Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

2019

Ira Lindholm

MIKROPALVELUN TOTEUTUS IKÄIHMISTEN VERKKOPALVELULLE

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma

2019 | 32 sivua

Ira Lindholm

MIKROPALVELUN TOTEUTUS IKÄIHMISTEN VERKKOPALVELULLE

Mikropalvelut ovat ohjelmistokehityksessä käytetty toteutustapa, jonka idea on luoda joukko pienempiä itsenäisesti toimivia osia isojen kokonaisuuksien sijaan. Tämä tuo merkittäviä muutoksia ohjelmistokehitykseen, ja siihen kuuluvaan liiketoimintaan, koska jokaista osaa pystytään päivittämään toisistaan riippumattomasti ja kevyesti. Tämä tarkoittaa kalliiden ja virhealttiiden järjestelmäpäivityksien unohtamista, koska mikropalveluita on mahdollista kehittää tarpeiden mukaan myöhemmin, ja niitä on helppo vaihtaa uuteen. Mikropalveluiden käyttöä tarkasteltiin opinnäytetyössä ohjelmistoprojektin näkökulmasta. Siinä käytiin läpi yhden palvelun toteutusprosessia. Opinnäytetyön tavoitteena oli kuvata mikropalvelun toteutusta käytännönläheisesti ja sitä, miten sillä voidaan tukea ikäihmisten neuvontaa ja ohjausta verkkopalvelussa.

Opinnäytetyön kokonaisuutta laatiessa keskityttiin isoksi osaksi mikropalveluiden tutkimiseen. Tämän lisäksi opinnäytetyössä tutkittiin Varsinais-Suomen keskitettyä asiakas- ja palvelunohjaushanketta nimeltä KomPAssi, jossa mikropalvelurakennetta hyödynnettiin erillisenä ohjelmistoprojektina. Opinnäytetyön tutkimusmenetelmä oli konstruktiivinen ja opinnäytetyön toteutuksessa hyödynnettiin mikropalveluihin sekä hankkeeseen perustuvia aineistoja ja kirjallisuutta. Lisäksi toteutuksessa hyödynnettiin mikropalvelun tekniseen toteutukseen liittyviä aineistoja, jotta siihen kuuluvat teknisen tason toimet pystyttiin kuvaamaan selkeästi.

Opinnäytetyön tavoitteet onnistuivat suurilta osin ja lopputulemana on ymmärrettävä selostus yhden mikropalvelun toteutuksesta, ja siihen liittyvän rajapinnan toiminnallisuudesta ikäihmisille tarkoitetussa verkkopalvelussa. Tavoitteista jäi uupumaan mikropalveluun liittyvän rajapinnan konkreettinen näkymä verkkopalvelussa, koska se julkaistaan verkkopalvelun myöhemmässä tuotantoversiossa.

Johtopäätöksenä voidaan todeta mikropalveluiden tuovan hyötyä, mutta myös monimutkaisuutta ohjelmistokehityksessä. Mikropalvelun käsite on melko tulkinnanvarainen ja herättää kysymyksiä siitä, mihin pisteeseen asti palvelua on hyödyllistä pilkkoa pienemmiksi osiksi ja minkä kokoista palvelua voidaan kutsua mikropalveluksi.

ASIASANAT:

mikropalvelut, verkkopalvelu, ikäihminen, keskitetty asiakas- ja palvelunohjaus, ohjelmistoprojekti

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2019 | 32 pages

Ira Lindholm

IMPLEMENTING A MICROSERVICE FOR A WEB SERVICE OF THE ELDERLY

Microservices are an implementation method used in software development. Their purpose is to create a set of smaller independently operating elements instead of larger entities. This brings significant changes to software development and its business because each part can be updated independently and lightly. This means eliminating expensive and error-prone system upgrades because it is possible to develop easily replaceable microservices later according to needs. This Thesis examines the use of microservices from software project's point of view where the process of one service was reviewed. The objective of the Thesis was to describe the implementation of a microservice pragmatically and how it can support counseling and guidance of elderly people in a web service.

In addition, the Thesis also examined Southwest Finland's centralized customer and service management called KomPAssi, where the microservice structure was utilized as a separate software project. The research method of the Thesis was qualitative and in the implementation of the thesis, materials and literature based on microservices and project were utilized. In addition, the material used in the technical implementation of microservice was utilized to clearly describe activities on technical level.

The objectives of the Thesis were mainly completed and the result is an understandable description of the implementation of one microservice and its interface functionality in a web service for the elderly people. The concrete view for the microservice's interface in the web service is not included in this Thesis because it will be published in a later version.

As a conclusion, microservices can bring benefits but also complexity in software development. The concept of a microservice is rather ambiguous and raises questions about the size of the service that can be called microservice and how much a service should be reduced to smaller segments that the microservice is still useful.

KEYWORDS:

microservices, web service, elderly, centralized customer and service guidance, software project

SISÄLTÖ

KUVAT

KUVIOT

SANASTO

API-yhdyskäytävä	Ohjelmisto, joka toimii järjestelmän palveluita ylläpitävässä tietoverkossa ja tarjoaa turvallisuus- sekä hallintaominaisuuksia. (Everard 2017).
Asiakas	Ikäihminen, joka käyttää verkkopalvelua.
CRUD	Akronyymi sanoille create, read, update ja delete. Se kuvaa neljää perus funktiota, jotka pitävät tallenteet ajankohtaisena ja pysyvänä tietokannassa (Watts 2018).
Drupal	Selainpohjainen sisällönhallintajärjestelmä, jonka avulla luodaan, hallitaan ja julkaistaan verkkosivulle sisältöä (Suomen Drupal-yhdistys 2014).
Funktio	Tietokoneohjelman pääohjelmaa pienempi itsenäinen aliohjelma, joka suorittaa tietyn toiminnon.
Granulariteetti	Julkaistavan käsitteen taso ja koko (Toivanen 2018).
HTTP	Joukko sääntöjä tiedostojen, kuten tekstin, kuvien ja äänien siirtämiseen maailmanlaajuisessa tietoverkossa (Rouse 2006).
Integraatio	Kahden erillisen asian yhdistäminen.
I&O-hanke	Kehitetään ikäihmisten kotihoitoa ja vahvistetaan kaikenikäisten omaishoitoa-kärkihanke.
JSON	Yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen (Copter Labs 2019).
Klusteri	Rykelmä, joka koostuu samoista asioista.
KomPAssi	Varsinais-Suomen keskitetyn asiakas- ja palvelunohjaushankkeen nimi.
Maven	Java-projektien rakentamiseen ja hallintaan käytettävä työkalu (The Apache Software Foundation 2019).
Nanopalvelu	Liian pienen raekoon omaava palvelu, joka julkaisee yksittäisen funktion palvelukerroksessa (Toivanen 2018).

OpenAPI määrittely	Määrittelee standardin kieliagnostisen käyttöliittymän REST(ful) ohjelmointirajapinnoille, jonka avulla ihmiset ja tietokoneet voivat löytää ja ymmärtää palvelun mahdollisuuksia ilman lähdekoodia, dokumentointia tai tietoverkon liikennetarkastusta (SmartBear Software Inc. 2019a).
PAT-ydintietomalli	Tietomalli, jota kuvataan palveluiden, digitaalisen asioinnin ja toimijoiden muodostamaksi kokonaisuudeksi ja on keskeinen osa Turun ydintiedon hallintaa (Drive Turku 2015).
RAI-järjestelmä	Järjestelmä, jolla arvioidaan asiakkaan palvelutarvetta ja laaditaan hoito-, kuntoutus- ja palvelusuunnitelmia (Terveyden ja hyvinvoinnin laitos 2017).
REST(ful) API	Ohjelmointirajapinta, joka käyttää HTTP pyyntöjä hakemaan, luomaan, päivittämään tai poistamaan tietoa (Rouse 2019).
Swagger	Ohjelmointirajapinnan rakentamisessa käytettävä työkalu (SmartBear Software Inc. 2019b).
Tunnistautumispalvelu	Palvelu, jonka kautta asiakas tunnistaa henkilöllisyytensä.
Verkon latenssi	Internetissä liikkuvan tiedon aikaviive.

1 JOHDANTO

Tämä opinnäytetyö käsittelee mikropalveluita ja ikäihmisten neuvonnan sekä ohjauksen tukemista mikropalveluita käyttäen. Aiheen valintaan vaikutti mikropalveluiden ajankohtaisuus ohjelmistokehityksessä sekä henkilökohtainen kiinnostukseni aiheeseen. Mikropalveluilla tarkoitetaan joukkoa itsenäisesti toimivia komponentteja, jotka yhdessä muodostavat isomman kokonaisuuden. Ne ovat suosittuja ohjelmistokehityksessä, koska ne ovat vikasietoisia ja helposti hallittavia. (Järvenpää 2017.) Lisäksi työskentelen ohjelmistoyrityksessä mikropalveluita hyödyntävän ohjelmistoprojektin parissa, joten työ tukee luonnollisesti myös ammatillista kasvuani.

Tavoitteeni on selvittää mitä mikropalvelut ovat, miten ja milloin niitä kannattaa hyödyntää sekä niiden suunnittelu- ja hallintaperiaatteet. Lisäksi selvitän mitä liittyy mikropalveluiden tuomiin haasteisiin ja monimutkaisuuteen. Näiden ymmärtäminen on tärkeää päätettäessä millä tavalla järjestelmä toteutetaan ja mikä on juuri toteutettavalle järjestelmälle sopiva toteutustapa. Mikropalvelut eivät sovi kaikkiin järjestelmäratkaisuihin, mutta niitä kannattaa harkita erityisesti monitoimittajaympäristössä niiden ketteryyden vuoksi. Mikropalvelut nimittäin mahdollistavat uusien ominaisuuksien käyttöönoton helposti ja kustannustehokkaasti. (Tirkkonen 2018.) Teoriaosuuden lisäksi opinnäytetyössä käsitellään mikropalvelun ja sen osan toteutus ohjelmistoprojektissa. Toteutuksella osoitetaan, miten mikropalvelua hyödyntävään verkkopalveluun suunnitellaan ja toteutetaan yksi mikropalvelu sekä siihen liittyvä rajapinta. Toteutuksen kohteena on ikäihmisten neuvonta- ja ohjausverkkopalvelu.

