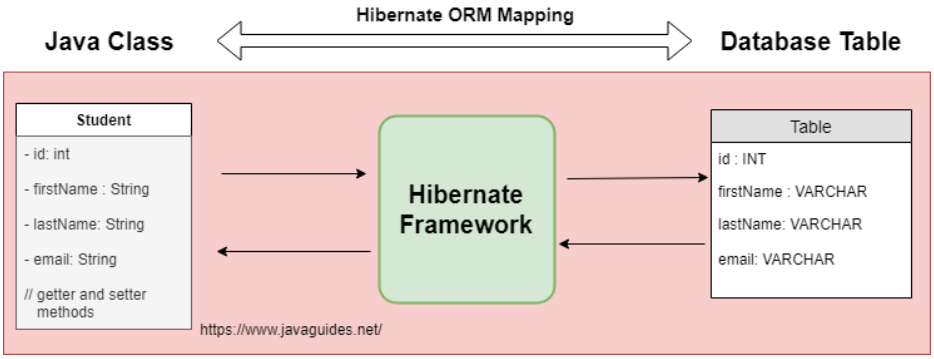
Edellä mainittujen syiden lisäksi Java valikoitui yrityksen käyttöön myös sen takia, että sitä pidetään helposti omaksuttavana ja osaajia on helppo löytää. Lisäksi Java noudattaa standardeja, jotka mahdollistavat esimerkeiksi REST-palveluiden hyödyntämisen. (Kolmakow, 2019)

Rest-palvelut tarjoavat arkkitehtuurimallin rajapintojen toteuttamiselle. Se mahdollistaa tiedon haun, muokkaamisen, lisäämisen ja poistamisen esimerkiksi tietokannasta. Rajapinta saa pyynnön yleensä asiakkaalta (Eng. Client), jonka jälkeen rajapinta suorittaa vain halutun toiminnon. Rest ei ota kantaa siihen, millaisessa formaatissa tieto tulee palvelulle, joten se pystyy esimerkiksi käsittelemään JSON tai XML formaattia. Restin etuna voidaan pitää sitä, että se pystyy käsittelemään Get-, Post-, Put- ja Delete-metodit, joilla voidaan helposti kuvata pyynnön luonnetta. Tästä syystä sitä pidetään parempana vaihtoehtona SOAP-arkkitehtuurille, jossa kaikki pyynnöt lähetetään post-metodien kautta. (Mikkonen, 2017)

4.3 Hibernate

Hibernate on Java-ympäristöön kehitetty ORM-ratkaisu (Object-Relational Mapping), joka mahdollistaa sovelluksen malliobjektien yhdistämisen relaatiotietokantaan ja päinvastoin (Kuva 5). Se mahdollistaa perinteisten CRUD-toimintojen käytön ja lisäksi se mahdollistaa vanhojen Java-objektien sekä XML-pohjaisen konfiguroinnin yhdistämisen tietokantatauluihin JPA-merkintöjä hyödyntämällä. (Fadatare, 2018)

JPA:ta eli Java Persistence APIa voidaan hyödyntää sovelluksissa relaatiotietojen hallitsemiseen sen tarjoamien määritelmien mukaan (Fadatare, 2018). Se on yksi tapa lähestyä ORM-ratkaisuja ja se mahdollistaa tietojen kartoituksen, tallennuksen, päivittämisen ja hakemisen relaatiotietokannoista Java-objekteihin ja päinvastoin. Sitä voidaan hyödyntää Java-EE ja Java-SE sovelluksissa. JPA tarjoaa kuitenkin vain määritelmät ORM-ratkaisujen pohjalle ilman toteutusta. Toteutuksia JPA:lle tarjoaa muun muassa juuri Hibernate sekä lisäksi EclipseLink ja Apache OpenJPA. (Fadatare, 2019)

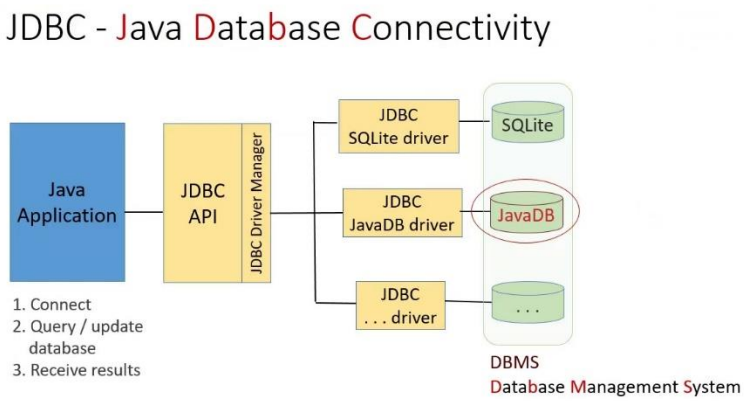


Kuva 5 ORM ratkaisu, joka yhdistää Javan Student luokan Student tietokanta tauluun Hibernatea hyödyntäen (Fadatare, 2018)

Hibernate hyödyntää JDBC:ta kaikessa tietokantaviestinnässä (Kuva 6) (Fadatare, 2018). JDBC tarjoaa tarvittavat standardit ja yhteydet Java-sovelluksen ja relaatiotietokantapalvelimien välillä. JDBC:n avulla voidaan hyödyntää relaatiotietokantoja, laskentataulukkoja ja tiedostoja. (Fadatare, 2018)

Kuva 6 Hibernate hyödyntää JDBC-tietokantayhteyttä tietokantaviestinnässä (Fadatare, 2018)

JDBC auttaa hallitsemaan kolmea eri ohjelmointitoimintoa, jotka ovat yhdistäminen tietolähteeseen kuten tietokantaan, kyselyjen lähettäminen ja tietojen päivittäminen tietokantaan sekä kyselyjen vastausten hakeminen ja käsittely tietokannasta. JDBC:n sovellusliittymään liittyy kaksi olennaista pakettia, jotka ovat java.sql ja javax.sql. Paketit ovat käytössä automaattisesti versiossa Java Platform Standard Edition (Java SE) 8. (Fadatare, 2018)



Kuva 7 JDBC:n mahdollistamat kolme ohjelmistotoimintoa (Fadatare, 2018)

JDBC API koostuu kahdesta eri osasta (Kuva 7). JDBC API on sovellusohjelmoijien hyödyntämä osa ja toinen osa on alemman tason sovellusliitäntä, joka hoitaa yhdistämisen tietokantapalvelimeen. Alemman tason sovellusliitäntää kutsutaan JDBC ajuriksi (Eng. JDBC Driver). JDBC Apia hyödynnetään, kun tietoa haetaan ja käsitellään tietolähteestä. JDBC-ajurit ovat Java-luokkia, joiden tarkoitus on toteuttaa JDBC-rajapinnat ja ne kohdistuvat tiettyyn tietokantaan. JDBC-rajapinnat saadaan tavallisen Java-version mukana ja rajapintojen toteutus on tehty tiettyyn tietokantaan, johon sovellusta ollaan yhdistämässä. (Fadatare, 2018)

Syitä siihen, miksi Hibernatea kannattaa hyödyntää Java-sovelluksissa, on monia. Hibernate hallinnoi resursseja ja poistaa 'boiler-plate' koodin (niin sanotun vakiokoodin), jotta sovelluksessa voidaan keskittyä vain liiketoimintalogiikkaan. Lisäksi se tukee XML- ja JPA-merkintöjä (Eng. JPA Annotations), minkä vuoksi koodin toteutus on riippumatonta. Hibernate tarjoaa kyselykielen (HQL), joka on hyvin samankaltainen, kuin SQL-kieli. HQL-kieli on kuitenkin täysin oliopainotteinen ja se ymmärtää olio-ohjelmoinnissa hyödynnettäviä käsitteitä kuten perintä ja assosiaatio. Hibernatea hyödynnetään ympäri maailmaa ja se perustuu avoimeen lähdekoodiin. Sen oppiminen on helppoa, koska siitä on saatavana paljon materiaalia ja ohjeita verkossa sekä keskustelupalstoilla. Lisäksi Hibernaten etuna voidaan pitää sitä, että se suorittaa kyselyt vain tarvittaessa eli tietokantakyselyitä ei suoriteta koskaan turhaan, mikä parantaa sovelluksen suoristuskykyä. (Fadatare, 2018)

4.4 Java EE-palvelimet

Sovellukset koostuvat kolmesta osiosta: käyttöliittymäpalvelin, tietokantapalvelin ja sovelluspalvelin. Sovelluspalvelimet suorittavat ohjelmaa ja mahdollistavat asennetun sovelluslogiikan käytön. Lisäksi Java EE-palvelimet määrittelevät sovelluspalvelimet. Niiden ensisijainen tehtävä onkin mahdollistaa sovellusten asentaminen, käyttäminen ja ylläpito. Java EE-palvelimia on tarjolla useita kuten Apache Tomcat, Jetty ja Glassfish. (Marshall, 2015) Näiden lisäksi tarjolla on myös WildFly-palvelin, jota Mediconsult Oy hyödyntää tuotekehityksessä.

Java EE-sovellukset koostuvat komponenteista, jotka kommunikoivat yhdessä. Ne ovat itsenäisesti toimivia ohjelmiston osia, joista kootaan Java EE-sovellus siihen liitetyistä Java-luokista ja -tiedostoista. Java EE-komponentit ovat nimensä mukaan kirjoitettu Java-ohjelmointikielellä. Nämä komponentit kootaan Java EE-sovellukseksi, jotka siirretään tuotantoon. Tuotannossa Java EE-palvelimet ylläpitävät ja hallinnoivat Java EE-sovelluksia. (Oracle)

Java EE-komponentit ovat määritetty kolmeen eri tyyppiin: asiakassovellus-, web- (Java Servlet, JavaServer Faces ja JavaServer Pages (JSP)) ja EJB-komponentit. Asiakassovellus-komponentit toimivat asiakassovelluksessa. Web-komponentit toimivat palvelimella ja EJB-komponentit sisältävät liiketoiminta komponentteja. (Oracle)

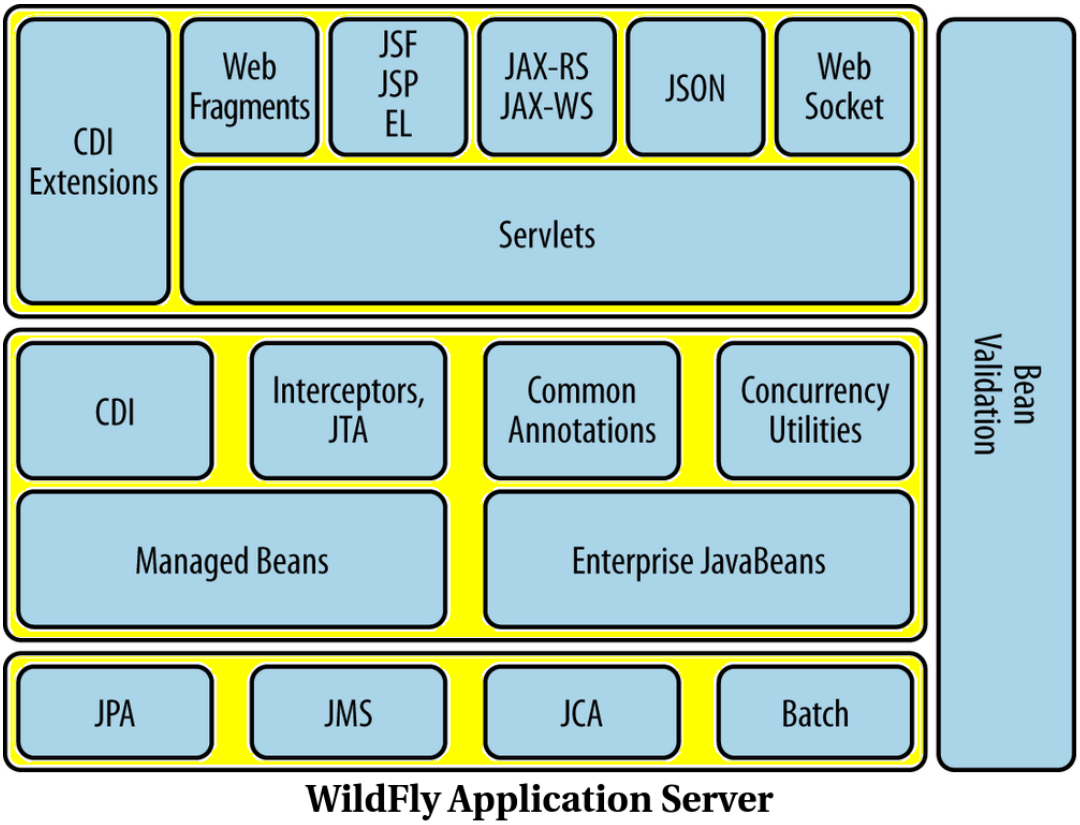
Java EE-palvelimet tarjoavat palvelut säiliöiden (Eng. Container) muodossa ja nämä palvelut vastaavat monitasoisen sovelluksen eri tasoja. Java EE-säiliöt (Eng. Java EE Container) ovat rajapinta käyt-

töliittymäkomponenttien ja alemman tason toimintojen välillä. Nämä toiminnot ovat tarkoitettu käyttöliittymäkomponenttien tukemiseksi. Alusta määrittelee jokaisen Java EE-säiliön toiminnallisuuden ja toiminnallisuus vaihtelee eri komponenteilla. (Oracle)

