JÄSENVERKON JA OLDTIMERTIMERIN INTEGRAATION SUUNNITTELU

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, Syksy 2016

Saara Karilaakso

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Visamäki
Tietojenkäsittely
Systeemityö

TIIVISTELMÄ

Tekijä Saara Karilaakso **Vuosi** 2016

Työn nimi Jäsenverkon ja OldTimerTimer integraation suunnittelu

TIIVISTELMÄ

Opinnäytetyön toimeksiantajana toimi Yoso Oy. Sen aikana suunniteltiin neljä erilaista integraatiosuunnitelmaa, kuinka OldTimerTimer ja Jäsenverkko voitaisiin integroida.

Työn tavoitteena oli saada aikaan suunnitelma, jonka perusteella OldTimerTimeria ja Jäsenverkkoa voitaisiin lähteä integroimaan. Lisäksi yritettiin arvioida mahdollinen työmäärä itse integraation toteutukselle.

Suunnitelmat toteutettiin tutustumalla neljään eri integraatiotapaan ja erilaisiin integraatioon liittyviin asioihin. Suunnitelmia pohdittiin tutkimalla toisia opinnäytetöitä, etsien mahdollisia integraatioideoita. Suunnittelun aikana tutustuttiin ja otettiin kantaa myös synkronisuuteen, reaaliaikaan ja tiedostomuotoihin.

Suunnittelussa tultiin siihen päätökseen, että integraation ei tarvitsisi olla kokonaan synkroninen, mutta käytettävyyden takia sen pitäisi olla lähes reaaliajassa. Integraation välillä kulkeva tieto tulisi olemaan XML-merkintäkieltä.

Työn lähdemateriaalina käytettiin pääasiassa toisia opinnäytetöitä ja englanninkielistä kirjallisuutta. Myös internetistä haettuja englanninkielisiä oppaita käytettiin jonkin verran.

Suunnitelmia kehitettiin jatkuvasti opinnäytetyön aikana ja jokaisesta integraatiotavasta tehtiin oma suunnitelma. Itse integraatio odottaa vielä toteutusta. Toteutuksen työmäärä riippuu kokonaan integraation tekijästä, mutta työmääräksi arvioitiin noin puoli vuotta.

Avainsanat Integraatio, Suunnittelu, Jäsenverkko, OldTimerTimer

Sivut 34 s.

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme in Business Information Technology
Software Engineering

Author Saara Karilaakso **Year** 2016

Subject of Bachelor's thesis Integration planning of Jäsenverkko and OldTimerTimer

ABSTRACT

The commissioner for this thesis was Yoso Oy. It consisted of planning four different integration plans how Jäsenverkko and OldTimerTimer could be integrated.

The objective of this thesis was to create a plan, based on which the integration of Jäsenverkko and OldTimerTimer could be started. In addition, an attempt was made to evaluate the potential workload of the actual integration execution.

The plans were implemented by studying four different integration methods and a variety of integration-related matters. The plans were pondered by studying other theses, seeking possible ideas for the integration. During the planning, synchronicity, real-time and data formats were familiarized and editorialized.

The planning stage gave the conclusion that the integration doesn't need to be completely synchronous, but for its availability it should be almost in real time. The information passing between the integration should be XML-based messages.

The source material for this thesis was mainly from other thesis and English literature. Some websites, in the form of lecture-materials, was also used.

The plans were developed continuously during the thesis and from each integration method a plan was made. Integration itself is still waiting for implementation. The workload of the implementation depends completely on the integrations implementer, but the estimated time, for the workload, is about half a year.

Keywords Integration, Planning, Jäsenverkko, OldTimerTimer

Pages 34 p.

Sanasto

HTTP	Hypertext Transfer Protocol. Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
IETF	The Internet Engineering Task Force on internetprotokollien standardoinnista vastaava organisaatio.
JavaScript	Web-ympäristössä käytettävä dynaaminen komentosarjakieli.
JSON	JavaScript Object Notation on yksinkertainen avoimen standardin tiedostomuoto.
RSS	Really Simple Syndication on XML-kieleen perustuva verkkosyötemuoto.
SMTP	Simple Mail Transfer Protocol. Protokolla, jota käytetään viestien välittämiseen sähköpostipalvelimien kesken.
SQL	Structured Query Language. Kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä.
Web Service	Ohjelmistojärjestelmä, joka mahdollistaa keskenään yhteensopivan tietokoneiden välisen vuorovaikutuksen tietoverkon yli.
XML	Extensible Markup Language. Tiedostomuodon merkintäkieli.
YAML	YAML Ain’t Markup Language. Tiedostomuodon merkintäkieli.

SISÄLLYS

1 JOHDANTO

Integraation perusideana on yhdistää kaksi tai useampi järjestelmä toisiinsa. Viime vuosina on yhä enemmän integroitu eri järjestelmiä keskenään. Syynä on ollut järjestelmien tarve käyttää samoja tietoja tai niiden yhdistäminen on ollut sisällön vuoksi järkevää. Eli järjestelmillä on sama perusidea tai niissä molemmissa on samanlaisia ominaisuuksia. Juuri tästä on nimenomaan kysymys tässä opinnäytetyössä, jonka toimeksiantajana toimi Yoso Oy. Integraatio ei aina kuitenkaan ole helppoa järjestelmien erilaisuuksien takia. Harvoin suunnitellaan etukäteen, että joku järjestelmä yhdistettäisiin toiseen järjestelmään. Sitä on kuitenkin otettu enemmän huomioon kuin ennen. Tästä syystä integraatiosuunnittelu on tehtävä ja on tärkeä osa integraatiota.