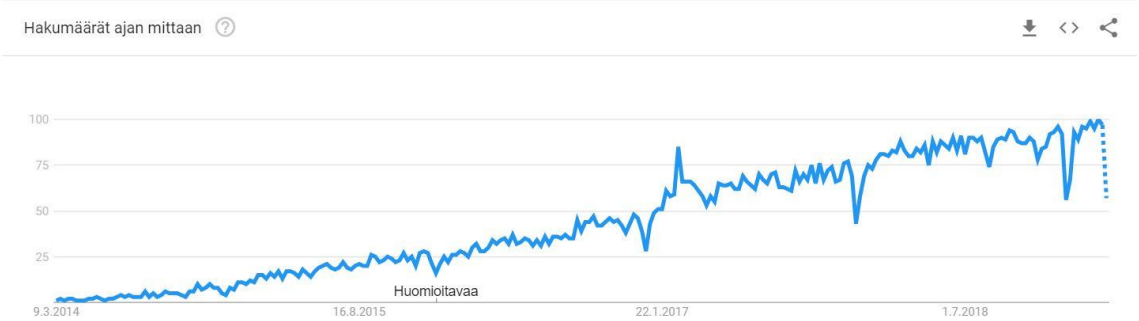
Opinnäytetyön toisessa luvussa perehdytään mikropalveluihin, niiden tarpeeseen, haittoihin ja siihen, mikä on tuonut ne ohjelmistokehittäjien suosioon. Tämän jälkeen kolmannessa luvussa esitellään KomPAssi-hanke, sen sisältö, laajuus ja haasteet. Neljännessä luvussa käsitellään teknistä toteutusprosessia eli mikropalvelun, ja siihen kuuluvan rajapinnan toteutuksen vaiheita ohjelmistoprojektissa. Lopuksi on omaa pohdintaani mikropalveluista sekä opinnäytetyön aikana heränneistä ajatuksista ja tavoitteista.

2 MIKROPALVELUT

Tässä luvussa kerrotaan mikropalveluista, ja niiden suosiosta ohjelmistokehityksessä. Mikropalvelut tuovat mukanaan paljon hyötyjä ja ovat joskus kannattavampi valinta perinteisen monoliittijärjestelmän sijaan. Monoliitilla tarkoitetaan järjestelmäratkaisua, joka on toteutettu yhtenä isona koodinippuna (Saarelainen 2016). Luvussa kerrotaan lisäksi mikropalveluiden suunnittelu- ja hallintaperiaatteista sekä tarpeista ja haitoista. Kuten kaikki järjestelmäratkaisut, myös mikropalvelut sisältävät haittoja. Haittoja voidaan kuitenkin hallita, kun kiinnitetään jo suunnitteluvaiheessa huomiota niihin.

Mikropalveluiden suosioon on vaikuttanut tarve julkaista usein ja nopeasti. Ennen mikropalveluita julkaisu on ollut mahdollista harvoin ja hitaasti, kun toteutusratkaisuna on käytetty perinteistä monoliittimallia. Nopean julkaisemisen lisäksi mikropalvelut ovat vikasietoisia ja helposti hallittavia. Se vaikuttaa paljon järjestelmän ylläpidettävyyteen pidemmällä tähtäimellä. Kehittäjien työ helpottuu, koska myös järjestelmän päivittäminen on vaivattomampaa. Vuosien saatossa kasvaneen monoliittijärjestelmän ylläpito on huomattavasti työläämpää sen suuruuden ja kankeuden vuoksi. (Saarelainen 2016.) Tämän pohjalta voidaan todeta mikropalveluiden tuovan ohjelmistokehitykseen joustavuutta uudenlaisella tavalla, mutta erityisesti uuden tavan toteuttaa järjestelmä.

Kuviosta nähdään hakumäärät sanasta mikropalvelut viiden vuoden ajalta koko maailmassa (Kuvio 1). Kuviosta voidaan huomata hakujen yleistyneen vuosi vuodelta suuremmaksi, alkaen vuodesta 2014. Kuviota voidaan tulkita suhteutettuna vasemmalla olevien numeroiden avulla.



Kuvio 1. Mikropalveluja koskevat hakumäärät (Google Trends 2019).

2.1 Mikropalvelut ja niiden tarve

Mikropalveluarkkitehtuuri on arkkitehtuurimalli, joka tarkoittaa järjestelmäkokonaisuuden toteuttamista hajautetusti ja itsenäisesti toimivina palveluina. Erilliset pienet mikropalvelut muodostavat keskenään suuremman kokonaisuuden. (Tirkkonen 2018.) Palvelut voidaan hajauttaa samanaikaisesti pilveen tai lähiverkkoon, ja ne pystyvät kommunikoimaan keskenään ohjelmointirajapintojen avulla (Ojala 2019).

Mikä sitten on pieni, kun puhutaan mikropalveluista? Mikropalvelun määritelmä on melko tulkinnanvarainen, ja sen kokoa voidaan määritellä esimerkiksi organisatorisen linjauksen näkökulmasta. Tämä tarkoittaa palvelun istuvuutta tiimin rakenteeseen eli miten hyvin pieni tiimi kehittäjiä pystyy hallitsemaan koodipesää. Tämän perusteella voidaan päätellä, että mikropalvelu on tarpeeksi pieni, kun se on helposti hallittavissa pienen tiimin kesken. Tämä on vain yksi näkökulma. Sam Newmanin yhden argumentin mukaan mikropalvelu voisi olla tarpeeksi pieni myös silloin, kun se ei tunnu enää isolta (Newman 2015, 2-3). Tämä herättää kysymyksen, voiko mikropalvelu olla liian pieni? Kutsutaanko sitä silloin enää mikropalveluksi? Mikropalvelua pienempää palvelua kutsutaan nanopalveluksi. Jos palvelu toteuttaa vain yksittäisen funktion, sitä voidaan pitää nanopalveluna. Tällöin palvelun raekokoa eli granulariteettia voidaan pitää todella pienenä. (Toivanen 2018.) Tämän perusteella raekokoa voidaan mitata sen sisältämien toiminnollisuuksien määrän mukaan, eikä sen fyysisen koon perusteella.

On tärkeää huomioida, mitä palvelun pilkkominen pienemmiksi osiksi tuo mukanaan. Pilkkominen maksimoi mikropalveluiden hyötyjä, mutta samalla niiden haittoja ja hajautettujen järjestelmien mukana kulkevaa monimutkaisuutta. (Newman 2015, 2-3.) Mikropalveluita tarvitaan niiden tuomien hyötyjen takia. Näitä ovat:

- vikasietoisuus
- käyttöönoton helppous
- teknologioiden erilaisuus
- skaalautuvuus
- organisatorinen linjaus
- koottavuus
- vaihdettavuuden optimointi. (Newman 2015, 4-8.)

Toisistaan riippumattomien palvelujen ansiosta nämä hyödyt ovat mahdollisia. Mikropalvelujen vikasietoisuudella tarkoitetaan sitä, ettei järjestelmän yhdessä palvelussa ilmenevä virhe vaikuta muihin palveluihin. Virhe on helppo eristää muista järjestelmän osista ja laittaa kuntoon. Muut palvelut jatkavat silloin toimintaansa normaalisti. Monoliittisen järjestelmän ongelma on se, että virheen sattuessa koko järjestelmä kaatuu. Newman toteaa kirjassaan (2015, 5), että on tärkeää ymmärtää hajautettujen järjestelmien erilaisia vikoja, koska niitä ymmärtämällä voidaan hyötyä mikropalvelujen vikasietoisuudesta asianmukaisesti. Mikropalveluita on samalla periaatteella myös helppo ottaa käyttöön ja liittää uusi palvelu osaksi isompaa kokonaisuutta muista palveluista riippumattomasti. Yksi palvelu on yleensä kirjoitettavissa noin kahdessa viikossa, koska ne koostuvat perinteisesti vain sadoista riveistä koodia. Tämä mahdollistaa uusien ominaisuuksien toimittamisen asiakkaalle nopeammin ja kustannustehokkaammin kuin monoliittisessa järjestelmäratkaisussa, eikä asiakas jää jälkeen kilpailijoista.

Monoliittijärjestelmä koostuu yhdestä isosta koodikimpusta, ja siitä johtuen uusia ominaisuuksia saadaan vietyä tuotantoon hitaasti ja kankeasti. Yksi pienikin muutos monoliittijärjestelmään aiheuttaa koko järjestelmän läpikäynnin ja testauksen. Tästä syystä uusia ominaisuuksia kasaantuu paljon ennen tuotantoon vientiä, joka lisää riskialttiutta. Isoissa järjestelmissä ei voida ennakoida vaikutuksia varmaksi ja siksi pienellä muutoksella voi olla suuret vaikutukset ja samalla suuret riskit. Mikropalvelut ovat tässä mielessä selkeästi riskittömämpiä. (Newman 2015, 5-6.)