Jokaista eri tyyppistä komponenttia vastaa saman niminen säiliö. Web säiliöt (Eng. Web Container) ovat rajapinta verkkokomponenttien ja verkkopalvelimen (Eng. Web Server) välillä. Se lähettää pyyntöjä sovelluskomponenteille ja hallitsee komponenttien elinkaarta. EJB säiliö (Eng. EJB Container) on rajapinta Java EE-sovelluksen ja Java EE-palvelimen välillä, joka tarjoaa liiketoimintalogiikan yrityksen sovelluksille. EJB-säiliö toimii Java EE-palvelimella ja hallinnoi yrityksen sovellusten suorittamista. Asiakassovellussäiliöt (Eng. The Application Client Container) tarjoavat rajapinnan Java EE-asiakassovelluksen ja Java EE-palvelimen välillä. Ne toimivat asiakaskoneella ja ovat ikään kuin portti asiakassovelluksen ja Java EE-palvelinkomponenttien välillä. (Oracle)

4.4.1 WildFly

WildFly-palvelin on JBoss yhtiön luoma Java EE-sovelluspalvelin, joka myytiin vuonna 2006 Red Hatille. (Gupta, 2016) Aiemmin WildFly tunnettiin nimellä JBoss AS. Se perustuu ilmaiseen lähdekoodiin, joka on kirjoitettu Java-ohjelmointikielellä ja se on kehitetty toimimaan useilla eri alustoilla kuten Windows ja Linux. (Jethva) Voidaan ajatella, että WildFly on osa Java-sovellusta ja sitä käyttämällä saadaan sovelluspalvelimen tarjoamat palvelut sovelluksen käyttöön. Kuvassa 8 on esitetty palvelut, jotka ovat käytettävissä, kun Java-sovellus asennetaan WildFly-palvelimen päälle. (Mastertheboss, 2019)



Kuva 8 Palveluita, joita voidaan hyödyntää WildFlyn avulla (Mastertheboss, 2019)

Mediatri ja Kanta-adapteri tarvitsevat tuekseen sovelluspalvelimen, joista Mediconsult Oy on valinnut käyttöönsä WildFlyn. Sen tukemat palvelut mahdollistavat, että ohjelmoijan ei tarvitse kehittää palveluja uudelleen ja näin ollen kehityksessä voidaan keskittyä sovelluksen liiketoimintalogiikan kehittämiseen. Muun muassa WildFlyn tarjoamat Rest-rajapinta, webpalvelut ja viestijonot on koettu hyödyllisiksi ominaisuuksiksi Kanta-adapterin kehityksessä. (Kolmakow, 2019) WildFly toteuttaa myös uusimmat Java-standardit yritysmaailmassa sekä uusimmat web-kehitysstandardit. Nämä vähentävät teknistä taakkaa huomattavasti, mikä antaa paremmin aikaa suunnitella kehitettävää sovellusta. (WildFly)

4.5 Tietokantapalvelimet

Tietokantapalvelimet ovat tarkoitettu tiedon varastoimiseen, hakuun ja ylläpitoon. Ne sisältävät aina jonkun hallintajärjestelmän (DBMS) ja tietokannat. Hallintajärjestelmiä on useita, mutta tunnetuimpia ovat muun muassa MySQL, Oracle ja SQL Server. Niiden ensisijainen tehtävä on etsiä tietokannasta halutut tiedot ja palauttaa ne takaisin järjestelmään, josta kysely on peräisin. Kommunikointi tapahtuu SQL-kieltä hyödyntäen. (Thakur)

Mediconsult hyödyntää MySQL ja SolidDB-tietokantoja. Mediatri tallentaa tällä hetkellä kaikki tiedot SolidDB tietokantaan. Kanta-adapteri puolestaan tallettaa järjestelmään liittyvät asetukset ja lokitiedot MySQL-tietokantaan. Kanta-Adapterin myötä järjestelmän konfigurointi on lisääntynyt huomattavasti ja asetus- ja lokitietoja tallennetaan paljon. Kanta-adapterin kehitysvaiheessa MySQL oli luonteva valinta tietokannaksi, koska sen käyttö on maksutonta. Se perustuu avoimeen lähdekoodiin ja siitä on luettavissa paljon dokumentaatiota. Yritys on miettinyt myös Mediatrin tallennustietojen siirtämistä MySQL-tietokantaan, mutta toistaiseksi Mediatri tallentaa tietoa edelleen SolidDB-tietokantaan. (Kolmakow, 2019)

4.5.1 MySQL

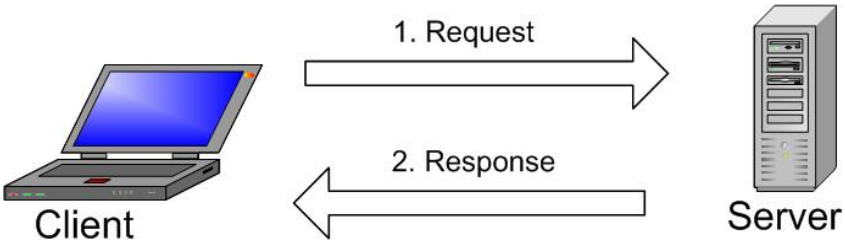
MySQL on alun perin kehitetty Ruotsissa vuonna 1994 MySQL Ab yrityksen toimesta. Se myytiin vuonna 2008 Sun Microsystemsille ja myöhemmin vuonna 2010 Oracle osti heidän yritystoimintansa. Tämän jälkeen MySQL on ollut Oraclen omistama. MySQL on avoimeen lähdekoodiin perustuvat relaatiotietokannan hallintajärjestelmä (RDBMS). Sitä hyödynnetään tietokantojen luomiseen ja halitaan relaatiomallia käyttäen. Relaatiomallissa tieto talletetaan ja järjestetään taulukoiksi ja jokainen taulukko liitetään toisiinsa määrätyllä tavalla. MySQL on yhteensopiva niin Linuxin, macOS kuin Windowsin kanssa. (Herawan, 2019)

Koska MySQL perustuu avoimeen lähdekoodiin, kaikki voivat käyttää ja muokata sitä haluamallaan tavalla. Kuka tahansa voi ladata ohjelman ja mukauttaa koodia omiin tarpeisiinsa sopivaksi. Ilmaisessa versiossa GNU Public Lisence (GPL) määrittää kuitenkin sen, miten käyttäjä voi muokata MySQLn toimintaa. Kaupallinen lisenssi antaa taas joustavuutta omien tarpeiden muokkaamiseen ja tarjoaa samalla parempaa tukea käyttäjälle. (Herawan, 2019)

MySQL hyödyntää asiakas-palvelinmallia (Eng. Client-Server Model). Asiakkaaksi kutsutaan tietokonetta, jolle tietokannan hallintajärjestelmä asennetaan. Kun asiakas esimerkiksi luo uuden tietokantataulun tai hakee tietoa tietokannasta, ottaa se yhteyttää palvelimeen, joka hallitsee tietokantaa. MySQL on vain yksi vaihtoehto tietokannan hallintajärjestelmistä. (Herawan, 2019) Muita RDBMS järjestelmiä ovat esimerkiksi SQL Server, IBM DB2 ja PostgreSQL (Rouse, 2018). Se on kuitenkin yksi suosituimmista ja sitä hyödyntävät muun muassa Google, Twitter ja Facebook (Herawan, 2019).

Asiakashallinta- ja palvelinhallintajärjestelmä keskustelevat keskenään hyödyntäen SQL-kieltä (Structured Query Language). SQL-kieli on kehitetty vuonna 1970 Ted Codd nimisen tietojenkäsittelytieteilijän toimesta. Vuonna 1974 SQL-kieli korvasi jo vanhentuneet kielet kuten ISAM ja VISAM. SQL-kieli kertoo palvelimelle, mitä käyttäjä haluaa tehdä tiedolle. Tietokysely on tietojen hakemista tietokannasta. Tietojen käsittely mahdollistaa taas tietojen lisäämisen, poistamisen, muuttamisen ja lajittelun. Näin voidaan esimerkiksi siis muokata sitä, missä järjestyksessä data näytetään tai jotain tietoa voidaan päivittää tietokantaan pysyvästi. Lisäksi tietokantaan voidaan määrittää datan tietotyyppi. Käyttäjä voi muuttaa esimerkiksi desimaaliluvut kokonaisluvuiksi. Lisäksi SQL-kielellä voidaan hallita kuka tietoja saa käyttää, mikä mahdollistaa tietojen suojaamisen. (Herawan, 2019)

MySQL:n toiminta on hyvin yksinkertainen. Asiakas muodostaa yhteyden palvelimeen verkon kautta ja kirjoittaa pyynnön graafisessa käyttöliittymässä SQL-kieltä hyödyntäen. Kysely lähetetään palvelimelle ja palvelin vastaa tehtyyn kyselyyn ja vastaus näytetään asiakassovelluksessa. MySQL voidaan käyttää usealla eri käyttöliittymällä, joista suosituimpia ovat muun muassa MySQL Workbench ja Admin Tool. Käyttöliittymiä on saatavana ilmaisena, mutta niitä on saatavilla myös maksullisina. (Herawan, 2019) Kuvassa yhdeksän on esitetty MySQL toiminta graafisesti.



Kuva 9 MySQL:n toiminta periaate (Herawan, 2019)

MySQL:n suosio perustuu avoimeen lähdekoodiin, joka tarjoaa joustavuutta ja suurin osa toiminnoista ovat ilmaisia. Tämän vuoksi ohjelmaa voi muokata omiin tarpeisiin myös ilmaiseksi GPL-sääntöjen puitteissa. Lisäksi se on helppo asentaa ja sen asennus on suhteellisen nopeaa. MySQL tarjoaa myös korkean suorituskyvyn, jonka mahdollistaa joukon klusteripalvelimia. Tämä mahdollistaa raskaidenkin kyselyjen tekemisen suhteellisen nopeasti ja näin MySQL-tietokantaan voidaan tallentaa paljon dataa. MySQL on myös turvallinen ja se sisältää Access Priviledge-järjestelmän ja käyttäjätilien hallinnan. (Herawan, 2019)

4.5.2 SolidDB

SolidDB on relaatiotietokanta, joka yhdistää muistipohjaisen- ja levypohjaisentietokannan edut. Muistipohjainentietokanta mahdollistaa korkean suorituskyvyn, kun taas levypohjainentietokanta tarjoaa lähes rajattoman kapasiteetin. Muistipohjaiset tietokannat hyödyntävät päämuistimoottoria (MME) ja levypohjaisettietokannat hyödyntävät taas levypohjaista moottoria (DBE). SolidDB käyttää molempia moottoreita samanaikaisesti. Ne ovat saman palvelinprosessin sisällä ja suoritettu SQL-lause voi hyödyntää molempia moottoreita. (Lindström;ym., 2013)

SolidDB on alkujaan neljän suomalaisen opiskelijan kehittämä. Sen kehitys on alkanut 1980-luvun loppupuolella ja tuote tuli markkinoille vuonna 1994. SolidDB:ta on pidetty helppokäyttöisenä ja luotettavana tuotteena, joka voidaan upottaa suoraan sovelluksiin. Sitä on käytetty muun muassa kauppojen kassapäätejärjestelmissä, kirjanpitojärjestelmissä, hammasröntgenlaitteissa ja pilvenpiirtäjähisseissä. (Leino, 2000)

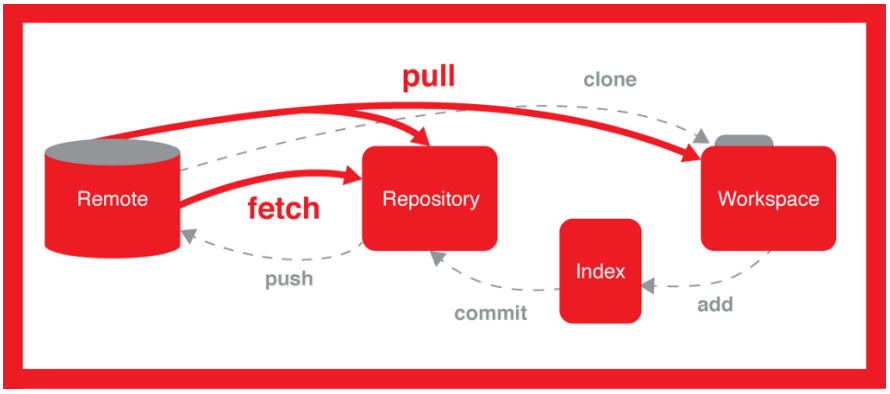
4.6 Eclipse

Eclipse on ilmainen kehitysalusta, joka pohjautuu Javaan. Se tarjoaa laajan valikoiman erilaisia laajennuksia, jotka mahdollistavat kehityksen ja testauksen myös muiden ohjelmointikielien osalta. Kehitysalusta sai alkunsa vuonna 2001 IBM:n toimesta. He lahjoittivat silloin kolme miljoonaa riviä koodia, jotta voitaisiin kehittää avoimeen lähdekoodiin perustuvaa kehitysympäristöä. Eclipsen kehityksessä arvostetaan laatua, sovellusohjelmointirajapinnan vakautta ja yhdenmukaisia julkaisuaikatauluja. (Rouse, 2017)