Opinnäytetyön tavoitteena oli suunnitella, kuinka olisi mahdollista toteuttaa integraatio OldTimerTimerin ja Jäsenverkon välillä. Itse integraation toteutus ei kuulunut opinnäytetyöhön, vaan se on tarkoitus suorittaa erillisenä opinnäytetyönä. Tarkoituksena oli siis vain suunnitella tarkka suunnitelma, jonka jälkeen järjestelmiä voitaisiin alkaa lähteä yhdistämään keskenään.

Opinnäytetyössä käydään läpi, mitä integroitavat järjestelmät OldTimerTimer ja Jäsenverkko ovat. Opinnäytetyössä selvitetään, mitä integraatio on ja mitä sen tapoja on. Niiden lisäksi selvitetään mitä integraatioon liittyvät asiat, reaaliaika, synkronisuus, asynkronisuus, SOAP ja REST, tarkoittavat.

Integraation suunnittelu toteutettiin pohtimalla, mitä kannattaisi integroida järjestelmistä, missä muodossa tieto integroitaisiin, miten mikäkin integraatiotapa voitaisiin toteuttaa. Lisäksi pohdittiin käytetäänkö synkronisuutta vai asynkronisuutta, mikä reaaliaika on suunnitelmassa. Pohtimisen jälkeen piirreltiin erilaisia integraatiosuunnitelmia. Aivan lopuksi pohdittiin toteutuksen mahdollinen työmäärä, mitä integraation toteuttajan pitäisi ottaa huomioon toteutuksessa ja valittiin omasta mielestä paras integraatiosuunnitelma. Lisäksi mietittiin, kumpaa kannattaisi käyttää integraatiossa SOAPia vai RESTiä.

Opinnäytetyöllä pyritään vastaamaan seuraaviin tutkimuskysymyksiin: Mitkä ovat integraatiotavat? Miten Jäsenverkko ja OldTimerTimer voidaan ja kannattaa integroida? Mikä on mahdollinen työmäärä työlle?

2 INTEGRAATIO

Tässä luvussa käydään läpi, mitä integraatio on, sen tapoja ja millaista integraatio on nykypäivänä. Niiden lisäksi käydään läpi integraatioon liittyviä seikkoja, joita pitää ottaa huomioon integraatiota suunnitellessa.

2.1 Mitä on integraatio?

Integraatio on kahden tai useamman järjestelmän yhdistämistä yhdeksi toimivaksi kokonaisuudeksi. Tällä ei tarkoiteta sitä, että järjestelmien yhdistettyä olisi yksi ainoa järjestelmä. Se tarkoittaa, että tietoa siirretään toisesta järjestelmästä toiseen järjestelmään. Järjestelmät siis säilyvät omina järjestelminä, mutta saavat tietoa toisista järjestelmistä. Integraatio suoritetaan valittujen järjestelmien tiettyjen komponenttien välille ja hyvässä integraatiossa ei huomata järjestelmän käyttävän toisen järjestelmän komponenttia. (*Linthicum 1999, 3*.)

Integraatiota voi tapahtua yrityksen sisällä tai kahden tai useamman eri yrityksen välillä. Puhutaan sisäisestä ja ulkoisesta integraatiosta. Sisäinen integraatio on yrityksen sisällä tapahtuvaa integraatiota, jossa tietoa siirretään yrityksen omien järjestelmien välillä. Ulkoinen integraatio puolestaan on kahden tai useamman eri yrityksen välistä integraatiota, josta puhutaan yleisesti business-to-business (B2B) -integraatiosta. Siinä tieto kulkee kahden tai useamman yrityksen välillä internetin kautta, mihin tämän opinnäytetyön aihe liittyy. (*Hophe & Woolf 2009, 37*.)

Integraatiolla on erilaisia tasoja, joita pitää ottaa suunnittelussa huomioon. Integraatiolla on kolme erilaista tasoa joilla integraatio voidaan lähteä toteuttamaan, data-, viesti- ja prosessitaso. Datatasossa järjestelmät asetetaan käyttämään yhteistä tietokantaa tai tietokantojen välillä siirretään tietoja. Datatason perusidea on siis järjestelmien tietokantojen integrointi. Viestitaso keskittyy tiedon siirtämistä viesteinä eri järjestelmien välillä. Sen perusidea on siis integraation tiedon siirrossa tarvittavien viestien määritteleminen. Prosessitasossa analysoidaan prosessien tuottamat tiedot ja liiketoimintaprosessien tiedontarpeet. Sen perusidea on siis analysoida integroitavien järjestelmien eri prosessit ja niiden tarvittavat tiedot. Suunnittelussa näiden tasojen kautta lähdetään suunnittelemaan integraatiota, mutta suunnittelussa pitää ottaa huomioon myös muita integraation tasoja. (*Linthicum 1999, 18–22*.)

Aikaisemmat tasot keskittyvät integraation toteutukseen kun seuraavilla tasoilla keskitytään näkökulmaan, jolla pyritään suorittamaan integraatio. Nämä tasot ovat datataso, metoditaso, applikaatiotaso ja käyttäjätaso. Datatasolla integraatiossa pyritään keskittymään tietojen jakamiseen järjestelmien välillä ja niiden viesteihin. Applikaatiotasolla integraatiossa pyritään, että samat prosessit olisivat jaettu järjestelmien kesken. Metoditasolla järjestelmät jakaisivat metodeja, toimintalogiikkaa ja palveluja keskenään. Käyttäjätasolla puolestaan

keskitytään integroimaan järjestelmät käyttäjän kannalta toimivaksi kokonaisuudeksi. Opinnäytetyössä katsottiin suurimmassa osin integraation suunnittelua käyttäjätasolla. (*Linthicum 1999, 18–22*.)