Toisistaan riippumattomat mikropalvelut mahdollistavat erilaisten teknologioiden käyttämisen ja testaamisen järjestelmän eri osissa. Näin ollen kehittäjiä ei ole sidottu vain yhden teknologian hyödyntämiseen. Esimerkiksi yhdessä palvelussa voidaan käyttää ohjelmointikielenä Javaa ja toisessa Pythonia. Jos uuden teknologian testaaminen epäonnistuu, se voidaan jälleen eristää muista palveluista riippumattomasti. Monoliittijärjestelmässä mikä tahansa muutos vaikuttaa heti koko järjestelmään, mikä tekee uuden teknologian testaamisesta todella hankalaa ja melkein mahdotonta. (Newman 2015, 4.) Tästä päätellen mikropalveluiden tuoma ominaisuus testata tehokkaasti eri teknologioita on sellaisille ohjelmistoyrityksille tarpeellista, jotka tahtovat olla järjestelmän kehityksen edelläkävijöitä.

Mikropalvelujärjestelmässä on mahdollista skaalata vain niitä palveluita, jotka sitä tarvitsevat. Skaalautuvuus mahdollistaa järjestelmän eri osien ajamisen pienemmillä ja heikoimmilla laitteistoilla, mikä esimerkiksi helpottaa verkkosivujen ruuhka-aikojen hallinnointia. Monoliittijärjestelmän hankaluus on, että kaikki on skaalattava yhteen.

Hajautetut järjestelmät ja palveluorientoituneet arkkitehtuurit mahdollistava toiminnollisuuksien uudelleenkäytön. Mikropalveluiden avulla sallitaan toiminnollisuuksien olevan käytettävissä eri tavoin ja eri tarkoituksiin eli niitä voidaan hyödyntää myös aivan toisessa järjestelmässä. Yksittäisten palveluiden pienen kokonsa ansiosta, niiden korvaamisesta tai poistamisesta ei aiheudu suuria vaivoja. Kynnys kirjoittaa palvelu kokonaan uudelleen on todella pieni ja se voidaan tehdä lyhyellä aikataululla. (Newman 2015, 5-8.)

2.2 Suunnitteluperiaatteet

Mikropalvelut voidaan toteuttaa pilkkomalla monoliittijärjestelmä moneksi itsenäiseksi palveluksi. Ne voidaan toteuttaa myös aloittamalla tyhjästä, mutta se asettaa tietynlaisia haasteita. On helpompaa pilkkoa jotakin olemassa olevaa ja tunnistettavaa kuin jotain, mitä ei ole vielä olemassa. Samassa yhteydessä on tärkeää tunnistaa yrityksen toiminnot ja laittaa mikropalvelut heijastamaan niitä. (Newman 2015, 249.) Kun mikropalvelut heijastavat yrityksen toimintoja, kehittäjätkin ymmärtävät paremmin järjestelmän käyttäjien tarpeita.

Mikropalvelut voi suunnitella hyvin, mutta siinä voi helposti mennä myös hakoteille. Niiden suunnittelussa kannattaa kiinnittää huomiota ainakin palvelujen yhteen liittämisen tapoihin ja yksittäisen palvelun sisäisen määrittelyn yhdenmukaisuuteen.

Ensin mainittu tarkoittaa, että palvelut voidaan liittää yhteen hyvin väljästi tai tiukasti. Kun palvelut ovat väljästi yhteen liitettyjä, uuden muutoksen tekeminen yhteen palveluun ei tulisi vaatia muutosta toiseen sen seurauksena. Mikropalveluiden koko idea on uuden muutoksen tekeminen yhteen palveluun ja mahdollistaa helppo käyttöönotto ilman, että se vaatii muutosta järjestelmän muihin osiin. Kireää yhteen liittämistä kuuluu välttää. Klassinen virhe on valita integraatiotyyli, joka sitoo kireästi yhden palvelun toiseen. On myös hyvä rajoittaa erilaisten kutsujen määrää palveluiden välillä, koska tämäkin voi johtaa pahimmillaan kireään yhteen liittämiseen.

Palvelun yhdenmukaisuus tarkoittaa, että samankaltaiset funktionaalisuudet ovat toteutettu yhteen palveluun. Jos palvelun toimintoja haluaa muuttaa, sen pystyy tekemään yhdessä paikassa ja julkaisemaan tehdyn muutoksen nopeasti. Jos samankaltaisia toimintoja on toteutettu monessa palvelussa, ja siihen halutaan tehdä julkaistava muutos,

monta palvelua joudutaan pahimmillaan julkaisemaan samanaikaisesti muutoksen toimittamiseksi. Moneen paikkaan tehtävien muutoksien toteuttaminen on hitaampaa ja monen palvelun käyttöönotto kerralla on riskialttiimpaa. (Newman 2015, 30.)

Mikropalvelut on hyvä suunnitella liiketoimintakonseptien ympärille, jossa yksi palvelu toteuttaa jonkin tietyn liiketoimintaprosessin. Tällaisen rakenteen omaava toteutus on paljon vakaampi kuin mikropalvelut, jotka on rakennettu teknisten konseptien ympärille. (Newman 2015, 246.) Teknisten konseptien ympärille suunnitellut palvelut eivät ole aina väärä ratkaisu. Se voi olla hyödyllistä, kun organisaatio haluaa saavuttaa tiettyjä suorituskykytavoitteita. (Newman 2015, 37.) Mallintamalla toimialue, jossa järjestelmä operoi, muodostetaan paljon vakaampia käyttöliittymiä. Sen lisäksi muutokset yrityksen liiketoimintaprosesseissa on helpommin heijastettavissa. (Newman 2015, 246.) Muita mikropalveluiden suunnittelussa huomioitavia asioita ovat:

- saatavuus
- skaalautuvuus
- suorituskyky
- joustavuus. (Subramanian 2015.)

Saatavuuden kultaisena sääntönä pidetään epäonnistumisten ennakointia ja suunnittelua siten, että järjestelmä on saatavilla 99 prosenttia ajasta. Se tarkoittaa, että järjestelmä voi olla kaatunut vuodessa vain viisi ja puoli minuuttia. Klusterimallia käytetään tukemaan korkeaa saatavuutta. Mikropalveluiden on oltava skaalautuvissa horisontaalisesti ja vertikaalisesti. Eli toisin sanoen järjestelmän suorituskykyä pitäisi pystyä nostamaan ja laskemaan yhtä lailla. Horisontaalisesti skaalautuvalla mikropalvelulla voi olla monia instansseja, jotka nostavat järjestelmän suorituskykyä. Suorituskyvyn vaatimukset on oltava selvillä jo suunnittelun alkuvaiheella, jotta vältetään saman asian työstäminen uudelleen myöhemmässä vaiheessa huonon suunnittelun seurauksena. Suorituskykyä mitataan vasteajan eli kertoimen perusteella. Joustavuudella mitataan mikropalveluiden sopeutuvuutta muutoksille. Mikropalveluiden ekosysteemissä muutoksia tapahtuu nopeammin kuin muissa järjestelmissä, koska palveluita kehitetään ketterällä menetelmällä ja eri tiimit vastaavat omista palveluistaan. Mikropalvelut eivät välttämättä ole yhdessä toimivia, jos ne eivät sopeudu muiden palveluiden muutoksiin. Sitä varten on oltava kunnollinen mekanismi, jotta kyseisiä tilanteita vältetään. (Subramanian 2015.)

2.3 Hallinta

Mikropalvelut tuovat mukanaan monimutkaisuutta järjestelmän hallintaan. Hankaluutta tuovat erilliset palvelut, joista osa kommunikoi toisten palveluiden kanssa toteuttaakseen tehtävänsä. Tässä tapauksessa monoliittijärjestelmä on tavallaan virheystävällisempi, koska sen virheitä on helpompi paikantaa. Niitä on helpompi paikantaa, koska järjestelmässä kaikki on koottu yhteen ja virheillä on yksi sama paikka. Mikropalvelujärjestelmässä hallittavia palveluita on monia ja suuren moduulijoukon yhteistoiminta voi muuttua sotkuiseksi. Sen seurauksena on monia lokitiedostoja ja paikkoja, joissa verkon latenssi voi aiheuttaa ongelmia. Tämä hankaloittaa piilevän ongelman sijainnin paikantamista järjestelmässä. Ongelman syyn löytämistä voi hallita helpommin järjestelmän tarkalla seurannalla ja koosteilla, joista kokonaiskuva ilmenee. (Newman 2015, 155.) Hallintaa varten on toteutettu monia järjestelmiä, jotka hoitavat palvelujen monitorointia järkevästi. Palveluita voidaan monitoroida

- seuraamalla saapuvaa vasteaikaa ja virhemääriä
- yhdenmukaistamalla miten ja minne metriikoita kerätään
- keräämällä lokitietoja yhdenmukaiseen sijaintiin
- seuraamalla valvomattomia prosesseja monitoroimalla perustana olevaa käyttöjärjestelmää
- tekemällä kapasiteettisuunnitelmaa. (Newman 2015, 167.)

Metriikoita kerätään järjestelmän pitkän aikavälin käyttäytymisestä, koska sen perusteella voi päätellä poikkeavuudet paremmin. Näin tiedetään, koska järjestelmässä tapahtuu jotakin normaalista poikkeavaa, ja milloin siihen on syytä reagoida. (Newman 2015, 159.) Järjestelmää voi olla hyvä skaalata, jotta voidaan käsitellä sen sisältämiä kuormituksia ja virheitä paremmin. Seuraavat kohdat auttavat paremmin ymmärtämään, milloin skaalaamista on hyvä harkita, ja tarvitaanko sitä ollenkaan:

- vasteaika ja latenssi
- saatavuus
- tietojen kestävyys. (Newman 2015, 206-207.)