Eclipse on suunniteltu integroitujen verkko- ja sovelluskehitystyökalujen rakentamiseen erilaisia laajennuksia (Eng. Plug-in) hyödyntäen. Se on suunniteltu toimimaan usealle eri käyttöliittymälle ja jokainen laajennus toimii samalla tavalla kaikissa tuetuissa käyttöjärjestelmissä. Laajennukset voivat suorittaa monenlaisia tehtäviä kuten määrittelyn, testauksen, animoinnin, koodin julkaisemisen tai kääntämisen ja virheenkorjauksen. Avoimenlähdekoodin vuoksi jokainen laajennuksia kehittävä taho voi keskittyä omaan osaamisalueensa ja lisätä laajennukset Eclipsen alustaan kehittäjien käytettäväksi. (eclipse)

Mediconsultilla Eclipse on todettu hyväksi työkaluksi backend-koodin kehittämiseen juuri monipuolisen laajennusvalikoiman ja jatkuvasti parantuvan kehitysalustan vuoksi. Laajennuksia, joita Eclipse tukee ja yritys käyttää kehitystyössään, ovat muun muassa Git ja Hibernate, joka on esitelty jo aiemmin opinnäytetyössä. (Kolmakow, 2019) Näiden lisäksi esimerkiksi Eclipsen tukema JBoss Tools mahdollistaa sovelluspalvelimien suoran hallinnan ja se tarjoaa valmiit JavaClientit, joita voidaan hyödyntää kehityksessä. Eclipse tarjoaa myös hyvät debug ominaisuudet. Eclipsen debug ominaisuudet mahdollistavat muuttujien selkeän tarkastelun ja koodin suorittamisen rivi kerrallaan halutusta kohdasta. (Kolmakow, 2019)

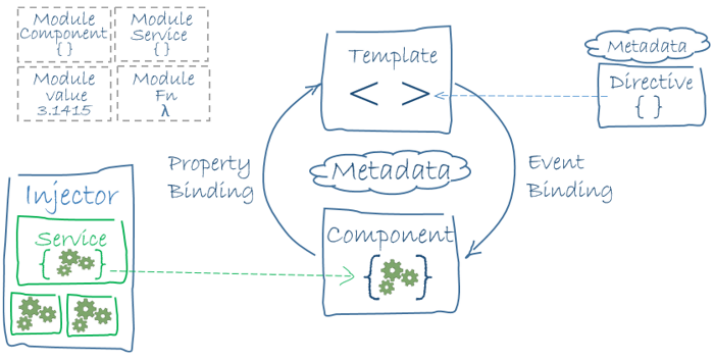
Git on avoimeen lähdekoodiin perustuvat versionhallintajärjestelmä, jolla voidaan hallita niin pieniä kuin suuria projekteja nopeasti ja tehokkaasti. Se mahdollistaa usean kehityslinjan luomisen ja niiden yhdistämisen. Lisäksi Git mahdollistaa hajautetun työskentelyn ja ehkäisee virheellisen tiedon syntymistä sekä tiedon katoamista. (Git) Gitin toiminta perustuu muutamiin peruskomentoihin, jotka mahdollistavat hyvän versionhallinnan (Kuva 10). Tieto Gitiin tallentuu 'kommitteina' (Eng. Commit), jotka sisältävät koodiin tehdyt muutokset. Tiedot voidaan tallentaa omaan haaraan (Eng. Repository) tai vaihtoehtoisesti ne voidaan lisätä päähaaraan (emo-repositorioon). Kun tiedot on tallennettu johonkin haaraan, ne voidaan julkaista puskemalla (Git Push). Julkaistut muutokset saadaan käyttöön hakemalla ne emo-haarasta (Git Pull). Git Fetch-komenolla voidaan hakea projektin uusin tila, mutta se ei tee muutoksia paikalliseen haaraan. Git Clone-komento mahdollistaa taas koko projektin kopioinnin omalle koneelle. (Lofhjelm)



Kuva 10 Gitin toiminta periaate (Paul, 2018)

4.7 Angular

Tällä hetkellä yritys pyrkii uudistamaan potilastietojärjestelmäänsä nykyaikaisemmaksi web-sovellukseksi. Uudistuksen myötä yritys on alkanut hyödyntämään sovelluskehityksessä frontend-puolella Angularia. Angular on kehitysalusta ja -kehys HTML- ja TypeScript-sovelluksille ja se on kirjoitettu TypeScriptia hyödyntäen. Kehittäjä tuo sovellukseen haluamansa TypeScript-kirjastot, joista sovellus suorittaa ydintoiminnot sekä myös vaihtoehtoiset toiminnot. Angular rakentuu useista eri elementeistä (Kuva 11) ja niistä tärkeimmät ovat NgModule ja komponentit. Lisäksi se hyödyntää palveluita (Eng. Services) ja reititystä, joka määrittää järjestyksen komponenteille. (Angular)



Kuva 11 Angularin elementit (Angular)

Angular-sovellukset rakentuvat NgModuuleista, jotka toimivat ikään kuin säiliöinä yhtenäisille koodilohkoille. Ne voivat sisältää komponentteja, palveluntarjoajia ja muita kooditiedostoja. Moduulit voivat tuoda toimintoja muista NgModuuleista ja lisäksi viedä toimintojaan muiden NgModuulien käytettäväksi. Jokaisessa Angular-sovelluksessa on vähintään yksi NgModule-luokka, joka nimetään tavallisesti AppModuleksi. Tätä kutsutaan usein juurimoduuliksi, joka käynnistää Angular-sovelluksen. Usein miten NgModuuleja on kuitenkin useampia ja ne muodostavat keskenään oman sovellushierarkiansa. (Angular)

Komponentit ohjaavat käytännössä päätelaitteita. Ne määrittävät mitä käyttäjälle näytetään ja niihin voidaan määrittää oma sovelluslogiikkansa TypeScriptia hyödyntäen. Käyttäjän käyttäessä Angular-sovellusta, sovellus luo, päivittää ja tuhoaa komponentteja käyttäjän käytön mukaan. Metatiedot määrittävät rakennuspalikat jokaisen näkymän luomiseen ja näyttämiseen jokaiselle komponenteille. Komponenttien näkymä voidaan myös määrittää mallin (Eng. Template) avulla. Malli on HTML-muoto, jossa voidaan hyödyntää esimerkiksi ngFor-rakennetta listojen näyttämiseen tai ngIf-rakennetta ehtojen rakentamiseen HTML-koodissa. (Angular)

Palvelu-luokat on tarkoitettu toiminnoille, jotka halutaan jakaa useammalle komponentille. Ne on erotettu komponenteista omaksi luokakseen ja ne lisätään komponentin käyttöön erikseen riippuvuutena. Tämä tekee komponentista kevyemmän ja tehokkaamman, kun niiden ei tarvitse hakea tietoa palvelimilta tai tarkistaa käyttäjän antamien arvojen oikeellisuutta. Palveluiden uudelleen käytettävyyden vuoksi kehittäjän ei tarvitse kirjoittaa samaa koodia uudelleen jokaiselle komponentille erikseen. (Angular)

Angularin reititys (Eng. Router) mahdollistaa navigoinnin näkymästä toiseen käyttäjän liikkuessa sovelluksessa. Angularin reititys pystyy tulkitsemaan siirtymiä URL-osoitteen perusteella kuten verkkoselain. Lisäksi se pystyy siirtymään uuteen näkymään napin painalluksella tai muusta lähteestä tulevalla ärsykkeellä. Näkymän vaihdon yhteydessä reititin voi siirtää mukanaan tarvittavat parametrejä, jotka voivat esimerkiksi määrittää sisällön, joka käyttäjälle halutaan näyttää. Lisäksi se kirjaa tiedot selaimen historiapäiväkirjaan, mikä mahdollistaa myös taakse- ja eteenpäin-painikkeiden käytön sovelluksessa. (Angular)

Opinnäytetyön käyttöliittymässä hyödynnetään Angular Material komponentteja. Se luo valmiiksi tyylitellyt käyttöliittymäkomponentit, jotka toimivat verkkoselaimella ja mobiililaitteilla. Angular Materialin ominaisuudet toimivat hyvin nykyaikaisissa selaimissa. Komponentit ovat mahdollista teemoittaa värien ja muotoiluiden avulla, jokaiselle sovellukselle tarpeen mukaan. (Angular)

4.8 Visual Studio Code

Käyttöliittymän ohjelmointia varten yritys hyödyntää Visual Studio Codea, joka on parempi alusta kehittää Angularia Eclipseen verrattuna. (Kolmakow, 2019) Visual Studio Code on kevyt ja tehokas koodieditori, joka toimii Windows-, macOS- ja Linux-käyttöjärjestelmissä. Se käyttää avoimen lähde-

koodin menetelmiä. Visual Studio Code mahdollistaa ohjelmoinnin useilla eri ohjelmointikielillä laajennusten avulla, joista suosittuja ovat muun muassa Python, C/C++, ESLint Angular ja C#. Siihen on myös sisäänrakennettu tuki JavaScriptille ja NodeJS:lle. (Microsoft)

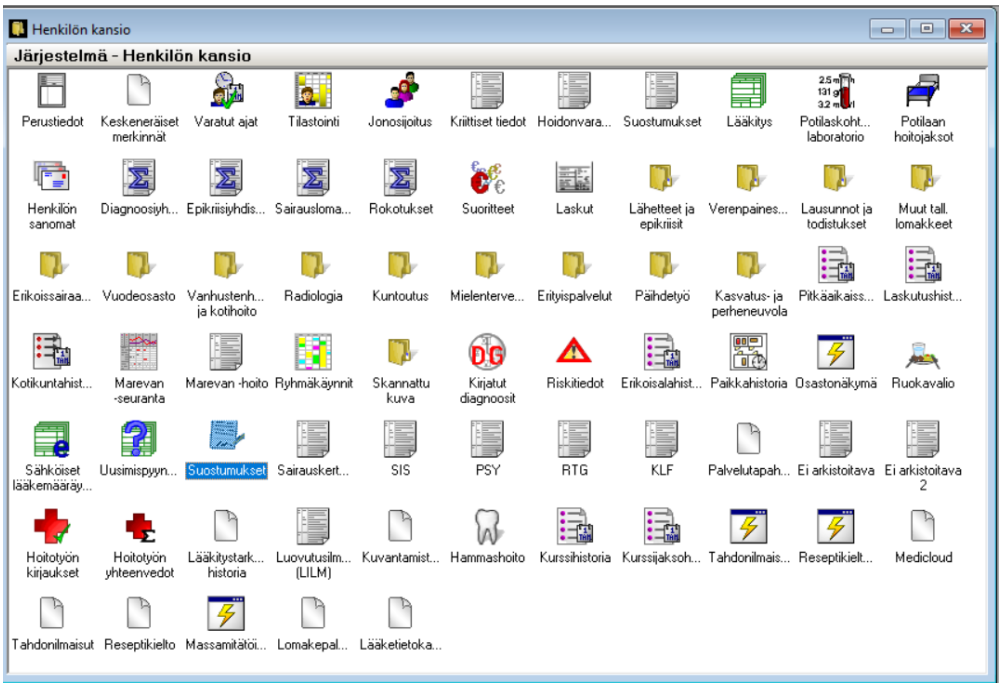
Visual Studio Code tarjoaa IntelliSense ominaisuuden, joka mahdollistaa älykkään syntaksin täydentämisen perustuen muuttujatyyppeihin, funktioiden määrityksiin ja tuotuihin moduuleihin. Ohjelma mahdollistaa testauksen suoraan editorissa. Testauksessa voidaan hyödyntää esimerkiksi pysäytyspisteitä (Eng. Breakpoint) ja interaktiivista konsolia, joka kertoo esimerkiksi koodissa olevista virheistä. Visual Studio Codeen on rakennettu sisään Git-komennot, joten työskentely Git-palvelun tai muiden SCM-palvelujen kanssa on helppoa ja vaivatonta. (Microsoft)

5 TOTEUTUS

Opinnäytetyön tavoitteena oli toteuttaa websovellus Mediatrin pääkäyttäjille, joka listaa Kanta-adapterin tarvitsemia Mediatri asetuksia ja tarkastaa tietojen oikeellisuuden. Listaamalla asetukset sovellukseen ja validoimalla ne, näkee pääkäyttäjä suoraan virheelliset tai puuttuvat asetukset. Tämä nopeuttaa ongelmien ratkaisua esimerkiksi tilanteissa, joissa organisaatio ei pysty arkistoimaan haluttua asiakirjaa Kanta-palveluihin virheellisten asetusten vuoksi.