2.2 Integraatio ennen ja nykyään

Integraatio ei ole uusi asia, mutta se ei myöskään ole vanha asia. Ennen järjestelmiä luodessa ei ajateltu, että tietoa tarvitaan jakaa järjestelmän ulkopuolelle, mikä on huomattavissa nykypäivänä integraatioita tehdessä. Tiedon määrä ja jakamisen tarve on kasvanut. Jotta vältyttäisiin mahdollisimman suurilta järjestelmämuutoksilta, oli se sitten yrityksen sisällä tai ulkopuolella, integraatioita on alettu tekemään yhä enemmän. Integraatio yrityksen sisällä tarkoittaa sitä, että yrityksessä on tarve käyttää jotain tietoa toisessakin järjestelmässä. Järjestelmien paremman käytettävyyden takia järjestelmät halutaan integroida, jotta nopeutettaisiin toimimista. Yrityksen ulkopuolista integraatiota tarvitaan kun esimerkiksi tietoa pitää jakaa yrityskumppanien kesken. (*Linthicum 1999, 23–25*.)

Integraatiota hankaluutta vanhat tietojärjestelmät, joita voisi kutsua savupiippujärjestelmiksi. Ne ovat tehty siten, että niitä tehdessä ei ole koskaan ajateltu, että tietoa pitäisi viedä järjestelmästä ulospäin. Tämä voi aiheuttaa suurta ongelmaa integraatiota tehdessä. Järjestelmien vanhanaikaisuuksista ja sen aiheuttamista ongelmista huolimatta integraatioita on alettu tekemään yhä enemmän viime vuosina. Tämän takia integraatio ajatusta on alettu yhä enemmän ajatella järjestelmiä tehdessä. (*Linthicum 1999, 11–12*.)

Nykyään järjestelmiä pitää usein alkaa integroida toiminnan helpottamiseksi, esimerkiksi terveysalalla. Isoja integraatioprojekteja on alkanut myös tulla puheenaiheeksi ja yksi tällainen projekti on Kansallinen palveluväylä, josta on lähiaikoina paljon puhuttu.

Kansallisen palveluväylän ideana on luoda tiedonvälityskerros. Sen tarkoitus on helpottaa ja yksinkertaistaa kansalaisten, yritysten ja yhteisöjen asiointia viranomaisten kanssa ja tehdä siitä turvallisempaa. Tällä yritetään edistää julkisen hallinnon avoimuutta ja parantaa julkisen hallinnon palvelujen laatua. Väylä tullaan toteuttamaan virolaisen tiedonvälityskerroksen pohjalta. Projekti on vielä vaiheessa ja toukokuussa 2015 julkaistiin palvelunväylän tilannetta, jossa ilmoitettiin projektin jatkokehityksestä (*Konttaniemi 2015*).

2.3 Synkronisuus ja asynkronisuus

Yksi tärkeä kysymys integraatiossa on, onko integraatio synkroninen vai asynkroninen. Suunnittelussa sen takia pitää miettiä, kummalla keinolla saadaan isompi hyöty integraatiosta. Mitä sitten synkronisuus ja asynkronisuus sitten tarkoittavat?

Synkronisessa järjestelmässä järjestelmä lähettää viestin kohdejärjestelmälle, joka ottaa sen vastaan ja vastaa alkuperäiselle järjestelmälle. Synkronisuudella tarkoitetaan siis sitä, että viestin lähettäjä odottaa vastausta viestin saajalta. Synkronisessa järjestelmässä ei voida lähettää uutta viestiä ennen kuin kohdejärjestelmä on vastannut ensimmäiseen viestiin. Viestit voivat synkronisessa järjestelmässä kulkea molempiin suuntiin, mikä tarkoittaa, että synkroninen viestintä on kaksisuuntaista viestintää. (*Linthicum 2003, 155*.)

Asynkronisessa järjestelmässä puolestaan järjestelmä lähettää viestin kohdejärjestelmälle, mutta sen ei tarvitse odottaa kohdejärjestelmän vastausta. Eli jos kohdejärjestelmä sattuu olemaan kaatunut tai muuten tavattomissa viesti siirtyy jonoon odottamaan kunnes kohdejärjestelmä vastaa. Alkuperäinen järjestelmä voi lähettää uusia viestejä kohdejärjestelmälle, vaikkei tämä ole vastannut vielä ensimmäiseen. Uudet viestit asettuvat jonoon niin kauan kuin jonon maksimimäärä ei tule täyteen. Asynkronisuudella tarkoitetaan, että kommunikointi ja sen kulku on yhdensuuntaista. (*Linthicum 2003, 155*.)

Yhtä oikeaa ratkaisua ei ole kysymykseen, kumpaa kannattaa integraatiossa käyttää, synkronisuutta vai asynkronisuutta. Molemmilla on sekä hyviä että huonoja puolia. Integraatiota suunnitellessa pitää ottaa huomioon molempien sekä hyvät että huonot puolet ja yrittää miettiä, mikä olisi paras ratkaisu juuri mietittävään suunnitelmaan. (*Apurva 2013*.)