Vasteaikaa ja latenssia mittaamalla saadaan selville, kuinka kauan järjestelmän operaatioiden tulee viedä aikaa. Niitä voidaan mitata käyttäjien määrän perusteella eli sillä, miten käyttäjistä aiheutuva kuormitus vaikuttaa vasteaikaan. Tietoverkosta johtuvia poik-

keavia havaintoja ilmentyy aina. Siksi on hyödyllistä asettaa tavoitteet seurattavista vasteajoista ilmeneville prosenttipisteille. Tavoitteen tulisi sisältää myös oletetut rinnakkaisyhteyksien ja järjestelmän käyttäjien määrät. Newmanin (2015, 207) kirjan esimerkissä tavoiteltiin verkkosivustolle 90 prosenttipisteen kahden sekunnin vasteaikaa, kun käsiteltiin kaksisataa rinnakkaista yhteyttä sekunnissa. Järjestelmän saatavuutta voidaan mitata sen perusteella, kauanko sen häiriöajat kestävät. Häiriöajan mittaamisesta on enimmäkseen hyötyä vain silloin, kun on tarve tutkia järjestelmän historiaan kuuluuvia raportteja. Tietojen kestävyyteen liittyen otetaan huomioon esimerkiksi hyväksyttävä tietojen menetyksen määrä ja tietojen säilytysaika. Nämä muuttuvat aina tapauskohtaisesti. Viitaten Newmanin kirjassa olleeseen esimerkkiin, käyttäjän istuntolokeja voidaan säilyttää vuoden tai vähemmän tilan säästämiseksi. Poikkeuksena esimerkiksi rahoitustapahtumarekistereitä voi joutua säilyttämään useita vuosia. Tässä luvussa mainittuja kohtia on syytä mitata järjestelmällisesti suorituskykytesteillä. (Newman 2015, 207.) On siis erityisen tärkeää huomioida järjestelmän toiminnassa tapahtuvat poikkeavuudet ja miettiä sitä, millaista on järjestelmän normaali toiminta. Jos normaalissa toiminnassa tapahtuu jokin selkeä poikkeavuus, siihen pystytään reagoimaan ajoissa.

2.4 Mikropalvelun haitat

Vaikka mikropalvelut tuovat järjestelmän toteutukseen paljon mahdollisuuksia, mukana kulkevat myös kaikki hajautettuihin järjestelmiin lukeutuvat haitat. Erityistä huomiota haittoihin pitää kiinnittää jo mikropalveluiden suunnitteluvaiheessa. Kiinnittämällä huomiota niihin, voidaan mikropalveluista hyötyä todellisesti. (Newman 2015, 11.) Joitakin mikropalveluiden haittoja ovat:

- testaushaasteet
- turvallisuushaasteet
- kasvanut resurssien ja tietoverkon käyttö
- järjestelmän monimutkaisuus. (Akshay 2018.)

Kaikki yllä mainitut haitat juontavat juurensa erillisistä palveluista. Testaamisesta tulee haasteellisempaa virheiden osalta, ja se kestää kauemmin. Viivästymiset johtuvatkin usein palveluiden testaamisesta. Kuten kaikissa muissakin järjestelmissä, myös mikropalveluilla toteutetussa järjestelmässä on erityisen tärkeää kiinnittää huomiota sen turvallisuuteen. Turvallisuuteen liittyviä haasteita ovat koko järjestelmälle määriteltävät todentamis- ja valtuutusmääritykset, joita jokaisen erillisen palvelun tulee noudattaa. Sen

lisäksi keskenään kommunikoivien palveluiden datavirtaa on todella hankalaa seurata. (Akshay 2018.)

Monen yhdessä toimivan palvelun on tärkeää kommunikoida keskenään joissain määrin. Tällaisesta kommunikoinnista aiheutuu tietoverkon käytön lisääntymistä, joka edellyttää nopeaa ja luotettavaa verkkoyhteyttä. Kustannukset kasvavat, jotta järjestelmiä voidaan suorittaa. Kaikki mikropalvelut tarvitsevat usein myös oman teknologiapaketin, joka tukee niiden tarvetta. Teknologiapaketilla tarkoitetaan palvelun perustana olevia elementtejä. Näitä voivat olla esimerkiksi viitekehykset ja ohjelmointikielet, joiden päälle kaikki muu rakennetaan. Jokaisella palvelulla on omansa, mikä johtaa monimutkaisuuteen, koska uudempien ominaisuuksien hallintaan ja rakentamiseen vaadittavat resurssit ja taidot ovat erittäin korkealla vaatimuslistalla (Akshay 2018). Kuten aiemmin tässä luvussa todettiin, erilliset palvelut lisäävät haasteita mikropalvelujen hallintaan ja monitorointiin eli järjestelmän virheiden paikantaminen hankaloituu. Hallinnan ja monitoroinnin perusteellisen suunnittelun lisäksi pitää tunnistaa yrityksen toteuttamat selkeät toimialueet, johon toteutettavat mikropalvelut liittyvät. Näin vältytään turhilta ja ylimääräisiltä mikropalveluilta, jotka olisi voinut toteuttaa yhtenä palveluna. Samaan kontekstiin liittyvät toiminnollisuudet olisi hyvä koota yhteen palveluun. Toisin sanoen, mitä vähemmän on palveluita, sitä vähemmän on hallintaan ja monimutkaisuuteen liittyviä haittoja.

3 KOMPASSI-HANKE

Tässä luvussa esitellään Varsinais-Suomen KomPAssi-hanke sekä sen sisältö, mihin opinnäytetyössä käsitelty mikropalvelu liittyy. Läpi käydään myös hankkeen sekä siihen kuuluvan ohjelmistoprojektin laajuus ja haasteet. Sosiaali- ja terveysministeriön kärkihankkeeseen perustuva KomPAssi on Varsinais-Suomen keskitetty asiakas- ja palvelunohjaushanke, jonka tavoite on kehittää ikäihmisten kotihoitoa ja vahvistaa kaikenikäisten omaishoitoa. Toteutettu mikropalvelu tukee osaltaan kyseisiä hankkeen tavoitteita eli auttaa ikäihmisiä löytämään sopivia kotihoidon palveluita verkon kautta. Hankkeen toteuttamiseen osallistui Varsinais-Suomen kaikki 27 kuntaa, joista hankevastaavana toimi Turku. Monet järjestöt maakunnan alueelta kuten Suomen Punainen Risti ja Perhehoitoliitto liittyivät mukaan tuomaan oman panoksensa hankkeeseen asiantuntijoiden näkökulmasta. Lisäksi hankkeen edellyttämän arviointitutkimuksen toteutti Turun ammattikorkeakoulu jokaiselta tavoitteen osalta. (Kompassi 2017, 4.) KomPAssi tarjoaa verkkopalvelun, johon palveluntuottajat voivat kirjata palvelunsa samassa muodossa, jolloin ne ovat vertailukelpoisia loppukäyttäjälle. Tämä mahdollistaa maakunnallisten yhtenäisten palveluiden kilpailuttamisen. Verkkopalvelua kehitettiin erillisessä ohjelmistoprojektissa, mihin kuului myös mikropalvelun toteutus. Projektissa tehtiin yhteistyötä eri toimittajien kesken. Tavoitteiden toteutuminen KomPAssi-hankkeessa tuki maakunnan yhteisen palvelujärjestelmän muodostumista ja valtakunnallisen Sote-hankkeen etenemistä (Kompassi 2017, 2).

3.1 Hankkeen taustaa

Varsinais-Suomen maakunnan ominaispiirteisiin kuuluu kuntien suuri määrä ja hajanainen kuntarakenne. Tämä ja pitkät välimatkat tuovat omat haasteensa palvelujen järjestämiselle, koska palvelut ovat alueellisesti hajallaan ja palvelutarjonta vaihtelevaa. Maakunnan haasteena palvelujen tarjonnassa on myös sen monimuotoisuus. Monimuotoisuudella tarkoitetaan kaupunki-, maaseutu- ja saaristoasutusta, ruotsinkielisyyden painottumista osassa kuntia ja kuntien koon vaihtelu. Maakunnan monimuotoisuus huomioon ottaen kartoitetaan ja yhtenäistetään kuntien nykyiset palveluiden kriteerit sekä palveluprosessit. (Kompassi 2017, 2.) Verkkopalvelulla, ja siihen kuuluvalla mikropalvelulla tavoitellaan edellä mainittujen haasteiden ratkomista.

KomPAssi - keskitetty asiakas- ja palvelunohjaustoimintamalli kohdentuu kahteen eri I&O-hankkeen toimintamalliin:

1. ikäihmisten palvelujen toimintamallit
2. kaikenikäisten omaishoidon vahvistamisen toimintamallit.

Hankkeessa kehitetään palvelunohjausta tekemällä asiakkaalle eli ikäihmiselle palvelutarpeen arviointi, jonka jälkeen saa neuvoa, apua, tukea ja palveluita keskitetystä asiakas- ja palvelunohjauksesta. Palvelutarve arvioidaan validoitujen mittarien avulla, kuten RAI-arviointijärjestelmällä, ja sen pystyy tekemään KomPAssin tarjoamassa verkkopalvelussa. Tukien ja palveluiden esille tuominen, palveluiden myöntämisen edellytyksien yhdenmukaisuus sekä myönnetyn palvelun toteutuminen ovat palvelunohjauksen kehittämisen tavoite. (Kompassi 2017, 2-9.) Keskitetyllä toimintamallilla helpotetaan arjen vaivattomuutta palveluita saaville ja hakeville sekä kohdennetaan palveluita oikea-aikaisesti ja järjestelmällisesti ikäihmisten arvioitujen palvelutarpeiden mukaisesti. Arvioinnin perusteella pystytään kohdentamaan palveluita ja palvelukokonaisuuksia erilaisille asiakasryhmille, kuten muisti- ja monisairaille ja omaishoidon asiakkaille asianmukaisesti. (Kompassi 2017, 9.)

Opinnäytetyössä käsitellyn mikropalvelun suunnittelussa ja toteutuksessa tehtiin ratkaisu käyttää tunnisteita, jotta ikäihmiset voivat niiden avulla lajitella heille tarjottuja palveluita. Ne mahdollistavat tietynlaisen palvelun löytymisen helpommin verkkopalvelussa. Esimerkiksi ruokapalveluita voi lajitella allergeenien mukaan, jolloin verkkopalvelu näyttää tietyssä tapauksessa vain gluteenittomia ruokapalveluita tarjoavat palveluntarjoajat.