Opinnäytetyö tekninen osuus perustuu Mediatrin asetusten tarkasteluun. Mediatrin asetukset ovat konfiguroitavissa olevia asetuksia, jotka voidaan määrittää jokaisessa järjestelmässä eri nimillä. Esimerkiksi nimitieto voi esiintyä nimillä 'Nimi', 'Etunimi' tai 'Kutsumanimi'. Tämä aiheuttaa omat haasteensa Kanta-yhteyden ongelmien selvittämisessä. Asetukset sisältävät erilaisia tietoja, kuten niin sanottuja perustieto-otsikoita ja yksittäisiä arvoja. Yksittäisiä arvoja sisältäviä asetuksia ovat esimerkiksi rekisterikonfiguraatioon liittyvät rekisteritiedot. Rekisteritietoja ovat esimerkiksi palveluntarjoajan OID-tunnus tai yhteystiedot. Otsikko-asetukset taas määrittelevät otsikoita perustietolomakkeelle. Otsikko-asetusten perusteella Kanta-adapteri hakee tietoa Mediatrin lomakkeista, joten otsikon puuttuminen voi aiheuttaa puutteellisien tietojen siirtymisen Kanta-arkistoon tai tiedon puuttuminen voi estää arkistoinnin.

Kanta-adapteri pitää sisällään paljon asetuksia, mutta tässä vaiheessa niistä valittiin olennaisimmat Kanta-liitännän kannalta. Asetukset sijaitsevat eri lomakkeilla ja lomakkeet sijaitsevat eri kansioissa (Kuva 12). Näin ollen asetukset hajoavat järjestelmässä useaan eri paikkaan. Tämän vuoksi asetuksia on hankala löytää ja esimerkiksi epäonnistuneen arkistoinnin syy voi olla hankala selvittää. Opinnäytetyössä tarkasteltiin 'Lääkkeet'-lomakkeen, 'Suostumukset'-lomakkeen ja rekisterikonfiguraatioon liittyviä asetuksia. Lisäksi tarkasteltiin Mediatriin ladattuja koodistoja ja tarkastettiin, onko Kanta-liitäntää varten ladattu olennaiset koodistot.



Kuva 12 Esimerkki Mediatri järjestelmästä 'Henkilö kansion' sisältämistä lomakkeista ja asetuksista

5.1 Suunnittelu

Opinnäytetyön suunnittelu ja aiheeseen perehtyminen alkoi yrityksen tietokantarakenteeseen tutustumalla. Kuten aiemmin tuli ilmi, Mediatrin tietokantarakenne jakautuu SolidDB- sekä MySQL-tietokantaan. Mediatrin tallentama tieto tallentuu SolidDB-tietokantaan ja Kanta-Adapteri tallettaa tietoa puolestaan MySQL-tietokantaan. Kaiken kaikkiaan tietokantarakenne koostuu yli neljästäsadasta tietokanta taulusta, joiden suhteet ovat tarkkaan mietitty kokonaisuus. Tämän vuoksi tietokantarakenteen omaksuminen ja oleellisen tiedon löytäminen opinnäytetyön kannalta vaati aikaa. Aiheen omaksumista helpotti Mediconsult Oy:lta saama tuki ja kesäharjoittelujakso, jonka aikana rakenne oli ehtinyt tulla osittain tutuksi.

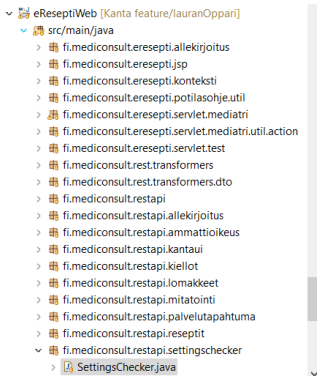
Tietokantarakenteeseen perehtymisen jälkeen, alkoi tutustuminen sovelluksen tarkempiin vaatimuksiin, jotka Mediconsult Oy oli määrittänyt. Vaatimukset oli pilkottu pieniin kokonaisuuksiin Jiraan, joka helpotti työn aloittamista ja aiheen kokonaiskuvan hahmottamista. Jiraan pilkotuista tehtävistä (Eng. Jira Task) valittiin aluksi kehitettäväksi tärkeimmät ominaisuudet Kanta-adapterin parametrisointityökalun kannalta.

Ennen varsinaista ohjelmointia täytyi sovellukselle luoda oma Angular-projekti (Kuva 13).

```
C:\Users\Laura.kemppainen\oppari\Kanta\eReseptiWeb>ng new settings-checker
```

Kuva 13 Uuden Angular-projektin luominen

Sen lisäksi sovellukselle luotiin oma Rest-rajapinta, jonka kautta välitetään kutsut käyttöliittymästä palvelimelle (Kuva 14). Sovelluksen Java- ja Angular-luokat sijoitettiin yrityksen projektirakenteen sisään.

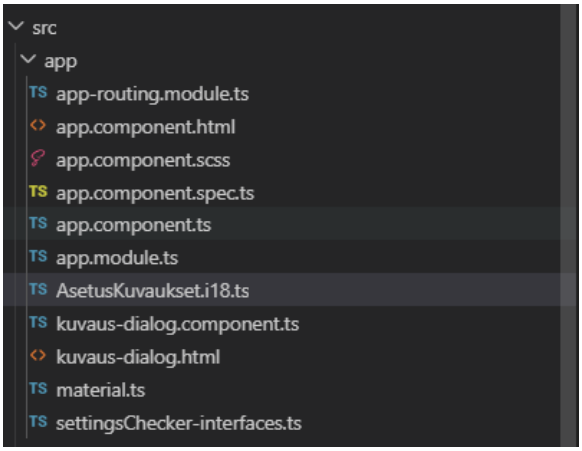


Kuva 14 Kanta-adapterin parametrisointityökalun Rest-rajapinta

Projektin luomisen jälkeen, projekti kansiossa syötettiin vielä seuraava komento:
$ng serve --open

Komento rakentaa sovelluksen, käynnistää kehityspalvelimen, tarkkailee lähdetiedostoja ja uudelleen rakentaa sovelluksen aina, kun tehdään muutoksia tiedostoihin. Lisäksi komento avaa sovelluksen

osoitteessa http://localhost:4200/. Luomisen yhteydessä komento muodostaa automattisesti Angular-sovelluksen perusrakenteen. Opinnäytetyössä muokattiin perusrakennetta lisäämällä 'AsetusKuvaukset.i18.ts'-tiedosto asetusten kuvausten ja sovelluksen ohjeiden generoimista varten sekä esimerkiksi mahdollisen kieliversioinnin helpottamiseksi. Lisäksi yhteistä lokalisointi-tiedostoa hyödyntämällä, voidaan minimoida muutosten määrä yhteen paikkaan, jos esimerkiksi asetuksen kuvaus muuttuu. Lokaalisointitiedoston lisäksi, projektille on lisätty kuvausdialogia varten omat tiedostot sekä liitäntätiedosto (Eng. Interface), jonne on sijoitettu asetusten tyypitystä varten luokat sekä luettelot (Eng. Enumerate Type) (Kuva 15).



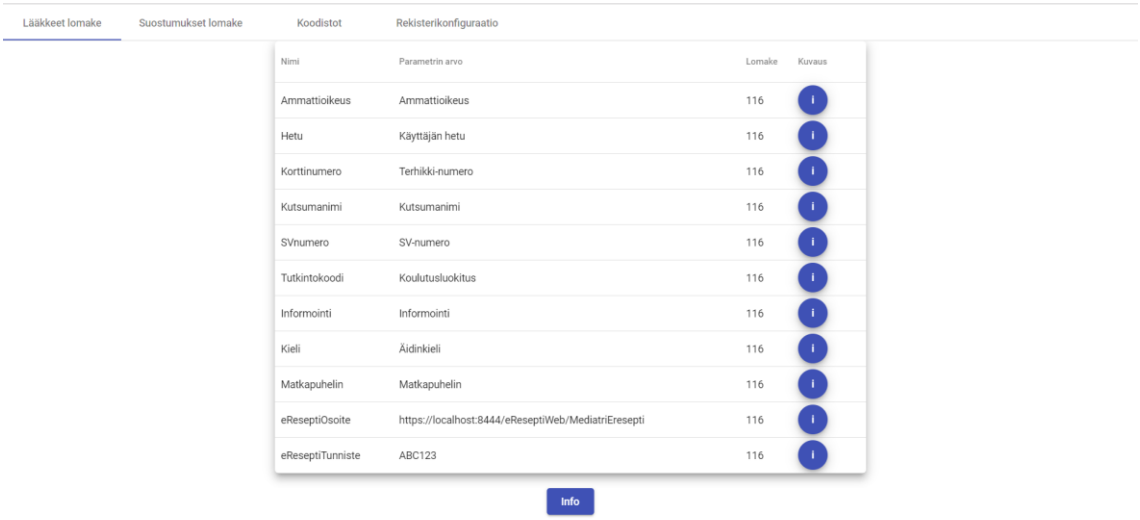
Kuva 15 Työkaluun luotu Angular-projektin rakenne

Käyttöliittymää suunnitellessa ja sopiva komponentteja etsiessä törmäsi lähes poikkeuksetta Angular Material-paketin tarjoamiin komponentteihin ja ratkaisuihin. Komponentit ja ratkaisut vaikuttivat yksinkertaisilta ja helppokäyttöisiltä, minkä vuoksi sovellukseen ladattiin Angular Material-paketti käyttöliittymä suunnittelua varten (Kuva 16).

```
C:\Users\Laura.kemppainen\oppari\Kanta\eReseptiWeb\settings-checker>ng add @angular/material
```

Kuva 16 Angular Material-paketin lataaminen

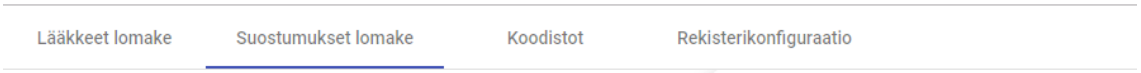
Angular Material-paketin lataaminen nopeutti käyttöliittymän suunnittelua ja mahdollisti valmiiden teemojen ja komponenttien hyödyntämisen lopullisessa ulkoasussa. Komponentteja löytyi käyttöön valmiina runsaasti ja pienillä tyylimuutoksilla sai aikaan selkeän käyttöliittymän, johon oli helppo jaotella esimerkiksi erilasia asetuksia lomakkeen tai kokonaisuuden mukaan. (Kuva 17).



Kuva 17 Kuvankaappaus toteutetusta käyttöliittymästä Angular Material-pakettia hyödyntämällä

5.2 Sovelluksen toteutus

Opinnäytetyössä kehitetty sovellus tehtiin Rest-pohjaisena websovelluksena. Sovellus rakentuu Angularilla rakennetusta käyttöliittymästä, Rest-rajapinnasta, Facade-luokasta ja validoinnista vastaavasta Java-luokasta. Facade-luokassa on tarkoitus suorittaa vain yksinkertaisia toimintoja, joka mahdollistaa selkeämmän käyttöliittymän toteutuksen. Facade-luokan on tarkoitus ikään kuin piilottaa monimutkaiset toiminnot alleen ja erottaa asiakasohjelman toteutus monimutkaisemmasta palvelin toteutuksesta. Käyttöliittymässä eri tyyppisiä asetuksia jaoteltiin välilehdillä, jonne yhdelle välilehdelle kerättiin esimerkiksi yhden lomakkeen tai yhden kokonaisuuden asetukset (Kuva 18).



Kuva 18 Käyttöliittymään toteutetut välilehdet

Sovelluksen toimintaperiaate hyvin yksinkertainen. Käyttöliittymästä lähetetään http GET-kutsuja palvelimelle Rest-rajapintaan. Rajapinta välittää kutsut Facade-luokkaan, jossa suoritetaan toiminnot, kuten asetusten hakeminen tietokannasta Hibernatea hyödyntäen ja haettujen asetusten lisääminen käyttöliittymälle menevään listaan. Facade-luokassa lisätään myös esiehtoja asetuksille, jotka niiden on täytettävä. Jos esiehdot eivät täyty, ei asetus ole validi. Kaikkien asetusten validointi ja esiehtojen tarkastus tapahtuu omassa luokassaan, jonne on rakennettu omat toiminnot asetusten tarkistamiseksi.