Synkronisuus mahdollistaa tiedon saamisen toisessa järjestelmässä lähes saman tien. Se mahdollistaa virheilmoitukset, jos viestin lähettämisessä ilmestyi ongelmia. Huonoa on se, että synkronisuus vaatii vastauksen kohdejärjestelmältä. Jos kohdejärjestelmä on saavuttamattomissa, tieto ei mene perille ennen kuin kohdejärjestelmä vastaa. Seuraavaa viestiä ei voida myöskään lähettää ennen kuin aikaisempi on käsitelty. (*Apurva 2013*.)

Asynkronisuudessa viestejä voidaan lähettää, vaikkei kohdejärjestelmä vastaakaan yksi toisensa jälkeen. Viestit menevät jonoon, josta ne sitten menevät kohdejärjestelmälle kun se vastaa. Tämä kuitenkin tarkoittaa sitä, että viestit eivät välttämättä ole välittömästi saatavilla toisessa järjestelmässä vähemmän vuorovaikutuksen takia. Ongelmana on myös se, että lähetetyt viestit voivat täyttää viestijonon, jos kohdejärjestelmä on saavuttamattomissa. Tästä ongelmasta ei tule mitään viestiä lähettäjäjärjestelmälle, eikä myöskään muita virheilmoituksia. (*Apurva 2013*.)

2.4 Reaaliaika

Integraation suunnittelussa on tärkeää miettiä integraation reaaliaikaa. Reaaliajalla tarkoitetaan reagoimista tapahtumiin oikea-aikaisesti, ja järjestelmän tilan olevan ajan tasalla jatkuvasti. Tosiaikainen järjestelmässä on mahdollista

nähdä sen hetkinen tieto millä hetkellä hyvänsä, esimerkiksi pankkitilin saldon tai Facebook-viestit.

Reaaliaikaisuuden ideana on, että jollekin tehtävälle asetetaan ensimmäinen ja viimeinen sallittu toteutumisaika. Esimerkiksi viesti lähetetään kahden eri järjestelmän välillä. Tästä lasketaan toteutusaika, jonka aikana viestin pitää saavuttaa kohdejärjestelmä. On kuitenkin tilanteita, joissa ei voi suoraan suorittaa komentoa vaan pitää odottaa ennen kuin komennon voi suorittaa. Aika, joka kuluu ennen kuin vaadittavaan komentoon reagoidaan, kutsutaan vasteajaksi. Vasteaika voidaan tyypillisesti laskea etukäteen, koska järjestelmät eivät koskaan ole niin tarkkoja, että vasteaika olisi vakio. Vasteaika tullaan ottamaan huomioon kun asetetaan sallittua toteutumisaikaa. (*Linthicum 2003, 459–460*.)

Toteutumisajan rikkoutumiseen voidaan reagoida kahdella eri tavalla. Nämä reagointitavat jakavat reaaliajan kovaksi- ja pehmeäksi reaaliajaksi. Kova reaaliaika tarkoittaa sitä, että toiminnolle annetut aikarajat ovat ehdottomia. Eli komennolle annetaan ensimmäinen ja viimeinen sallittu ajanhetki. Jos komento on hitaampi tai nopeampi kuin aikarajat, eikä komentoa saada toimimaan aikarajan sisällä, katsotaan, että sitä ei kannata tuottaa lainkaan. Pehmeässä reaaliajassa aikarajojen rikkominen ei ole toimintavirhe, vaan toiminta tulee tehdä loppuun vaikka se myöhästyisikin. (*Electronics Hub 2015*.)

Integraatiota suunnitellessa reaaliaikaisuutta pitää miettiä siten, että pitääkö järjestelmien olla ajan tasalla vai voivatko ne olla hieman jäljessä. Oikeaa reaaliaikaa, eli aina ajan tasalla olemista ilman mitään pieniä heittoja, ei kuitenkaan ole mahdollista saavuttaa.

2.5 Integraatiotavat

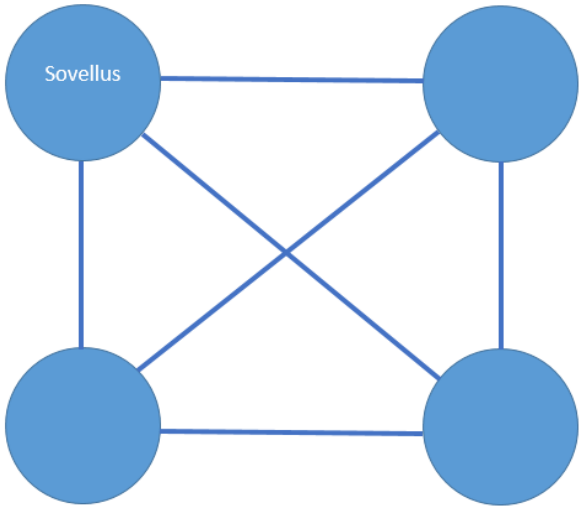
Integraatiolla on monta erilaista tapaa, kuinka toteuttaa integraatio. Jokaisella tavalla on omat vahvuutensa ja heikkoutensa, eikä koskaan ole oikeaa vaihtoehtoa, mitä tapaa käytetään integraatiota toteuttaessa. Jokainen tapa voi olla sekä oikea, että väärä vaihtoehto. Integraation päämäärä on yksi isoimmista asioista, mikä määrää mitä integraatiotapaa käytetään. Toinen on integroitavat kohteet. Lopputulos riippuu kokonaan siitä, miten integraatio on suunniteltu.