3.2 Laajuus

Maakunnallisena yhteistyönä toteutettavan hankkeen ensimmäiseen vaiheeseen kuuluu kuntien toiminta- ja päätöksentekomallien kartoittaminen. Tavoitteena on laatia yhdenmukainen käytäntö ja prosessi omaishoidon toimintamalleihin sekä keskitettyyn palveluja asiakasohjaukseen. Kunnissa on laadittava päätökset erikseen yhdenmukaiseen toimintamalliin sitoutumisesta. (Kompassi 2017, 23.) Nykytilanteen kartoittamisen jälkeen luodaan yhteiset toimintamallit projektin eri tavoitteisiin. Toimintamallit ovat seuraavat:

- asiakaslähtöisten maakunnallisten ja tasalaatuisten ohjaus- ja neuvontapalvelujen järjestäminen sähköisen alustan avulla

- tarpeita vastaavien palveluiden saaminen tasa-arvoisesti ja oikea-aikaisesti. (Kompassi 2017, 23.)

Toiseen kokonaisuuteen liittyy omaishoito. Siinä tarjottuja palveluita monimuotoistetaan tukemaan omaishoitajien jaksamista ja laaditaan malli ohjaamaan omaishoidon äkillisissä kriisitilanteissa. (Kompassi 2017, 23.)

Hankkeen toteuttamiseen ovat osallistuneet

- hankejohtaja
- hankesihteeri
- prosessikoordinaattorit
- hankkeen seurantaryhmä
- IT-projektipäällikkö
- projektityöntekijä
- projektisuunnittelija.

Hanke on alkanut lokakuussa 2016 ja valmistunut lokakuun 2018 lopussa.

3.3 Haasteet

Varsinais-Suomen KomPAssi-hankkeessa on toteutettu riskienarviointi. Niiden tunnistamista on jatkettu säännöllisesti koko hankkeen ajan projekti- ja ohjausryhmien toimesta. Sen sisäisiä riskejä on

- osallistuvien kuntien sitoutuminen tavoitteisiin
- toimintamalleihin liittyvät näkemyserot kuntien kesken
- hankkeen aikataulu ja sisällöllinen laajuus
- hankkeen sisäisen organisoinnin hajaantuminen
- uuden teknologian käyttöönottamiseen liittyvät riskit
- hankkeen avainresurssien kuormitus
- kuntien omien riskinhallintakeinojen vajavaisuus. (Kompassi 2017, 27.)

Hankkeen ulkoiset riskit on

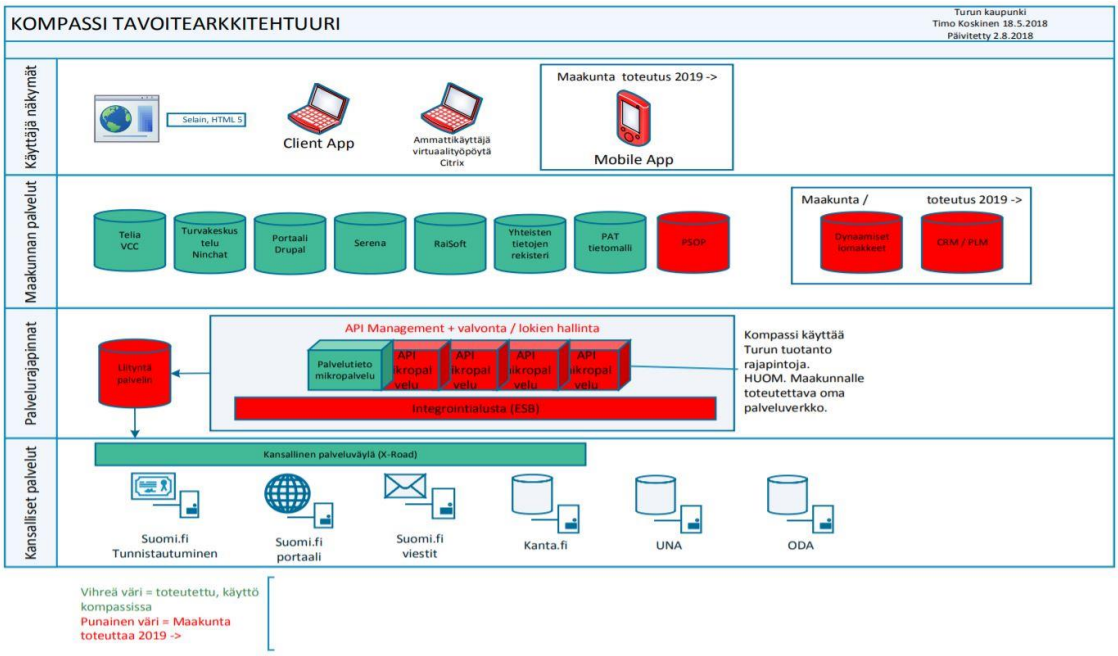
- ulkoisen toimintaympäristön muutokset

- kuntien muiden toimialojen kehittäminen ja sen vaikutus hankkeeseen. (Kompassi 2017, 27.)

Hankkeeseen kuuluvaan ohjelmistoprojektiin liittyy erinäiset haasteet. Myös siihen sisältyy omat aikataululliset ja tekniset riskit. Tässä projektissa erityisesti teknisen sisällön muuttuminen asettaa tiettyjä haasteita, ja siten lisää samalla aikataulullisia haasteita saattaa projektissa kaikki toivotut kriteerit päätökseen ajoissa. Aikataulullisiin riskeihin kuuluvat erityisesti kaikkien tehtävien mukaan ottaminen, toimitusaikojen huomioon ottaminen sekä työmääräarviot, ja niiden laatiminen. Projektissa voi ilmetä myös tarpeellisia lisä- ja muutostöitä, jotka vaativat neuvotteluja. Joskus tuotekehitykseen kuuluva lisenssien saanti voi olla haaste ja viedä aikaa. Mikropalvelu sopii toteutusratkaisuna hankkeeseen, koska sen idea on tukea helppoa käyttöönottoa ja mukautua muutoksiin nopealla aikataululla.

4 MIKROPALVELUN TOTEUTUS

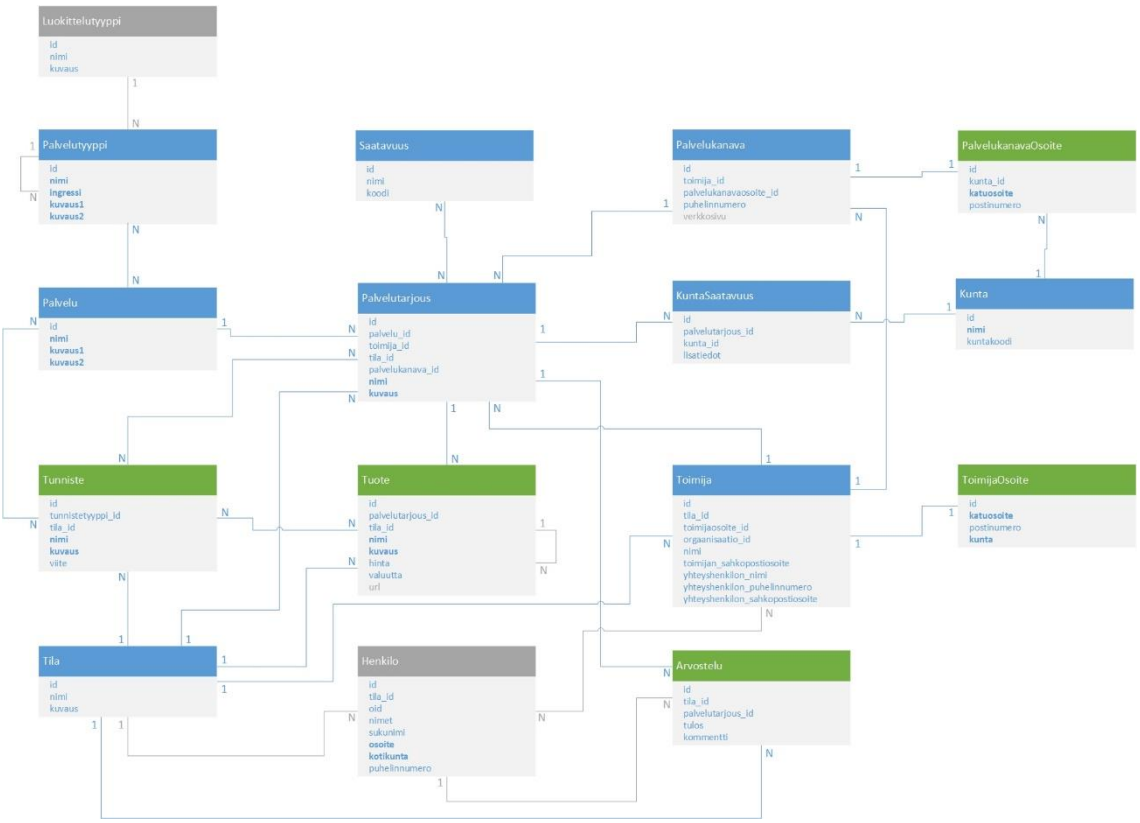
Tässä luvussa kuvataan KomPAssi-hankkeeseen liittyvän ohjelmistoprojektin teknistä toteutusprosessia. Projektin arkkitehtuurisena tavoitteena oli kehittää mikropalveluperiaatteiden mukaisesti eriytettävä komponentti, joka ei ylläpidä tilaa tai välimuistia. Tekninen toteutusprosessi käsittelee mikropalvelua nimeltä palvelutieto, joka koostuu KomPAssin tarjoaman verkkopalvelun asiakasnäkymän taustapalveluista. Kompassin tavoitearkkitehtuurikuvasta ilmenee sen kokonaisuus, palvelutieto-mikropalvelun asema ja muut komponentit (Kuva 1). Luvun lopussa perehdytään tarkemmin yhteen taustapalveluun, arvostelu-rajapintaan.