Selkeä esimerkki ketjusta on esimerkiksi postinumero tiedon hakeminen, validointi ja palauttaminen käyttöliittymään. Käyttöliittymästä lähetetään Rest-kutsu, jolla halutaan hakea 'Suostumukset'-lomakkeen asetukset. Rest-rajapinta lähettää kutsun Facade-luokan metodille, joka suorittaa tietokantahaun Hibernatea hyödyntäen. Facade-luokassa lisätään tiedot käyttöliittymälle menevään listaan. Lisäksi Facade-luokasta kutsutaan postinumeron validointiin tarkoitettua metodia luokasta, johon on lisätty kaikki validointiin tarkoitetut toiminnot. Postinumeron tapauksessa metodi tarkistaa onko postinumeron pituus viisi merkkiä (Kuva 19). Jos ehto täyttyy, viedään käyttöliittymälle tieto asetuksen

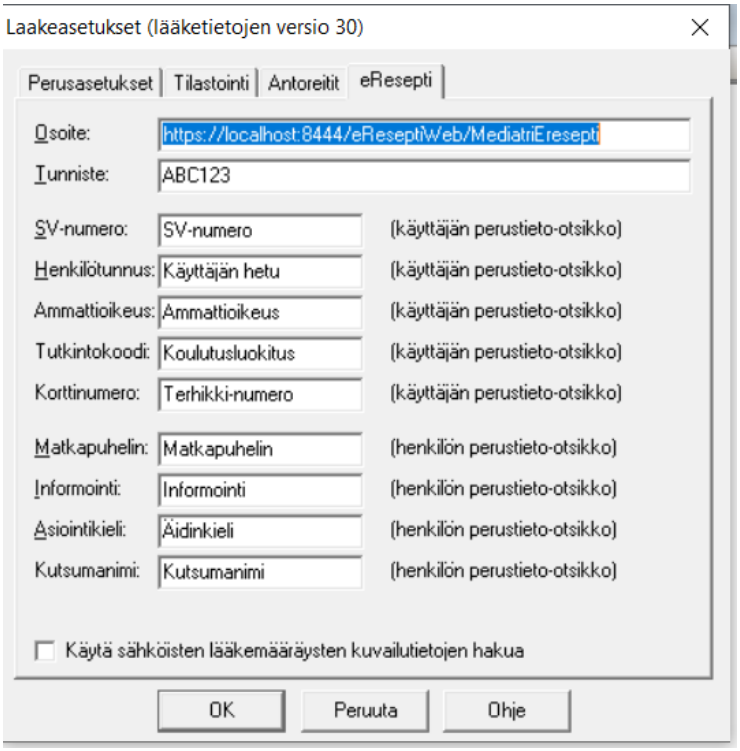
oikeellisuudesta. Mikäli ehto ei täyty, viedään myös tästä tieto käyttöliittymälle ja korostetaan käyttöliittymässä virheellistä tietoa punaisella värillä.

```
final Pattern pattern = Pattern.compile("^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$");
final Matcher m = pattern.matcher(numero);
final boolean isValid = m.matches();
return isValid;
```

Kuva 19 Esimerkki puhelinnumeron validoinnista säännöllisen lausekkeen avulla, joka tarkistaa, että puhelinnumero on 10 merkkiä pitkä

5.2.1 'Lääkkeet lomake'-välilehti

Ensimmäisessä vaiheessa toteutettiin 'Lääkkeet'-lomakkeeseen liittyvien asetusten haku, jotka koskevat sähköisiä reseptejä (Kuva 20).



Kuva 20 Sähköisiä lääkemääräyksiä koskevat asetukset 'Lääkkeet'-lomakkeelta

Asetukset ja niiden oikeellisuus ovat olennaisia, jotta sähköisen lääkemääräykset voidaan lähettää Kanta-palveluihin. Asetusten haku suoritettiin Hibernatea hyödyntäen. Ennen varsinaisten asetusten hakua täytyi hakea lomakenumero lomaketyypin perusteella (Kuva 21). Kun lomakkeen numero oli tiedossa, voitiin hakea lomaketta koskevat asetukset tietokannasta HQL-kyselyllä.

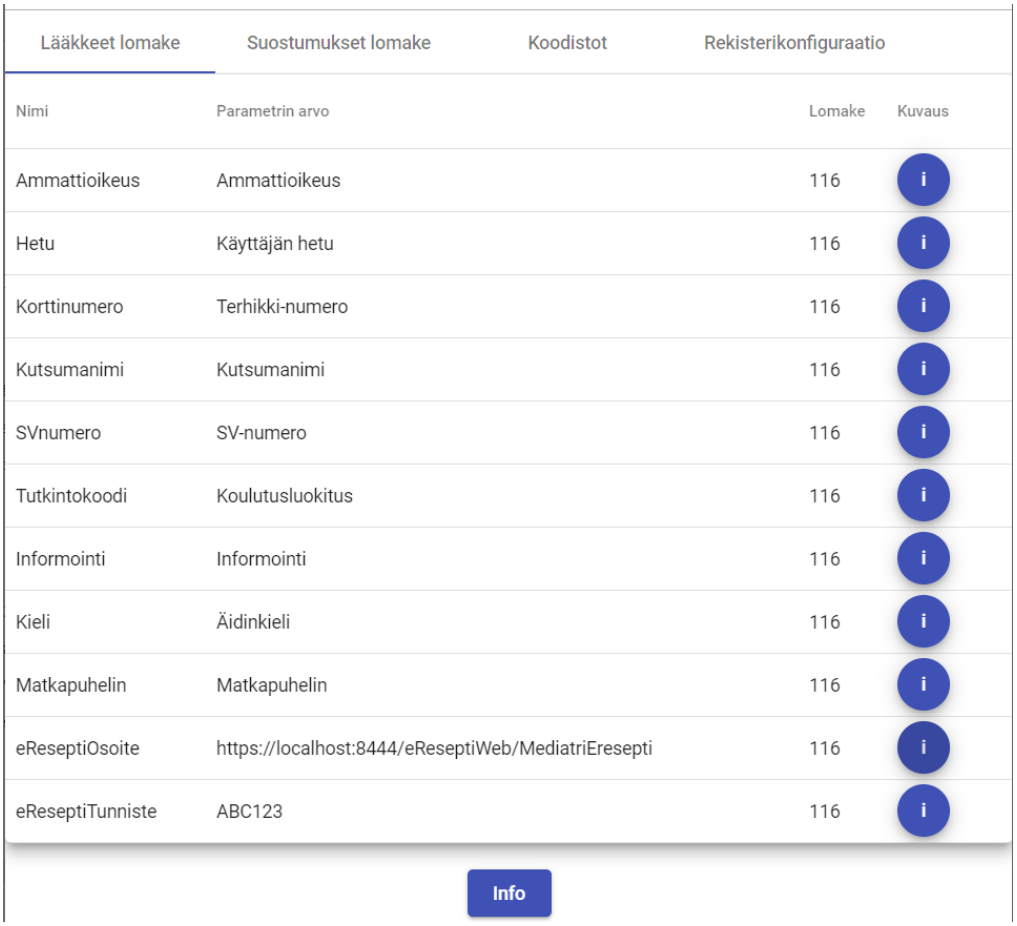
```
private static final String SQL_LOMAKETYYPPIKYSELY =

final List<? extends LomakeEntity> kyselynTulos =
    entityManager.createQuery(SQL_LOMAKETYYPPIKYSELY).setParameter(PARAM_TYYPPI, lomaketyyppi).getResultList();
```

Kuva 21 Lomakenumeron selvittäminen Hibernatea hyödyntäen (Kuva salattu osittain salassapitosopimuksen vuoksi)

’Lääkkeet’-lomakkeen asetusten validointi oli helppoa, koska kaikki asetukset ovat pakollisia, jotta sähköisten lääkemääräysten lähettäminen Kanta-palveluihin on mahdollista. Osoitteen osalta oli olennaista tarkistaa, että osoite noudattaa URL-osoitteen vaatimuksia. URL-tarkastus tehtiin säännöllisiä lausekkeita (Eng. Regular Expression) hyödyntämällä. Niillä voidaan käsitellä tehokkaasti tekstimuotoisia merkkijonoja ja tarkastaa, että merkkijono on halutun muotoinen. URL-osoitteen kannalta voidaan tarkastaa, että se alkaa halutulla tavalla (esimerkiksi http:// tai https:// tai www.) ja että sen kauttaviivat (/) ovat määritetty URL-osoitteen kannalta oikein päin.

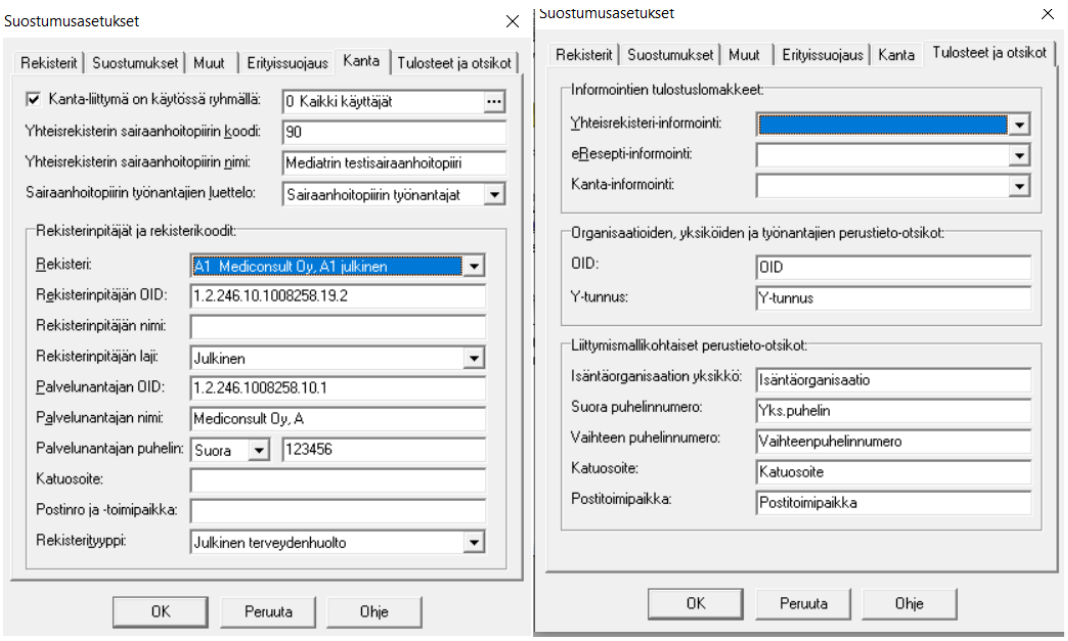
Käyttöliittymässä ’Lääkkeet’-lomakkeen asetukset sijoitettiin Angular Material taulukkoon ’Lääkkeet lomake’-välilehdelle (Kuva 22). Tiedoista muokattiin luettava taulukko, jossa näytetään parametrin nimi ja arvo sekä lomakenumero, josta asetus löytyy. Kuvaussarakkeeseen luotiin oma painike, jonka taakse generoitiin jokaiselle asetukselle oma kuvauksensa. Kuvauksen tarkoitus on kertoa käyttäjälle mihin kyseistä asetusta hyödynnetään. Jos taulukon arvot ovat korostettu punaisella värillä, on tieto jollain tapaa virheellinen. Tällaisissa tapauksissa tieto voi puuttua kokonaan tai esimerkiksi URL-osoite ei ole oikean muotoinen.



Kuva 22 'Lääkkeet'-lomakkeen asetukset käyttöliittymässä

5.2.2 'Suostumukset lomake'-välilehti

'Suostumukset'-lomakkeeseen liittyy paljon asetuksia (Kuva 23), joista täytyi tarkistaa niiden pituutta (esimerkiksi postinumero), oikeaa muotoa (esimerkiksi puhelinnumero ja OID-tunnus), löytyykö niiden arvo otsikkorekisteristä ja löytyykö niiden arvo tietystä koodistosta. Lisäksi niille täytyi määrittää ehdollisia pakollisuuksia varten erilasia ehtoja, joiden on täytyttävä, jotta tieto on validi.



Kuva 23 Suostumukset-lomakkeen asetuksia

Kuten 'Lääkkeet'-lomakkeen tapauksessa, myös 'Suostumukset'-lomakkeelle täytyi hakea lomakenumero lomaketyypin perusteella. Lomakenumeron avulla voitiin hakea sen jälkeen 'Suostumukset'-lomakkeen asetukset. Kun asetukset oli haettu, käytiin ne yksitellen läpi, validointiin ja lisättiin mahdolliset ehdot pakollisuuksille (Kuva 24).

```
final IMediatriPersistenceContext mpc = new MediatriPersistenceContext("ABC123");
final List<ILomakeGetters> findAllByOmistajaAndTyyppi = lomakeFinder.findJarjestelmaLomake((short) 141, mpc);
final List<Integer> lomakkeet = findAllByOmistajaAndTyyppi.stream().map(x -> x.getLomake()).collect(Collectors.toList());
final List<IAsetusGetters> asetukset = asetusFinder.findAll(0, lomakkeet, mpc);
switch (asetus) {
case ("Isäntä-otsikko"):
  lomakeAsetuksetDTO.setArvo(filteroidytAsetukset.get(i).getTeksti());

  // Listataan tarvittavat esiehdot: Isäntä-otsikko pakollinen on rekisterinpitäjän laji on yksityinen
  final YhdistettyEhto yhdistettyEhtoIsanta = new YhdistettyEhto();
  yhdistettyEhtoIsanta.getYhdistettavat().add(new Esiehto("eArkisto käytössä", "Kyllä"));
  yhdistettyEhtoIsanta.getYhdistettavat().add(new Esiehto("Rekisterinpitäjän laji", "Yksityinen"));
  lomakeAsetuksetDTO.getMahdollinenEsiehto().add(yhdistettyEhtoIsanta);

  // Tarkistetaan onko otsikko otsikkorekisterissä
  isOtsikko = getSettingsValidator().checkIsOtsikko(filteroidytAsetukset.get(i).getTeksti(), mpc);
  lomakeAsetuksetDTO.setOtsikko(isOtsikko);
  if (isOtsikko == false) {
    lomakeAsetuksetDTO.setWarning(true);
  }
  break;
```