Integraatio on usein mahdollista toteuttaa millä tahansa tavalla, mutta se voi olla raskasta ja kallista ylläpitää, jos integraatiotapa ei ole soveliain tapa toteuttaa integraatio. Opinnäytetyössä tutustuttiin neljään erilaiseen integraatiotapaan, joilla integraatio voitaisiin toteuttaa.

2.5.1 Point-to-point

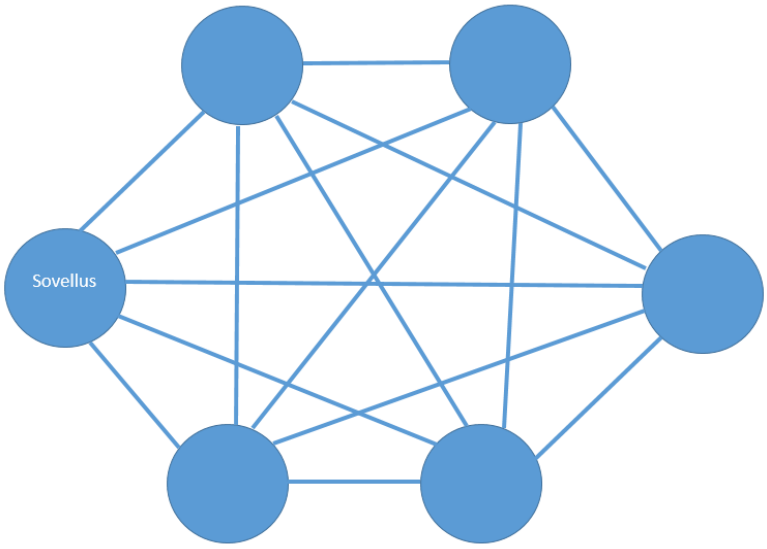
Point-to-point-integraatiotapa on integraatiotavoista yksinkertaisin. Nimensä mukaan point-to-point-integraatiossa järjestelmät integroidaan yhdestä pisteestä toiseen pisteeseen, ilman että niiden välissä olisi mitään välikonetta. Eli siinä tieto viedään yhdestä järjestelmästä suoraan toiseen järjestelmään. Kuviossa 1 nähdään, kuinka neljä eri järjestelmää on yhdistetty ja jokaisesta järjestelmästä lähtee vain yksi yhteys kuhunkin järjestelmään. Tavan rakenne on yksinkertainen ja on tämän takia se on tehokas ja nopea tapa toteuttaa integraatio. (*Linthicum 1999, 4–5*.)

Point-to-point-integraatiossa ei tarvitse olla useita yhteyksiä ja parhaimmassa tapauksessa integraatio on suoritettu vain kahden järjestelmän välillä, joiden välillä on vain yksi yhteys. Sitä on helppo hyödyntää pienissä integraatioratkaisuissa yksinkertaisuutensa takia. (*Linthicum 1999, 4-5*.)



Kuvio 1 Point-to-point-integraatio

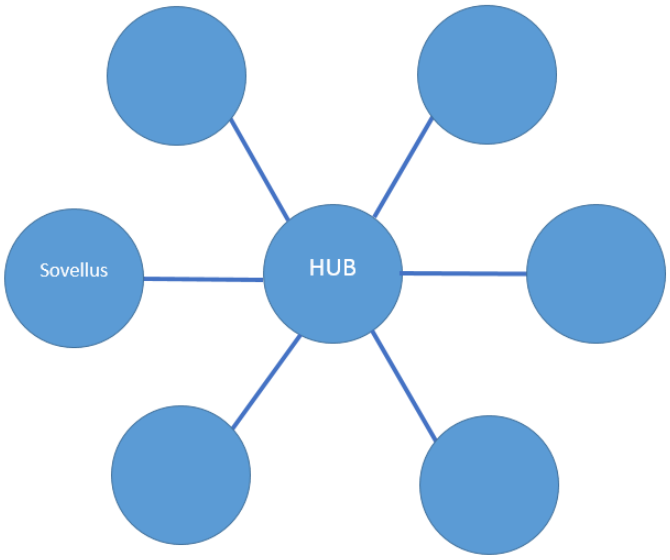
Vaikka Point-to-point-integraatio on yksinkertainen, mikä on hyvä asia, se on myös sen heikkous. Järjestelmien määrän kasvaessa rakenteesta tulee sekavampi ja sen joustavuus heikkenee. Tämän takia point-to-point-integraatiosta voi tulla hyvin raskasta ja kallista ylläpitää. Kuviossa 2 nähdään, kuinka kuusi järjestelmää on integroitu point-to-point-tavalla. Vaikka se on vielä selkeä kuva, siitä huomaa kuinka rakenne voi muuttua nopeasti sekavaksi. Lisäksi jos integraatioon halutaan lisätä myöhemmin uusi tai uusia järjestelmiä, niiden lisääminen ei ole helpointa hommaa. Point-to-point-integraatiotapa sopii muutaman järjestelmän yhdistämisessä, mutta se ei ole enää niin suosittu integraatiotapa. (*Linthicum 1999, 5–9*.)



Kuvio 2 Sekavampi point-to-point-integraatio

2.5.2 Hub-and-spoke

Hub-and-spoke on kehittyneempi tapa point-to-point-integraatiotavasta. Siinä kaikki lähetettävät tiedot käsitellään erillisessä, sovellusten väliin sijoitetussa, integraatiojärjestelmässä (hub). Integraatiojärjestelmä osaa erotella lähettävät ja vastaanottavat sovellukset, jonka takia se osaa lähetetään tiedon oikealle järjestelmälle käsiteltyään sen. Hub-and-spoke-tavan rakenne on selkeämpi, eikä niin monimutkainen kuin point-to-point-tavan rakenne, kuten kuviossa 3 nähdään. (*Tahvanainen 2012, 4*.)