4.1 Mikropalvelun tietomalli

Mikropalvelussa käytetään hyödyksi rajattua joukkoa käsitteitä PAT-ydintietomallista. Kehitin käsitteistä tietomallikuvan verkkopalvelun asiakasnäkymän tietokantatoteutuksen pohjaksi (Kuva 2). Tietomalli kuvaa mikropalveluun liittyvät käsitteet, ja niiden väliset relaatiot tietokantatasolla. Koska kaikki abstraktiot eivät ole sopineet sellaisenaan, mallia on laajennettu hienojakoisemmalla tasolla. Siniset taulut ovat hyödynnetty PAT-ydintietomallista, vihreät ovat itse lisättyjä ja harmaat ovat poistuvia, mutta mahdollisesti

myöhemmin lisättävissä olevia tietokantatauluja. N:n suhde N:n relaatio kuvastaa monen suhdetta moneen, kun taas yhden relaatiot kuvastavat yhden suhdetta yhteen. Esimerkiksi monella palvelutyypillä voi olla monta palvelua ja toisinpäin tai yhdellä toimijalla voi olla yksi toimijaan liittyvä osoite ja toisinpäin. Taulujen sisällä on kenttiä, kuten id ja nimi. Id tarkoittaa tietokantataulun rivin identifioivaa teknistä tietokanta id:tä eli tunnistekenttää. Nimi tarkoittaa esimerkiksi jonkin palvelutyypin nimeä, joka ilmenee verkkosivulla tekstitietona käyttäjälle. Mikropalvelu keskustelee tietokannan kanssa, josta se noutaa verkkopalvelun vaatimia tietoja, kuten tietomallissa ilmenevät palvelutarjoukset ja arvostelut.



Kuva 2. Mikropalvelun tietokannan looginen toteutus.

Jotta lukija saa paremman käsityksen siitä, mitä verkkopalvelun taustalla olevassa tietomallikuvassa tapahtuu, kuvaan tämän luvun teknisen toteutuksen kannalta olennaisimmat käsitteet ja etenemisen ylätasolla käyttäjän näkökulmasta.

Verkkopalvelun etusivulta ilmenee käyttäjälle lista erilaisista palvelutyypeistä, joita sivusto tarjoaa. Nimensä mukaisesti palvelutyyppi kuvaa palvelun tyyppiä, kuten ruoka- ja

ravitsemuspalvelua tai siivouspalvelua. Keskitymme tarkemmin tässä läpikäynnissä palvelutyyppiin ruoka ja ravitsemus. Käyttäjän valitessa palvelutyypin, sen alta käyttäjä voi valita siihen liittyvän tarkemman palvelun kuvauksen, kuten ateria-, kauppa- tai ravintolapalvelun. Käyttäjä voi myös siirtyä selaamaan palvelutarjouksia. Palvelutarjous tarkoittaa toimijan eli palveluntarjoajan tarjoamaa tiettyä ateria-, kauppa- tai ravintolapalvelua, joita käyttäjä voi itselleen tilata. Palvelutarjous voi sisältää tuotteita. Esimerkiksi Askon Ateriat tarjoaa Askon ateriapalvelua, jonka tuotteita ovat lämmin lounas ja keittolounas. Käyttäjä voi antaa palvelutarjousta koskevan arvostelun saamastaan palvelusta.

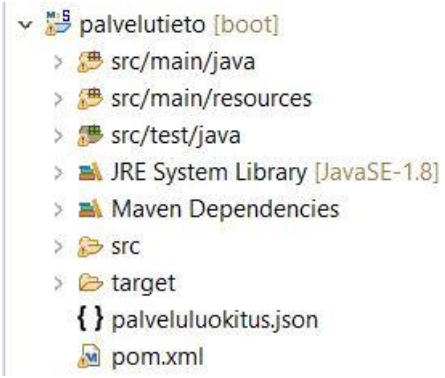
4.2 Mikropalvelun määrittely ja rakenne

Mikropalvelun REST-rajapinnat tuli kuvata OpenAPI 3 määrittelyn mukaisesti. Automaattisesti generoitava dokumentaatio on helppo päivittää rajapinnan päivittyessä. Yleiseen toimintatapaan kuuluu:

- onnistunut pyyntö palauttaa HTTP-status 2xx
- epäonnistunut pyyntö palauttaa HTTP-status 4xx tai 5xx
- sanomien tietosisältö on JSON-muodossa ja perustuu tietomalliin.

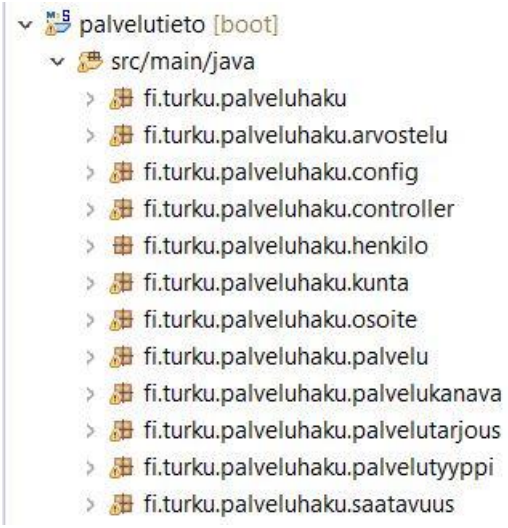
Rajapinnat ovat toteutettu CRUD-toimintaperiaatteella eli rajapinnoissa on mahdollista luoda, lukea, muokata ja poistaa tietoa. Kehitystyötä tehtiin käyttäen Java 8 versiota ja Spring Bootia. Spring Boot on kevyt ajoalustan tarjoava sovelluskehys mikropalvelun logiikalle. Sen avulla voidaan eriyttää koodi ja ajoalusta toisistaan helposti, eikä siihen muodosteta tiukkoja riippuvuuksia.

Mikropalvelun rakenne luotiin Maven-projektina kehitysympäristössä (Kuva 3). Projektin lähdekoodit ovat main/java-kansiossa, resurssit ovat main/resources-kansiossa ja testauskoodit ovat test/java-kansiossa. Resurssikansion alla on application.properties-tiedosto, joka sisältää applikaation oletusasetukset. Pom.xml on konfiguraatiotiedosto, joka sisältää valtaosan projektin rakentamiseen vaadittavasta datasta. Pom muodostuu sanoista project, object ja model eli nimensä mukaisesti se myös määrittelee projektin mallin.



Kuva 3. Mikropalvelun projektirakenne kehitysympäristössä.

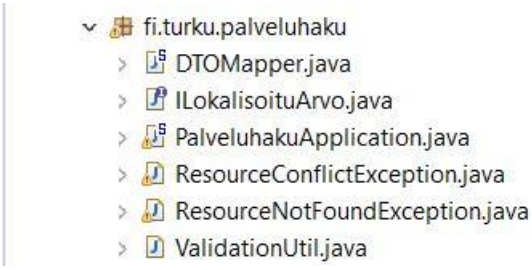
Projektirakenteen main/java-kansion alla on paketteja, joiden sisälle rajapintoihin liittyvät eri lähdekoodit ovat asetettu (Kuva 4). Kuvasta ilmenee paketit, jotka sisältävät applikaation varsinaisen logiikan, asetustiedostot sekä osan rajapintapaketeista. Samaan aiheeseen liittyvät tiedostot ovat lisätty loogisesti ja helposti löydettävästi omiin paketteihinsa.



Kuva 4. Mikropalveluun liittyvät paketit, jotka sisältävät lähdekoodit.

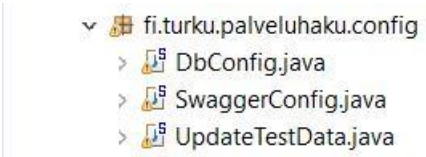
Palveluhaku-paketti sisältää varsinaiseen applikaatiologiikkaan liittyvät tiedostot (Kuva 5). Logiikka on jaettu eri tiedostoihin, jolloin niillä on oma selkeä tehtävänsä. Se tekee tietyn tiedoston paikantamisesta paljon helpompaa. DTO muodostuu sanoista data, transfer ja object. DTOMapper.java muuttaa tietomalliobjektin applikaatioon soveltuvaan muotoon. PalveluhakuApplication.java sisältää applikaatioluokan, joka käynnistää applikaation sen sisältämällä metodilla. ResourceConflictException.java ja ResourceNot-

FoundException.java tiedostoilla hallitaan mahdolliset poikkeukset. Ne poimivat määrätyllä tilalla palaavat vastaukset. Näin saadaan näytettyä ennalta määritetty virheviesti käyttöliittymällä tai tarvittaessa kirjoitettua ylös poikkeustilanne. ValidationUtil.java tiedostoa käytetään objektien validointiin.



Kuva 5. Paketti, joka sisältää varsinaisen applikaatiologiikan.

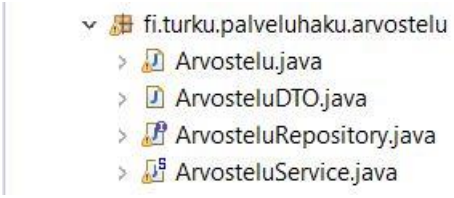
Config-paketin sisälle on sijoitettu asetustiedostoja ja yksi testidatan käsittelyyn liittyvä tiedosto (Kuva 6). DbConfig.java sisältää applikaation tietokantaan liittyvät konfiguraatiot ja SwaggerConfig.java sisältää Swagger-pystytyksen, joka generoi interaktiivisen dokumentaation REST-ohjelmointirajapinnoille. Sen avulla voi tehdä testejä applikaatioon liittyvien HTTP-metodien kanssa. UpdateTestData.java on testitiedosto, jossa päivitetään testidataa testitietokantaan.