Kuva 24 Esimerkki koodista, jossa on toteutettu asetuksien haku, ehdollisten pakollisuuksien lisääminen ja tietojen validointi

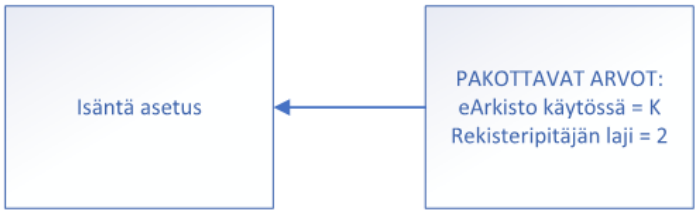
Arvojen pituuden ja muodon tarkastuksen lisäksi tarkasteltiin tietokantaa. Tietokannasta tarkistettiin esimerkiksi, että asetuksen annettu arvo löytyy otsikkorekisteristä tai että annettu käyttäjäryhmä

löytyy tietokannasta (Kuva 25). Otsikkorekisteri sisältää kaikki Mediatrin käyttämät otsikot ja niiden parametrisoinnit. Kanta-adapteri hyödyntää tiettyjen tietojen etsimiseen juuri otsikkoasetuksia, minkä takia arvon tarkistaminen on olennainen osa validointia.

```
public boolean validateKayttajaryhma(String kayttajaryhmaNro, IMediatriPersistenceContext mpc) {
  if (kayttajaryhmaNro == null) {
    return false;
  }
  final IKayttajaryhmaGetters kayttajaryhma = kayttajaryhmaFinder.find(kayttajaryhmaNro, mpc);
  if (kayttajaryhma == null) {
    return false;
  }
  return true;
}
public IOtsikkoGetters haeOtsikko(String haettavaOtsikko, IMediatriPersistenceContext mpc) {
  final IOtsikkoGetters otsikot = getOtsikkoFinder().find(haettavaOtsikko, mpc);
  if (otsikot != null) {
    return otsikot;
  }
  return null;
}
```

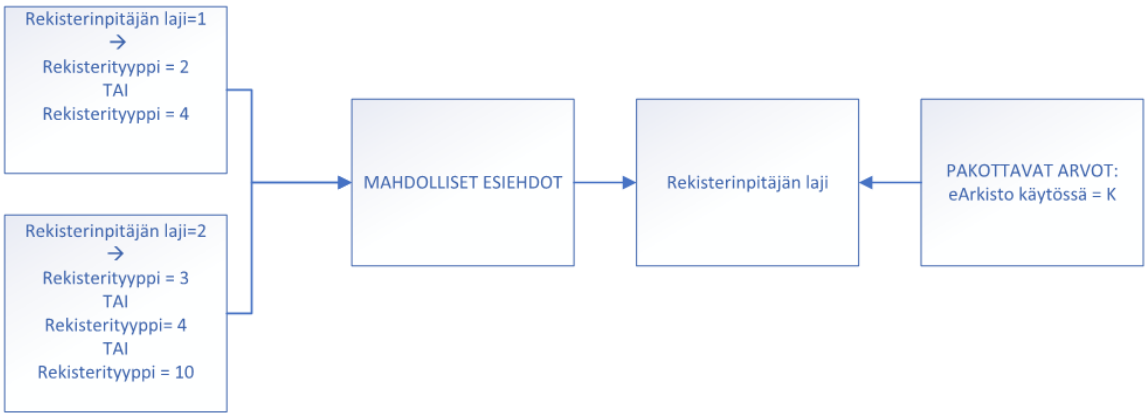
Kuva 25 Käyttäjäryhmän ja otsikon arvon tarkastaminen tietokannasta

Ehdollisilla pakollisuuksilla tarkoitetaan sellaisia ehtoja, jotka pakottavat tietyn arvon pakolliseksi. Helppo esimerkki pakottavista arvoista on Isäntä-otsikon pakollisuus (Kuva 26). Jos 'eArkisto käytössä'-asetuksen arvoksi on asetettu 'K' ja jonkun 'Rekisterinpitäjän laji'-asetuksen arvo on kaksi, on myös 'Isäntä-otsikko'-asetus pakollinen. Jos edellisistä ehdoista jompikumpi ei täyty, ei 'Isäntä-otsikko'-asetus ole enää pakollinen.



Kuva 26 'Isäntä-otsikko'-asetuksen pakollisuus

Sen lisäksi, että joku arvo tekee asetuksesta pakollisen, voi asetuksen arvo vaikuttaa toisen asetuksen oikeisiin arvoihin (Kuva 27). Esimerkiksi 'Rekisterinpitäjän laji'-asetuksen kohdalla tietty arvo pakottaa 'Rekisterityyppi'-asetuksen arvoksi tietyn arvon. Jos 'Rekisterinpitäjän laji'-asetus saa arvokseen arvon yksi ja kyseisen 'Rekisterityyppi'-asetuksen arvo olisi kolme, tekisi tämä yhdistelmä 'Rekisterityyppi'-asetuksesta epävalidin.



Kuva 27 'Rekisterinpitäjän laji'-asetuksen esiehdot

'Suostumukset'-lomakkeen pakollisuuden ehtojen yhteydessä hyödynnettiin JUnit-testejä ehtojen testaamiseen. JUnit-testeissä oli nopea testata asetettujen ehtojen toimivuus. Yksinkertaisuudessaan testiin tehtiin oma kovakoodattu lista tarkasteltavista asetuksista ja annettiin niille arvoja sen mukaan, mitä ehtoa oltiin testaamassa. Kun asetukset oli listattu, kutsuttiin Facade-luokan funktiota, joka käy läpi listatut asetukset, validoi ne ja lisää mahdolliset ehdolliset pakollisuudet. Kun funktio palauttaa listan validoiduista asetuksista voidaan JUnit-testissä hyödyntää Assert.assertEquals('odotettu arvo', 'palautunut arvo') -toimintoa (Kuva 28). Siinä 'odotettu arvo' kohtaan asetetaan esimerkiksi odotettu boolean-arvo (tosi tai epätosi) ja 'palautunut arvo'-kohtaan sijoitetaan Facade-luokasta palautunut arvo. Jos 'odotettu arvo' ja 'palautunut arvo' eivät vastaa toisiaan, ilmoittaa testi siitä epäonnistumisella.

```
Assert.assertEquals(true, lomakeAsetuksetList.get(0).isRequired());
Assert.assertEquals(true, lomakeAsetuksetList.get(1).isRequired());
Assert.assertEquals(true, lomakeAsetuksetList.get(1).isOtsikko());
Assert.assertEquals(true, lomakeAsetuksetList.get(2).isRequired());
Assert.assertEquals(true, lomakeAsetuksetList.get(3).isRequired());
Assert.assertEquals(true, lomakeAsetuksetList.get(4).isRequired());
Assert.assertEquals(false, lomakeAsetuksetList.get(5).isRequired());
Assert.assertEquals(true, lomakeAsetuksetList.get(5).isKoodistossa());
```

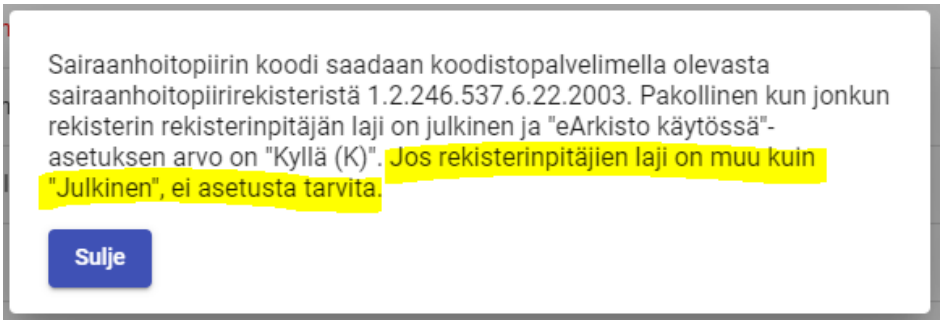
Kuva 28 Esimerkki Assert.assertEquals-funktion käytöstä

Kuten 'Lääkkeet'-lomakkeen asetusten kohdalla, myös 'Suostumukset'-lomakkeen osalta toteutettiin käyttöliittymään vastaava taulukko Angular Materialia hyödyntäen. Käyttöliittymän puolella erotettiin kuitenkin osa asetuksista 'Rekisterikonfiguraatio'-välilehdelle, jonne oli helppo kerätä kaikki rekisterikonfiguraatioon liittyvät asetukset yhteen. Jäljelle jääneet asetukset, jotka koskivat yleisesti suostumuksen hallintaa, listattiin 'Suostumukset lomake'-välilehdelle (Kuva 29).

Nimi	Parametrin arvo	Lomake	Kuvaus
Isäntä-otsikko	Isäntäorganisaatio	1016	i
Katuosoite-otsikko	Katuosoite	1016	i
OID-otsikko	OID	1016	i
Postitoimipaikka-otsikko	Postitoimipaikka	1016	i
Puhelin-otsikko	Yks.puhelin	1016	i
Sairaanhoitopiirin koodi	90	1016	i
Sairaanhoitopiirin nimi	Mediatrin testisairaanhoitopiiri	1016	i
Työnantajaluettelo	Sairaanhoitopiirin työnantajat	1016	i
Vaihde-otsikko	Vaihteenpuhelinnumero	1016	i
Y-tunnus-otsikko	Y-tunnus	1016	i
Yhteisrekisteri-informoinnin tuloste	0	1016	i
eArkisto käytössä	Kyllä	1016	i
eArkiston käyttäjäryhmä	Kaikki käyttäjät	1016	i
eResepti-informoinnin tuloste	0	1016	i
Rekisterinpitäjähistoria	Rekisterihistoria	1016	i

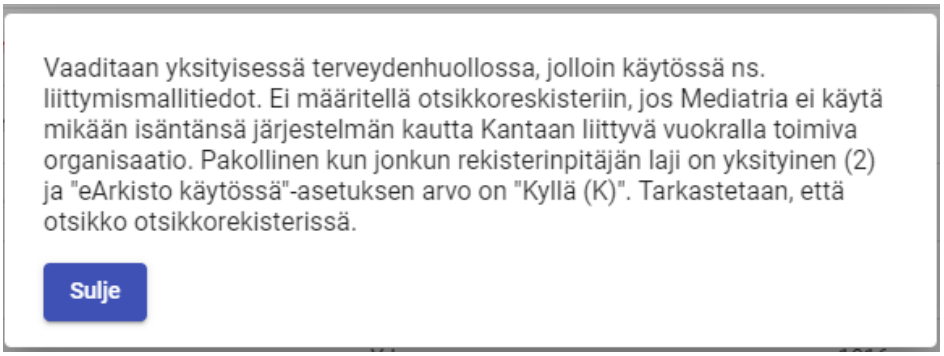
Kuva 29 'Suostumukset asetukset'-välilehti

Virheelliset tiedot korostettiin taulukossa punaisella värillä. Lisäksi listalla voi esiintyä oransilla värillä korostettu asetus. Oranssi väri kertoo tällä välilehdellä siitä, että otsikko puuttuu otsikkorekisteristä tai jos jonkun asetuksen olemassaolo ei ole välttämätöntä nykyisillä asetuksien arvoilla. Esimerkiksi ’Sairaanhoitopiirin koodi’-asetusta ei tarvita ollenkaan, jos yhdenkään ’Rekisterinpitäjän laji’-asetuksen arvo ei ole ’Julkinen’ (Kuva 30).



Kuva 30 Esimerkki ehdosta, jolla voidaan antaa varoitus ja korostaa asetus mahdollisesti oranssilla värillä

Ehdolliset pakollisuudet ovat tuotu esille käyttöliittymässä jokaisen asetuksen kuvausdialogiin (Kuva 31). Näin käyttäjän on helpompi selvittää minkä takia jokin tieto voi olla virheellinen, kun asetuksen ehdot on määritetty kyseisen asetuksen kuvaukseen.