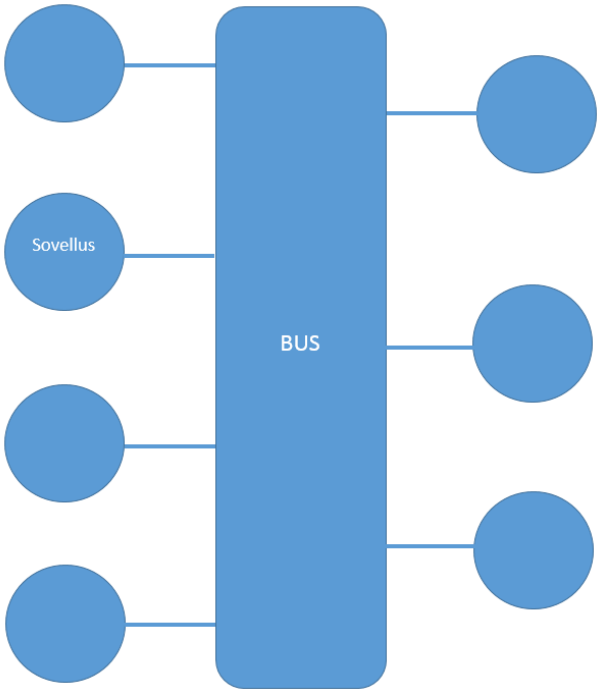
Kuvio 3 Hub-and-spoke-integraatio

Vaikka hub-and-spoke on selkeämpi kuin point-to-point, sillä on myös omat heikkoutensa. Suurin haaste integraatiotavassa on kaksisuuntaisen yhteyden hankaluus. Integraatiojärjestelmän pitää käsitellä tietoa jakamattomasti eri järjestelmien välillä. Järjestelmän yhdistäminen integraatiojärjestelmään on helppoa, mutta suurin työ integraatiossa kohdistuu järjestelmien yhdistämispisteisiin, josta järjestelmät yhdistetään. Järjestelmien pitää olla tietoisia toisista järjestelmistä, jotka ovat liitetty integraatiojärjestelmään, prosessoinnin mahdollistamaksi. Tämä ei ole ongelma yrityksen sisällä tapahtuvassa integraatiossa, mutta voi olla isokin kynnys yritysten välisessä integraatiossa. (*Tahvanainen 2012, 5*.)

Integraatiojärjestelmä on myös usein kovan rasituksen alla kaiken tiedon käsittelemisen takia. Tämän takia se usein tarvitsee paljon tehoa ja levytilaa. Sen pitää myös olla turvallinen, jonka takia tietoturvan pitää olla erittäin tarkka sekä integraatiojärjestelmässä ja järjestelmien yhdistämispisteessä. Ongelmista huolimatta hub-and-spoke on yksi suosituimmista tavoista suorittaa integraatio keksikokoisissa projekteissa. (*Tahvanainen 2012, 5–6*.)

2.5.3 Bus

Bus-integraatiotavassa ideana on erillinen tietoväylä, joka vastaanottaa tietoa järjestelmiltä ja hakee tietoa järjestelmille. Kuviossa 4 nähdään bus-integraation rakenne 7 eri järjestelmän välillä. Bus-integraatiossa järjestelmä lähettää tiedon tietoväylään, josta tietoa tarvitsevat järjestelmät käyvät hakemassa sen. Bus-integraatiotavassa erittäin hyväpuoli on se, että järjestelmien ei tarvitse olla tietoisia toisten järjestelmien olemassaolosta, kuten hub-and-spoke-integraatiossa, vaan tietoväylä hoitaa kaiken. Yritysten välinen integraatio ei ole siis, bus-integraatiossa, niin iso ongelma tai niin iso kynnys kuin hub-and-spoke-tavassa. (*Hophe & Woolf 2009, 135–138*.)