Kuva 6. Paketti, joka sisältää asetustiedostoja.

4.3 Mikropalvelun arvostelu-rajapinta

Mikropalvelun arvostelu-paketti sisältää arvostelu-rajapintaan liittyvät tiedostot (Kuva 7). Siihen liittyy myös ArvosteluController.java-tiedosto, joka on sijoitettu controller-paketin sisälle. Arvostelu-rajapinnan HTTP-metodit, GET ja POST, hakevat ja tallentavat tietoa. GET-metodilla haetaan toimijaan ja palvelutarjoukseen liittyvät arvostelut verkkopalvelussa ja POST-metodilla tallennetaan arvostelu yksittäisille palvelutarjouksille. Näistä koostuu arvosteluun liittyvä kirjoitusrajapinta, ja ne kuuluvat REST(ful) ohjelmointirajapintaan. Arvostelu-rajapinta toteuttaa CRUD-periaatteesta vain osan, tiedon luomisen ja lukemisen.

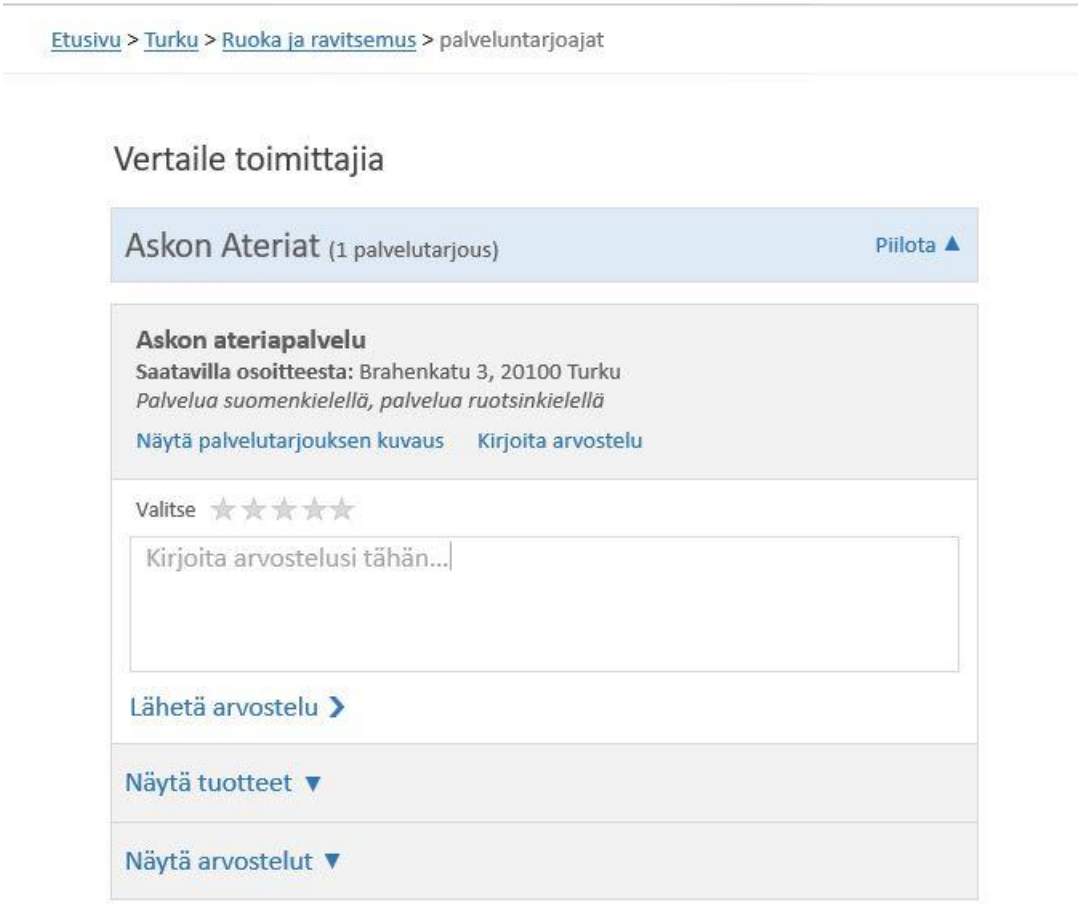


Kuva 7. Paketti, joka sisältää arvostelu-rajapintatiedostot.

ArvosteluController.java on ohjainluokka, joka käsittelee GET- ja POST-pyynnöt arvostelulle ja toimii rajapinnan kuvauksena. Luokassa on myös @Autowired-annotaatio, joka alustaa esimerkiksi ArvosteluService-luokan. Alustuksella ohjainluokka voi kutsua ArvosteluService luokan sisältämiä metodeita. Arvostelu.java kuvaa arvostelu-taulun ja linkittää siihen liittyvät toiset taulut, palvelutarjous- ja tila-taulun. Tietokannasta luettu data luetaan kyseisiin tauluihin. ArvosteluDTO.java on luokka, minkä ohjainluokka ottaa vastaan. Se määrittelee sen, missä muodossa arvosteluun liittyvä data siirretään. ArvosteluRepository.java tekee kyselyt tietokantaan, josta se etsii arvostelut toimijalle ja palvelutarjoukselle. ArvosteluService.java sisältää applikaation liiketoimintalogiikan.

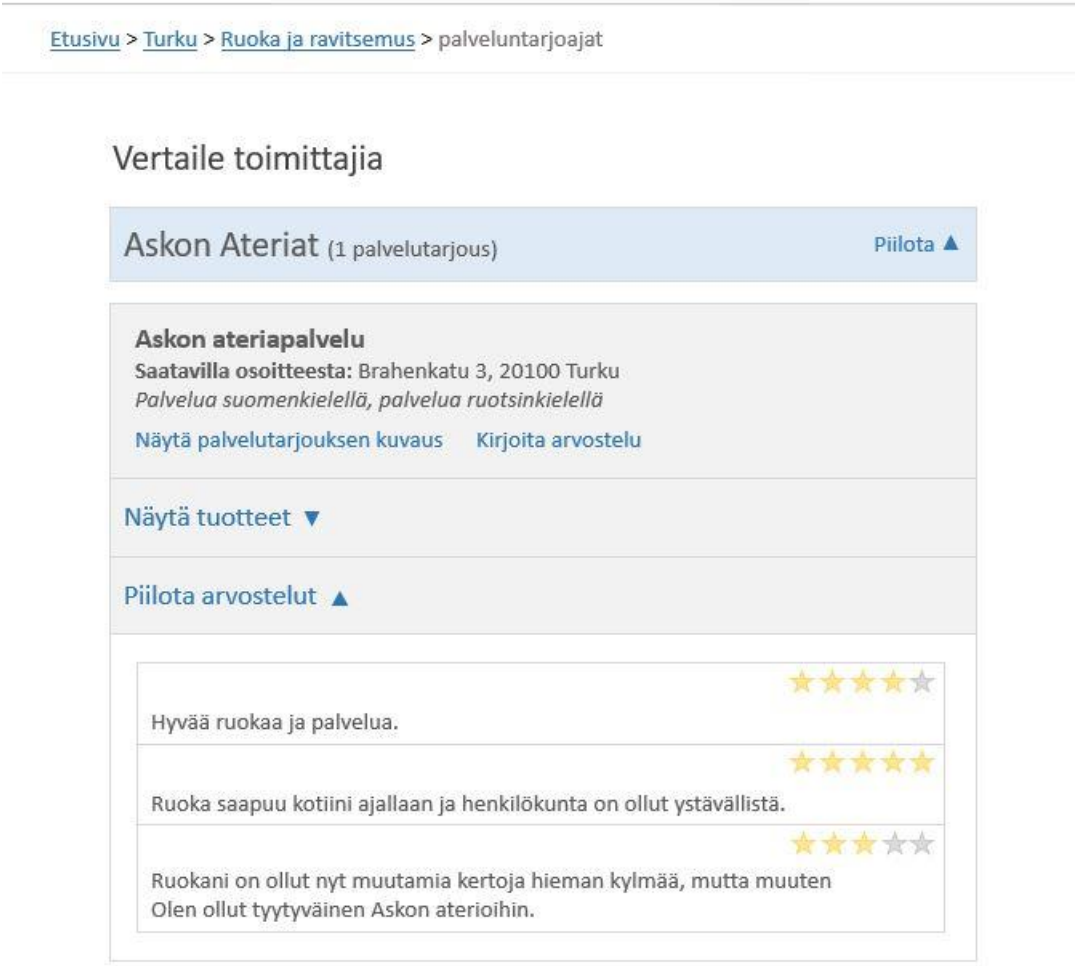
4.4 Arvostelu-rajapinnan käyttöliittymä loppukäyttäjälle

Drupal on sivuston esityskerros, joka toimii verkkopalvelun käyttöliittymänä käyttäjälle. Se sisältää dynaamiset näkymät mikropalvelun tarjoamalle tiedolle. Palvelutarjoukseen liittyvää arvostelu-rajapintaa ei hyödynnetty ensimmäisessä tuotantoversiossa, joten loin havaintokuvat siitä, miten arvostelun lisääminen ja näyttäminen voisi ilmentyä verkkopalvelussa. Arvostelu voitaisiin lisätä toimittajien vertailunäkymässä painamalla ”Kirjoita arvostelu”-tekstiä (Kuva 8). Sen jälkeen ylimmän harmaan laatikon alapuolelle avautuu toinen laatikko, jossa ohjeistetaan käyttäjää valitsemaan tulos käyttäen tähtikuvia sekä kirjoittamaan haluamansa arvostelu. Arvostelu lähetetään anonyyminä painamalla ”Lähetä arvostelu”-tekstiä.