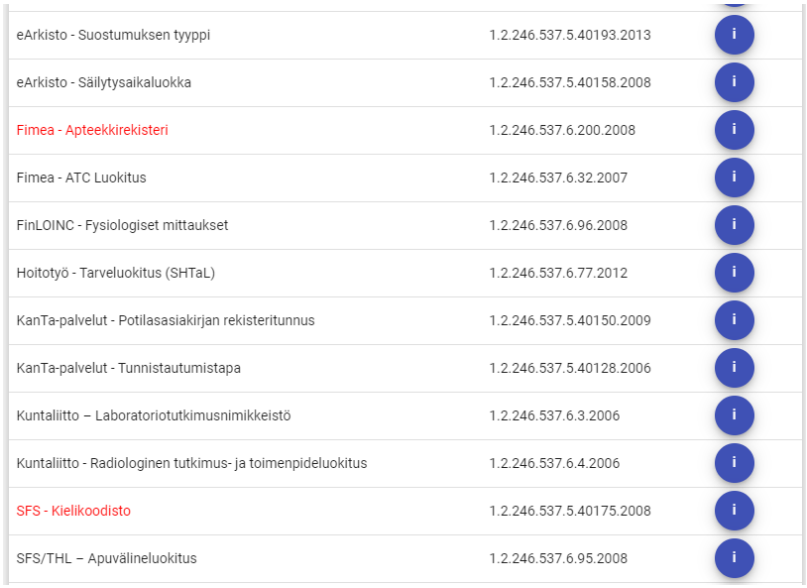


Kuva 31 Ehdollisten pakollisuuksien esille tuominen infodialogissa

5.2.3 'Koodistot'-välilehti

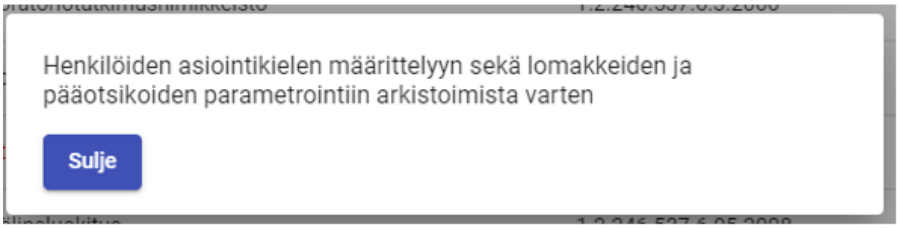
Koodistojen tarkoitus on yhtenäistää tietokantoja niin, että Kanta-arkistoon meneviä tietoja voidaan yksilöidä samalla koodilla, vaikka tieto tulisi eri järjestelmästä. Tällä tavoin saavutetaan ehyt kokonaisuus uudelleen luettavaa tietoa. Kanta-arkistoinnin kannalta olennaisten koodistojen tapauksessa tehtiin lista, johon koodistot on listattu. Koodistojen tarkastukseen hyödynnettiin koodistopalvelinta, joka suorittaa kyselyn tietokantaan. Kysely etsii koodistoa tietyllä koodistotunnisteella ja palauttaa tiedon käyttöliittymään, onko kyseinen koodisto ladattu Mediatriin.

Käyttöliittymässä näytetään käyttäjälle luettava lista olennaisista koodistoista (Kuva 32).



Kuva 32 Koodistoista koostettu välilehti

Puuttuvat koodistot korostetaan punaisella värillä ja jokaiselle koodistolle on asetettu kuvaus, joka kertoo mihin koodistoa käytetään (Kuva 33).

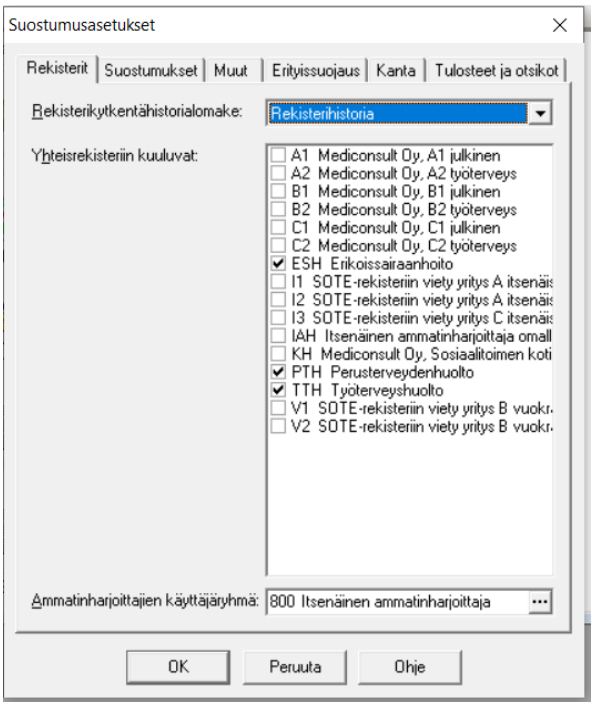


Kuva 33 Koodistolle generoitu kuvaus käyttötarkoituksesta

5.2.4 'Rekisterikonfiguraatio'-välilehti

Jokaisen julkisen ja yksityisen organisaation tiedot tulee löytyä terveydenhuollon rekistereistä, jotta organisaatioiden on mahdollista liittyä Kanta-palveluihin. Rekisterikonfiguraatioon liittyvien asetusten tarkastelu liittyy juuri terveydenhuollon rekisteriin ja sen tietoihin. Vaillinaiset tiedot voivat vaikuttaa suoraan arkistoinnin epäonnistumiseen, koska jokaisen organisaation perustiedot ovat pakollisia Kanta-yhteyttä varten. Sen takia rekisterikonfiguraatioon liittyvien asetusten läpikäyminen on tärkeä osa parametrien tarkastelusta ja validoinnista.

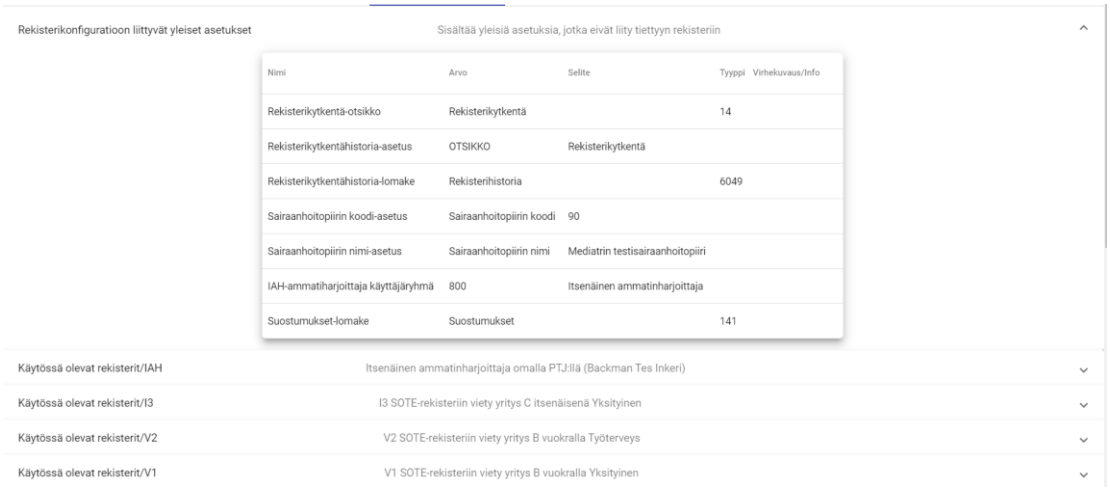
Rekisterikonfigurointiin liittyviä asetuksia haettiin ja tarkastettiin osittain jo aiemmin, kun haettiin 'Suostumukset'-lomakkeen 'Kanta'- ja 'Tulosteet ja otsikot'-välilehtien asetuksia. Tässä vaiheessa haettiin näiden lisäksi vielä 'Suostumukset'-lomakkeen 'Rekisterit'-välilehden asetukset (Kuva 34). Toteutuksessa päädyttiin tuomaan kuitenkin selkeyden vuoksi kaikki rekisterikonfiguraatioon liittyvät asetukset ja niiden oikeellisuus omalle välilehdelleen. Rekisterikonfiguroinnin kannalta uusia olennaisia asetuksia oli tarkastaa, että 'Suostumukset'-lomake on määritetty ja että sen asetuksiin on määritetty 'Rekisterikytkentähistoria'-lomake. Uutena asetuksena haettiin myös mahdollisesti määritetty ammatinharjoittajien käyttäjäryhmä. Lisäksi tarkastettiin, että 'Rekisterikytkentähistoria'-lomakkeeseen on määritetty otsikko, jonka alle listataan kaikki käytössä olevat rekisterit.



Kuva 34 Rekisterikonfiguraatioon liittyvät asetukset

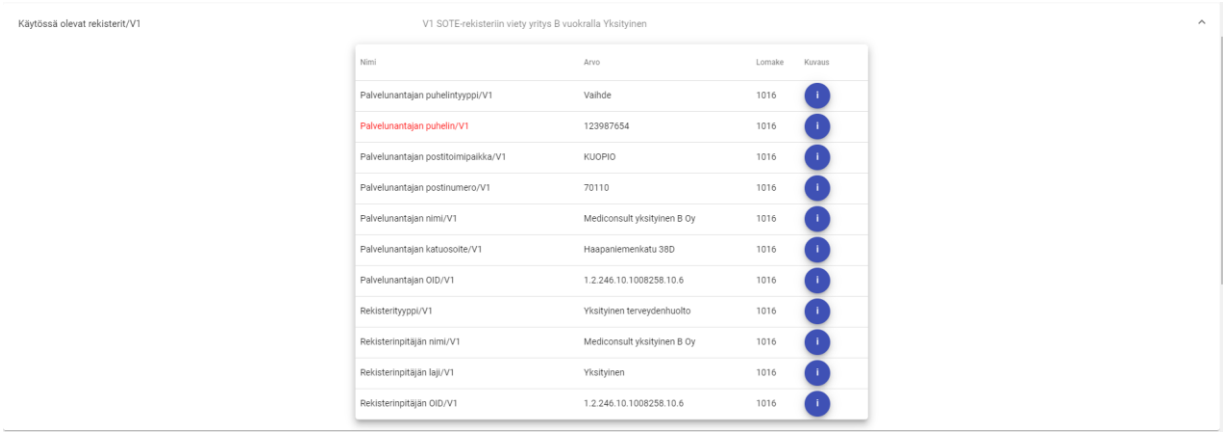
'Rekisterikonfiguraatio'-välilehdelle käyttöliittymään tuodaan jo aiemmassa vaiheessa haettuja ja validoituja rekisterien tietoja (muun muassa yhteystiedot, OID-tunnukset). Näiden lisäksi välilehdelle tuotiin yleisiä asetuksia liittyen rekisterikonfiguraatioon.

Käyttöliittymässä yksittäisten rekisterien tiedot ja yleiset rekisterikonfiguraatioon liittyvät asetukset ja tarkastukset on jaettu Accordion-tab-komponentteihin (Kuva 35).



Kuva 35 Accordion-tab-komponentteihin jaetut rekisterikonfiguraatioon liittyvät asetukset

Accordion-tab-komponentti toimii siten, että yhden otsikon alle on kerätty otsikon alle sopivia asetuksia taulukkoon. Yleisten asetusten lisäksi jokainen käytössä oleva rekisteri on jaettu omaan accordion-tab-komponenttiin (Kuva 36).



Kuva 36 Yksittäisen rekisterin tiedot kerättynä yhden otsikon alle

6 JATKOKEHITYS

Työkalun kannalta jatkokehitysmahdollisuuksia on edelleen paljon. Opinnäytetyöhön varatun ajan puitteissa onnistuttiin kuitenkin toteuttamaan hyvä pohja, jota voidaan jo nyt hyödyntää ohjelmistotuessa ja tekniikassa. Nykyinen toteutus tukee tärkeimpien asetusten listaamisen ja validoinnin. Validoitavia ja tarkasteltavia parametrejä löytyy kuitenkin edelleen paljon, mikä mahdollistaa työkalun jatkokehityksen.

Yksi hyvä esimerkki jatkokehityksestä olisi käyttäjänhallinta, jossa kirjautuneen käyttäjän tiedot voidaan lukea Mediatrin kontekstista. Tällaisessa tapauksessa opinnäytetyössä kehitetty työkalu käynnistettäisiin suoraan Mediatrista. Tällöin vältyttäisiin erillisen käyttäjärekisterin ylläpidolta ja erilliseltä käyttöoikeuksien hallinnalta.

Käyttäjänhallinnan lisäksi olennaisia tarkasteltavia parametrejä olisi vielä käyttäjien yksilöintitietojen asetusten tarkastelu ja ylläpito, kirjautumisyksiköiden asetusten tarkastelu ja ylläpito ja testipotilasasetusten tarkastelu. Testipotilaat täytyy pyytää erikseen aina Kanta-palveluilta ja niiden käyttäminen on sallittua esimerkiksi potilastietojärjestelmien testaus ja kehitystyössä.

7 YHTEENVETO

Opinnäytetyön lopputulos oli hyvä ja koen sen tärkeänä osana opintoja. Kokonaisuudessaan prosessi oli opettavainen, mutta myös haastava. Opinnäytetyö vaati pitkäjänteistä työtä, mutta työ palkittiin toimivalla työkalulla, johon saatiin toteutettua teknisesti tärkeimmät osiot työntilaajan kannalta. Lopputuloksen myötä yritys pystyy hyödyntämään työkalua omassa ohjelmistotuessaan ja heillä on toimiva pohja jatkokehittää työkalua eteenpäin.

Opinnäytetyön aikana pääsin oppimaan paljon uutta. Projektin alkuvaiheessa minulla oli jonkinlainen kuva Rest-pohjaisen websovelluksen kehittämisestä, mutta prosessin aikana huomasin, että joudun omaksumaan ja opettelemaan paljon uutta. Etenkin käyttöliittymä puolta pääsin miettimään paljon enemmän kuin esimerkiksi koulussa opiskeltujen kurssien aikana. Tarkoitus oli kehittää luettava ja selkeä käyttöliittymä, koska työkalu tulee yrityksen käyttöön ja koska kehitetyn työkalun on tarkoitus helpottaa virheellisten asetusten löytämistä.