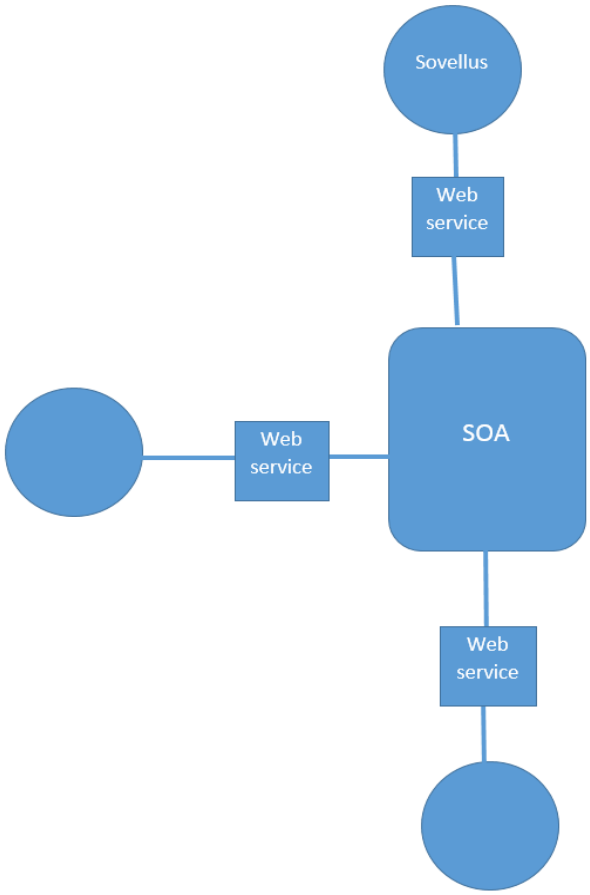


Kuvio 4 Bus-integraatio

Vaikkakin bus mahdollistaa helpomman tavan integroida yritysten välillä silläkin löytyy muiden tapaan huonoja puolia. Bus-tietoväylä on vain mekanismi, jolla ei ole omaa logiikkaa tai tietoa, miten tieto pitäisi siirtää ja kenelle. Siksi järjestelmien pitää tietää, mitä tietoa he haluavat. Yleinen ongelma on myös siirrettävän tiedonmuoto, joka ei aina ole järjestelmien välillä yhteen sopiva. Silloin tietoväylässä pitää suorittaa tietomuunnoksia, mikä voi tehdä ratkaisusta monimutkaisen. (*Hophe & Woolf 2009, 135–138*.)

2.5.4 SOA

SOAn (Service oriented architecture) eli palvelukeskeinen integraation perusidea on luoda hyvin skaalautuva ja uudelleen käytettävä integraatiojärjestelmä. Siinä tieto tuodaan toisesta järjestelmästä tarjoamalla sitä yleisen käyttörajapinnan kautta. Integroidut järjestelmät ovat sidottu löysästi SOAhan. Järjestelmissä on suunniteltu erilaisia avoimia prosesseja ja toimintoja. Näiden prosessien ja toimintojen tehtävä on toimia joustavina, itsenäisinä ja avoimina palveluina. Kuviossa 5 nähdään SOAn perusrakenne. Sen keskeinen ajatus on, että järjestelmät käyttävät web servicejä, joita pitäisi pystyä aina käyttämään avoimien standardi rajapintojen kautta. Niiden avulla pystytään mahdollistaa joustava ja järjestelmästä riippumaton vuorovaikutus järjestelmien välille. Vuorovaikutus voidaan toteuttaa vaikka bus-integraatiotavalla. (*Tahvanainen 2012, 7–8*.)



Kuvio 5 SOA-integraatio

SOA on hyvä mahdollinen tapa toteuttaa integraatio yritysten sisällä tai eri yritysten välillä. Tämä johtuu siitä, että verkkopalvelut toimivat vaikka järjestelmiä kehitettäisiin, eivätkä sovelluskehitystekniikat vaikuta niiden toimintaan. Avoimet rajapinnat mahdollistavat muutosten yhdenmukaisuuden ja käytön säilymisen. (*Tahvanainen 2012, 8.*)

Ongelmia kuitenkin SOAssa on kuten muissakin integraatiotavoissa. Suurin ongelma ilmenee yhteyksien määrän kasvaessa. Silloin vasteaikakin kasvaa, mikä hidastuttaa toimintaa ja tuottaa ongelmia. SOA perustuu myös XML:n (Extensible Markup Language), mikä on SOAn yksi suurimmista heikkouksista. XML:n tietoturva ei ole turvallisin. Sen sekaan on helppo lisätä SQL (Structured Query Language) lausekkeita, mikä on yksi hakkerien hyökkäys tapa. Pahimmassa tapauksessa voidaan tuhota tärkeitä tietoja tietokannoista tai arkaluonteista tietoa voi joutua vääriin käsiin. Täten tietoturva pitää olla tärkeä asia SOAn toteutuksessa. (*Tahvanainen 2012, 8–9.*)

2.6 SOAP ja REST

Web service on ohjelmistojärjestelmä, joka mahdollistaa vuorovaikutuksen kahden tai useamman tietokoneen välillä internetin kautta. Se on tärkeä osa integraatiota, varsinkin B2B-integraatioiden kanssa, sillä juuri sen avulla järjestelmien vuorovaikutus on mahdollista. On monta erilaista tapaa toteuttaa web service ja niistä kaksi on SOAP ja REST.

SOAP on tavoista vanhempi. Se oli alun perin Microsoftin perustama, mutta siirtyi myöhemmin IETF:n omistukseen. SOAP tukee useita laajennuksia, mutta vain niitä laajennuksia käytetään, mitä tarvitaan tiettyyn tehtävään. Sen yksi tärkeä osa on, sen sisälle rakennettu virheiden käsittelijä. Jos pyynnössä on virhe, vastaus sisältää virheinformaation, jota voi hyödyntää ongelmien korjaamiseen. Tämä on tärkeää varsinkin silloin kun kyseessä ei ole oma web servicesi. Muuten mietittäisiin missä meni vikaan. (*Mueller 2013*.)