Kuva 8. Havaintokuva arvostelun lisäämisestä verkkopalvelussa.

Käyttäjä voi nähdä toimijakohtaiset palvelutarjouksen arvostelut painamalla alimman harmaan laatikon ”Näytä arvostelut”-tekstiä. Teksti muuttuu painamisen jälkeen ”Piilota arvostelu”-tekstiksi (Kuva 9). Harmaan laatikon alapuolelle avautuu palvelutarjouksen kaikki arvostelut, joita käyttäjät ovat antaneet. Saatu tulos ilmentyy tähtikuvista, joissa värilliset tähdet kuvaavat käyttäjien antamaa tulosta.



Kuva 9. Havaintokuva arvostelujen näyttämisestä verkkopalvelussa.

5 POHDINTA

Opinnäytetyön viimeisessä pohdintaluvussa kuvaan työn tavoitteita, toteutumista ja tuloksia. Kerron myös työn aikana heränneistä omista havainnoistani ja vastaan tulleista haasteista mikropalvelun toteutuksen aikana. Opinnäytetyön tavoite oli kasvattaa käsitystä mikropalveluista ja kuvata yhden mikropalvelun toteutusta käytännönläheisesti ohjelmistoprojektissa. Mikropalvelu kehitettiin tukemaan ikäihmisten neuvontaa ja ohjausta verkkopalvelussa, jonka kautta ikäihminen saa hankittua apua arjen haasteisiin kodistaan käsin vaivattomasti itse tai valtuutetun avulla. Tavoitteissa onnistuttiin siten, että tuloksena on ymmärrettävä kuvaus mikropalvelun toteutuksesta, ja sen sisältämän arvostelu-rajapinnan toiminnallisuudesta. Tavoitteista jäi uupumaan rajapinnan todellinen näkymä verkkopalvelussa, koska se julkaistaan myöhemmässä tuotantoversiossa. Vaikka todellista käyttöliittymäkuvaa mikropalvelun arvostelu-rajapinnalle ei saatu, toteutin sille havaintokuvat ja kuvasin toiminnallisuuden. Opinnäytetyön tuloksia voidaan hyödyntää silloin, kun punnitaan mikropalveluita vaihtoehtona järjestelmän toteuttamiselle.

Opinnäytetyön aikana pohdin mikropalvelun koon käsitettä ja palvelun granulariteetin eli raekoon määrittelyä, joka on todellisuudessa haastavaa. Työn pohjalta tehtyjen havaintojeni perusteella mikropalvelua voidaan pitää liian suurena silloin, kun se toteuttaa useamman liiketoimintaprosessin. Yhden funktion toteuttava mikropalvelu olisi selkeästi liian pieni ja epäkäytännöllinen, jolloin puhuttaisiin mikropalvelun sijaan nanopalvelusta. Tällainen toteutus lisäisi myös suuresti mikropalveluun liittyviä hallinnollisia haittoja, koska palveluiden määrä voisi kasvaa valtavaksi. Mielestäni mikropalvelun kokoa ei voida mitata sen fyysisen koon perusteella, vaan sen sisältämien toiminnollisuuksien määrän mukaan. Opinnäytetyössä käsitellyn palvelutieto-mikropalvelun toiminnallisuus on rajattu siten, että se toteuttaa CRUD-rajapinnat tietokantaan. Se kasvoi projektissa tapahtuneiden määrittelyjen myötä melko suureksi, mikä sai minut pohtimaan sen olemista palveluna mikropalvelun sijaan. Sitä voidaan mielestäni pitää mikropalveluna, koska sitä ei voida käyttää sellaisenaan, ja liitetään siitä syystä aina johonkin suurempaan kokonaisuuteen. Se voidaan myös mikropalvelun tavoin eristää muusta kokonaisuudesta siitä riippumatta. Palveluna sen taas tulisi toteuttaa jokin end-to-end lisäarvon tuottava liiketoimintaprosessi. End-to-end lisäarvolla tarkoitetaan valmista toiminnallista ratkaisua, joka ei vaadi kolmatta osapuolta. Opinnäytetyössä kuvattuja mikropalveluihin liittyviä hallinnollisia haasteita ja kompleksisuutta ei tämän toteutuksen yhteydessä tullut ilmi, koska keskityttiin vain mikropalvelun toteutukseen.

Suurimmat opinnäytetyön aikana ilmenneet haasteet olivat oman aikani allokointi töiden, opinnäytetyön ja henkilökohtaisen vapaa-aikani välillä. Lisäksi koin haastetta kuvata selkeästi mikropalvelun rajapintojen ja tiedostojen välisiä riippuvuuksia, koska niitä on paljon. Mikropalvelun toteutukseen liittyvää lukua ei myöskään voinut aloittaa ennen kuin kaikki ohjelmistoprojektin määrittelyt olivat saatettu loppuun. Jos sen kirjoittamisen olisi aloittanut ennen määrittelyiden lukitusta, lukua olisi pitänyt muuttaa moneen kertaan. Määrittelyt elivät koko ohjelmistoprojektin ajan ja muuntautuivat useasti. Tietomalliin liittyvät muutokset aiheuttivat aina muutoksia mikropalvelun lähdekoodissa ja rajapinnoissa. Mikropalvelun arvostelu-rajapinta päätettiin olla julkaisematta ensimmäisessä tuotantoversiossa, koska siitä olisi koitunut liikaa aikataulullisia haasteita projektille. Mikropalveluiden teorian kirjoittamisessa ei tullut vastaan suuria haasteita, koska materiaalia oli saatavilla paljon ja asia oli ymmärrettävää.

LÄHTEET

Akshay, P. 2018. 9 Fundamentals To A Successful Microservice Design. Viitattu 17.3.2019 https://www.lambdatest.com/blog/9-fundamentals-to-a-successful-microservice-design/.

Copter Labs 2019. JSON: Whats it is, how it works & how to use it. Viitattu 21.3.2019 https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/.

Drive Turku 2015. PAT-tietomalli digitaalisen asioinnin selkärankana. Viitattu 11.11.2018 https://blogit.turku.fi/driveturku/2015/11/11/pat-tietomalli-digitaalisen-asioinnin-selkarankana-malli-julkaistaan-avoimena-datana-23-11-2015/.

Everard, D. 2017. What is an API gateway. Viitattu 17.3.2019 https://www.quora.com/What-is-an-API-gateway.

Google Trends 2019. Mikropalveluja koskevat hakumäärät. Viitattu 6.3.2019 https://www.google.com/trends.

Järvenpää, L. 2017. Mikropalvelut ratkaisuna digitalisaation haasteisiin. Viitattu 1.11.2018 https://lamia.fi/blog/mikropalvelut-ratkaisuna-digitalisaation-haasteisiin.

KomPAssi 2017. KomPAssi Varsinais-Suomen keskitetty asiakas- ja palvelunohjaushanke. Viitattu 3.11.2018 https://kimpassa-allihopa.fi/sote/varsinais-suomen-maakunnallisia-sote-hankkeita/kompassi-varsinais-suomen-keskitetty-asiakas-ja-palveluohjaushanke/materiaalipankki.

Koskinen, T. 2018. Kompassin tavoitearkkitehtuuri. Esityksessä Ritvanen, J. Kuntien, maakuntien ja muiden toimijoiden palveluketjujen ja asiakasohjauksen hallinta. Viitattu 1.4.2019 https://www.kuntaliitto.fi/sites/default/files/media/file/03_maakuntafoorumi%20esitys%20161018.pdf.

Newman, S. 2015. Building Microservices. 1., ensimmäinen painos. Sebastopol: O'Reilly Media, Inc.

Ojala, H. 2019. Mikropalvelut. Viitattu 5.5.2019 https://www.dianti.fi/ords/f?p=118:15.

Rouse, M. 2006. HTTP (Hypertext Transfer Protocol). Viitattu 22.3.2019 https://searchwindevelopment.techtarget.com/definition/HTTP.

Rouse, M. 2019. RESTful API. Viitattu 21.3.2019 https://searchmicroservices.techtarget.com/definition/RESTful-API.

Saarelainen, A. 2016. Mikropalvelut korvaavat it-möhkäleet. Viitattu 11.11.2018 https://www.tivi.fi/Kaikki_uutiset/mikropalvelut-korvaavat-it-mohkaleet-6588283.

SmartBear Software Inc. 2019a. OpenAPI Specification. Viitattu 21.3.2019 https://swagger.io/specification/.

SmartBear Software Inc. 2019b. What is Swagger? Viitattu 24.3.2019 https://swagger.io/docs/specification/2-0/what-is-swagger/.

Subramanian, S. 2015. Microservices Design Principles. Viitattu 15.3.2019 https://dzone.com/articles/microservices-design-principles.

Suomen Drupal-yhdistys 2014. Mikä on Drupal. Viitattu 11.11.2018 http://www.drupal.fi/.

Terveyden ja hyvinvoinnin laitos 2017. Tietoa RAI-järjestelmästä. Viitattu 5.11.2018 https://thl.fi/fi/web/ikaantyminen/rai-vertailukehittaminen/tietoa-rai-jarjestelmasta.

The Apache Software Foundation 2019. What is maven? Viitattu 18.4.2019 https://maven.apache.org/what-is-maven.html.

Tirkkonen, J. 2018. Mikropalvelut selkokielellä – kenelle ne sopivat? Viitattu 5.5.2019 https://www.alfame.com/blog/mikropalvelut-selkokielella-kenelle-ne-sopivat.

Toivanen, A. 2018. Mini- vai mikropalvelu. Viitattu 11.11.2018 http://blogi.hiqfinland.fi/mini-vai-mikropalvelu.

Watts, S. 2018. REST vs. CRUD: What's the difference. Viitattu 12.11.2018 https://www.bmc.com/blogs/rest-vs-crud-whats-the-difference/.