Teoriaosuuden työstäminen ja teknisen osion toteuttaminen oli mielenkiintoinen prosessi. Teoriaa varten täytyi sisäistää todella paljon uutta tietoa ja samalla syventää omaa osaamista käytettyjen tekniikoiden osalta. Heti alkuvaiheessa huomasin, että vain koulussa opitut asiat eivät olisi riittäviä hyvän lopputuloksen saamiseksi. Uusien asioiden oppiminen ja oivaltaminen teki kuitenkin prosessista mielenkiintoisen ja palkitsevan. Teknisen toteutuksen aikana sain hyvin tukea yrityksestä, mistä oli suuri hyöty opinnäytetyön etenemisen kannalta.

Opinnäytetyön aikana pääsin toteuttamaan teknisesti kokonaisen websovelluksen, jossa pääsin työstettiin sekä palvelin- ja käyttöliittymäkoodia. Oma tietämys websovellusten kehityksestä lisääntyi huomattavasti ja prosessin aikana opin tekemään uudelleen käytettävää koodia, jota voidaan hyödyntää useamman kerran sovelluksen eri osissa.

Kokonaisuudessaan koin oman ammatillisen tietämyksen lisääntyneen ja koin työn lisäävän ammatillisia valmiuksia ohjelmistokehittäjänä. Työ tarjosi hyvän kokonaiskuvan ohjelmistokehittäjän työstä ja sen vaatimuksista.

LÄHTEET JA TUOTETUT AINEISTOT

Angular. Angular Material. [Online] [Viitattu: 6. Marraskuu 2019.] https://material.angular.io/.

—. Architecture overview. [Online] [Viitattu: 28. Syyskuu 2019.] https://angular.io/guide/architecture.

—. Introduction to components. [Online] [Viitattu: 28. Syyskuu 2019.] https://angular.io/guide/architecture-components.

—. Introduction to modules. [Online] [Viitattu: 28. Syyskuu 2019.] https://angular.io/guide/architecture-modules.

—. Introduction to services and dependency injection. [Online] [Viitattu: 28. Syyskuu 2019.] https://angular.io/guide/architecture-services.

—. Services and dependency injection. [Online] [Viitattu: 28. Syyskuu 2019.] https://angular.io/guide/architecture.

Bhatia, Krishna. GeeksforGeeks - Difference between JDK, JRE and JVM. [Online] [Viitattu: 26. Tammikuu 2019.] https://www.geeksforgeeks.org/differences-jdk-jre-jvm/.

Craig, Guyer;Gene, Milener ja Carl, Rabeler. 2017. What is ODBC? [Online] Microsoft, 1. Tammikuu 2017. [Viitattu: 29. Kesäkuu 2019.] https://docs.microsoft.com/en-us/sql/odbc/reference/what-is-odbc?view=sql-server-2017.

eclipse. What is Eclipse? [Online] [Viitattu: 28. Syyskuu 2019.] https://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm.

Evans, Ben ja Warburton, Richard. 2014. Java SE 8 Date and Time. [Online] Helmikuu 2014. [Viitattu: 8. Elokuu 2019.] https://www.oracle.com/technetwork/articles/java/jf14-date-time-2125367.html.

Fadatare, Ramesh. 2018. Hibernate Framework Overview - Architecture and Basics. [Online] 27. Marraskuu 2018. [Viitattu: 12. Syyskuu 2019.] https://www.javaguides.net/2018/11/hibernate-framework-overview-architecture-bascis.html.

—. 2019. Introduction to the Java Persistence API. [Online] 3. Tammikuu 2019. [Viitattu: 15. Syyskuu 2019.] https://www.javaguides.net/2019/01/introduction-to-java-persistence-api.html.

—. 2018. Java JDBC API Overview. [Online] 11. Lokakuu 2018. [Viitattu: 15. Syyskuu 2019.] https://www.javaguides.net/2018/10/java-jdbc-api-overview.html.

Git. About. [Online] [Viitattu: 28. Syyskuu 2019.] https://git-scm.com/about.

Gupta, Krishna. 2016. ToTheNew - Introduction to Wildfly. [Online] 9. Marraskuu 2016. [Viitattu: 28. Tammikuu 2019.] http://www.tothenew.com/blog/introduction-to-wildfly/.

Gupta, Lokesh. HowToDoInJava - What is Java programming language? [Online] [Viitattu: 26. Tammikuu 2019.] https://howtodoinjava.com/java/basics/what-is-java-programming-language/.

Herawan, Dwika P. 2019. What is MySQL: MySQL Explained For Beginners. [Online] 29. Huhtikuu 2019. [Viitattu: 21. Syyskyy 2019.] https://www.hostinger.com/tutorials/what-is-mysql.

Jaakkola, Ensi-Maria. 2019. *Mediconsult-esittely.* s.l. : Mediconsult Oy, 2019.

Javin, Paul. 2018. 20 Useful Libraries Java Programmers Should Know. [Online] 27. Heinäkuu 2018. [Viitattu: 8. Syyskuu 2019.] https://dzone.com/articles/20-useful-open-source-libraries-for-java-programme.

Jethva, Hitesh. How to install WildFly (JBoss) Java Application Server on Ubuntu 18.04. [Online] [Viitattu: 28. Syyskuu 2019.] https://www.howtoforge.com/tutorial/ubuntu-wildfly-jboss-installation/.

Khatri, Prem. THE ELUSIVE DELPHI PROGRAMMERS. [Online] [Viitattu: 11. Elokuu 2019.] https://www.chetu.com/blogs/technical-perspectives/delphi-evolution.php.

Kolmakow, Tom. 2016. *Järjestelmäarkkitehtuurikuvaus.* 2016.

—. 2019. *Sovellusarkkitehtuuri.* Heinäkuu 2019.

Leahy, Paul. 2019. ThoughtCo - What is Java? [Online] 19. Tammikuu 2019. [Viitattu: 26. Tammikuu 2019.] https://www.thoughtco.com/what-is-java-2034117.

Leino, Raili. 2000. Solid Information Technologyn älyverkko syntyy suomalaisvoimin. [Online] 30. Maaliskuu 2000. [Viitattu: 30. Syyskuu 2019.] https://www.tekniikkatalous.fi/uutiset/solid-information-technologyn-alyverkko-syntyy-suomalaisvoimin/338987c4-7b8c-31c5-98cf-d8c0510a4f7a.

Lindström, Jan;ym. 2013. IBM solidDB: In-Memory Database Optimized for ExtremeSpeed and Availability. [Online] IBM Helsinki Lab Oy IBM Finland Ab, Tammikuu 2013. [Viitattu: 22. Syyskuu 2019.] https://www.researchgate.net/publication/269402357_IBM_solidDB_In-Memory_Database_Optimized_for_Extreme_Speed_and_Availability.

Lofhjelm, Janne. Tietokone työvälineenä - Lapio. *Osa 2 - Versionhallinta: Git ja Github.* [Online] [Viitattu: 13. Marraskuu 2019.] https://tkt-lapio.github.io/git/.

Marshall, Alex. 2015. Top 10 Open Source Java and JavaEE Application Servers. [Online] 9. Huhtikuu 2015. [Viitattu: 15. Syyskuu 2019.] https://blog.idrsolutions.com/2015/04/top-10-open-source-java-and-javaee-application-servers/.

Mastertheboss. 2019. utorials for WildFly Application Server, Openshift, JBoss Projects and Enterprise Applications. [Online] 11. Tammikuu 2019. [Viitattu: 11. Elokuu 2019.] http://www.mastertheboss.com/other/faqs/what-is-wildfly.

Mediconsult Oy. Keitä olemme: Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 29. Kesäkuu 2019.] https://www.mediconsult.fi/yritys/keita-olemme.

—. *Mediatri perusterveydenhuoltoon.* s.l. : Mediconsult Oy.

—. *Mediatrin eResepti- ja eArkisto-liityntä.*

—. Ratkaisut: asiakkuuden hallinta. *Mediconsult Oy kotisivusto.* [Online] [Viitattu: 29. Kesäkuu 2019.] https://www.mediconsult.fi/ratkaisut/asiakkuuden-hallinta.

—. Ratkaisut: Mobiilikirjaaminen, Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 29. Kesäkuu 2019.] https://www.mediconsult.fi/ratkaisut/mobiilikirjaaminen.

—. Ratkaisut: Omahoito, Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 29. Kesäkuu 2019.] https://www.mediconsult.fi/ratkaisut/omahoito.

—. Ratkaisut: Sähköinen asiointi, Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 29. Kesäkuu 2019.] https://www.mediconsult.fi/ratkaisut/sahkoinen-asiointi.

—. Ratkaisut: Sähköinen resepti, Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 29. Kesäkuu 2019.] https://www.mediconsult.fi/ratkaisut/sahkoinen-resepti.

—. Ratkaisut: Tiedolla johtaminen, Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 2019. Kesäkuu 2019.] https://www.mediconsult.fi/ratkaisut/tiedolla-johtaminen.

—. Ratkaisut: Toiminnanohjaus, Mediconsult Oy. *Mediconsult Oy kotisivu.* [Online] [Viitattu: 29. Kesäkuu 2019.]

Microsoft . Visual Studio Code. [Online] [Viitattu: 25. Tammikuu 2019.] https://code.visualstudio.com/.

Mikkonen, Juuso. 2017. Rest on nettipalveluiden yhteinen kieli. [Online] 27. Toukokuu 2017. [Viitattu: 29. Syyskuu 2019.] https://www.tivi.fi/uutiset/rest-on-nettipalveluiden-yhteinen-kieli/23703ab5-dd19-383e-a422-ebfc3d910583.

Oracle. Java Platform, Enterprise Edition (Java EE) 8. *Your First Cup: An Introduction to the Java EE Platform.* [Online] [Viitattu: 23. Syyskuu 2019.] https://javaee.github.io/firstcup/java-ee002.html.

—. The Java EE Tutorial. *Distributed Multitiered Applications.* [Online] [Viitattu: 23. Syyskuu 2019.] https://javaee.github.io/tutorial/overview004.html.

Patsov, Martin. 2017. Top 10 Java libraries for saving time. [Online] Syyskuu 2017. [Viitattu: 28. Heinäkuu 2019.] https://jaxenter.com/top-10-java-libraries-137587.html.

Paul, Javin. 2018. Top 5 Free Courses to Learn Git and Github—Best of Lot. [Online] 7. Marraskuu 2018. [Viitattu: 28. Syyskuu 2019.] https://hackernoon.com/top-5-free-courses-to-learn-git-and-github-best-of-lot-2f394c6533b0.

Penland, Jon. 2018. Learn Delphi Programming: Build Compact Applications. [Online] 12. Joulukuu 2018. [Viitattu: 11. Elokuu 2019.] https://www.whoishostingthis.com/resources/delphi/.

Rissanen, Kimmo. 2019. *Mediatrin konfigurointi.* Kuopio, 7. Elokuu 2019.

—. 2019. *Parametrointi työkalun tarve.* Kuopio, 2019. Heinäkuu 2019.

Rouse, Margaret. 2017. Eclipse (Eclipse Foundation). [Online] Elokuu 2017. [Viitattu: 13. Elokuu 2019.] https://searchmicroservices.techtarget.com/definition/Eclipse.

—. 2018. RDBMS (relational database management system). [Online] Huhtikuu 2018. [Viitattu: 21. Syyskuu 2019.] https://searchdatamanagement.techtarget.com/definition/RDBMS-relational-database-management-system.

—. 2015. SearhSoftwareQuality - polygot programming. [Online] Syyskuu 2015. [Viitattu: 4. Elokuu 2019.] https://searchsoftwarequality.techtarget.com/definition/polyglot-programming.

Terveyden ja hyvinvoinnin laitos (THL). 2018. Potilastiedon arkiston toimintamallit. [Online] Terveyden ja hyvinvoinnin laitos (THL), 18. Syyskuu 2018. [Viitattu: 8. Joulukuu 2019.] https://www.kanta.fi/documents/20143/106832/Potilastiedon+arkiston+toimintamallit.pdf/39510f48-3aec-0bcb-1513-af182fc00d5d.

Thakur, Dinesh. What is a Database Server. [Online] [Viitattu: 28. Syyskuu 2019.] http://ecomputernotes.com/fundamental/what-is-a-database/what-is-a-database-server.

Tutorialspoint. Java DOM Parser - Overview. [Online] [Viitattu: 8. Syyskuu 2019.] https://www.tutorialspoint.com/java_xml/java_dom_parser.htm.

—. Java XML - Parsers. [Online] [Viitattu: 8. Syyskuu 2019.] https://www.tutorialspoint.com/java_xml/java_xml_parsers.htm.

w3schools. w3schools - Java. [Online] [Viitattu: 26. Tammikuu 2019.] https://www.w3schools.com/java/.

WildFly. What is WildFly? [Online] [Viitattu: 29. Syyskuu 2019.] https://wildfly.org/about/.

Vogel, Lars ja Pfaff, Fabian. 2012. Unit tests with Mockito - Tutorial. [Online] 2012. [Viitattu: 8. Elokuu 2019.] https://www.vogella.com/tutorials/Mockito/article.html.

Vogella. 2007. Unit Testing with JUnit - Tutorial. [Online] 2007. [Viitattu: 8. Syyskuu 2019.] https://www.vogella.com/tutorials/JUnit/article.html.