SOAP käyttää XML-viestintää, mikä toimii paremmin internetin kautta viestittämisessä kuin aikaisempien teknologioiden binaarinen viestintä. XML:stä voi tulla kuitenkin erittäin kompleksi. Joissakin ohjelmointikielissä joudutaan kirjoittamaan pyynnöt manuaalisesti, mikä tulee ongelmaksi SOAPissa sillä se ei hyväksy virheitä. Jotkut kielet voivat käyttää oikoteitä, mitä SOAP tarjoaa. Tämä voi auttaa vähentämään vaivaa pyynnön luomisessa ja vastauksen lähettämisessä. Vaikeusaste riippuu kielestä jota käytetään. (*Mueller 2013.*)

REST on uudempi tapa toteuttaa web serviceitä. Se on kevyempi vaihtoehto kuin SOAP koodin kannalta. SOAP voi osoittautua hankalaksi esimerkiksi JavaScriptiä koodatessa, jolloin koodin määräksi tulee hirveän suuri. REST, XML:n sijaan, käyttää yksinkertaista URL:ää pyynnön teossa. Joissakin tapauksissa lisätietoa joudutaan toimittamaan erityisellä tavalla. Suurin osa, RESTiä käyttävät web servicet, käyttävät URL-lähestymistapaa tiedon hakemiseen. Toisinkuin SOAP, REST ei tarvitse käyttää XML:ää tuottamaan vastauksen. RESTissä voidaan käyttää vaikka JSON- tai RSS-kieltä. RESTillä voit saada ulostulon, jonka tarvitset, muodossa, mikä on helppo kopioida kieleen, mitä tarvitset applikaatiossa. (*Mueller 2013*.)

Tapaa valittaessa pitää miettiä RESTin ja SOAPin vahvuuksia. SOAP on vaihtoehdoista raskaampi valinta. Sen hyviä puolia on RESTiin verrattuna kuitenkin sen kieli ja alusta. SOAP ei myöskään vaadi HTTP:tä, toisin kuin REST, ja sitä voidaan käyttää myös SMTP:n kanssa, eikä ole syitä, miksi sitä ei voisi käyttää muillakin keinoilla. SOAP toimii myös hyvin hajautetussa yritysympäristössä, RESTin vaatiessa Point-to-Point-kommunikaation. (*Mueller 2013*.)

REST on puolestaan helpompi käyttää suurimmaksi osin ja on joustavampi kuin SOAP. Se ei tarvitse kalliita välineitä vuorovaikutuksessa web servicen kanssa ja sen oppiminen on nopeampaa. REST tukee monia eri kieliä, on nopeampi ja lähempänä muita web teknologioitten suunnittelua. (*Mueller 2013*.)

Tapaa valittaessa pitää vielä muistaa, että web servicet eivät aina tue molempia vaihtoehtoja, joten se pitää ottaa huomioon valinnassa, jos ei rakenna omaa web serviceä. Hyvin harva tukee molempia. Jotkut sanovat toisen keinon olevan parempi kuin toinen, mutta tämä ei ole totta. On tilanteita jolloin REST on parempi kuin SOAP ja toisin päin.

3 OLDTIMERTIMER

Tässä luvussa käydään läpi lyhyesti, mikä OldTimerTimer on. Sen lisäksi käydään läpi, mikä OldTimerTimerin tilanne on ja kuka omistaa sen.

OldTimerTimer on Hämeenlinnan ammattikorkeakoulun, HAMKkin, omistama ja valmistama android-sovellus. Sovelluksen on tarkoitus toimia tapahtumien tarjoajana, muistuttajana ja seuraajana vanhuksille.

Sovelluksen perusideana on se, että käyttäjä voi hakea erilaisia tapahtumia. Sovellus myös tarjoaa hänen lähellä olevia tapahtumia. Käyttäjä voi tallentaa nämä tapahtumat sovellukseen, jos hän haluaa osallistua niihin. Sovellus muistuttaa tapahtumista hälytyksellä käyttäjälle ennen niiden alkua. Tapahtuman jälkeen sovellus kysyy osallistuiko käyttäjä tapahtumaan vai ei. Jos käyttäjä vastaa ei, niin sovellus kysyy miksi. Tämän ideana on se, että voitaisiin seurata käyttäjien, tässä tilanteessa vanhusten, aktiivisuutta tapahtumiin ja mahdollisesti kirjata niitä ylös.

Alun perin idea oli suunnattu vain vanhuksille ja OldTimerTimer sisälsi myös muita ominaisuuksia kuten lääkkeiden muistutuksen. Nämä tuli kuitenkin karsittua pois sovelluksen kehityksen edistyessä. Sovellusta on myös ajateltu jakaa muillekin kuin pelkästään vanhuksille. Opinnäytetyötä on tehty siinä näkökulmassa, että OldTimerTimer oltaisiin jakamassa kaiken ikäisille.

Sovelluksen on tarkoitus tulla käytettäväksi tabletilla, mutta mobiilille suunniteltu käyttöliittymä on myös tehty. Tabletti valittiin siksi, että se todettiin paremmaksi vaihtoehdoksi vanhuksille, sen suuremman näytön takia.

Sovellus on vasta kehitysvaiheessa, eikä vielä tarjolla muille. Opinnäytetyön aloitusvaiheessa valmistajilla oli ideana seurata tapahtumia selvittääkseen, miksi vaikka joku todella aktiivinen käyttäjä ei osallistukaan enää niin aktiivisesti tapahtumiin. Myös oli tarkoituksena tarjota vanhusten sukulaisille mahdollisuus seurata vanhusten toimintaa. Tämä toteutettiin toisena opinnäytetyönä, jossa luotiin REST-arkkitehtuurin mukainen web service, mikä vaatii vielä integroimista OldTimerTimeriin (Jussilainen 2015).

4 JÄSENVERKKO

Tässä luvussa käydään läpi työnantaja, tehtävänanto ja työnantajan omistama Jäsenverkko, mikä se on ja sen tilanne. Lisäksi käydään lävitse, mitä yhteistä sillä on OldTimerTimerin kanssa ja miksi ne kannattaisi yhdistää.

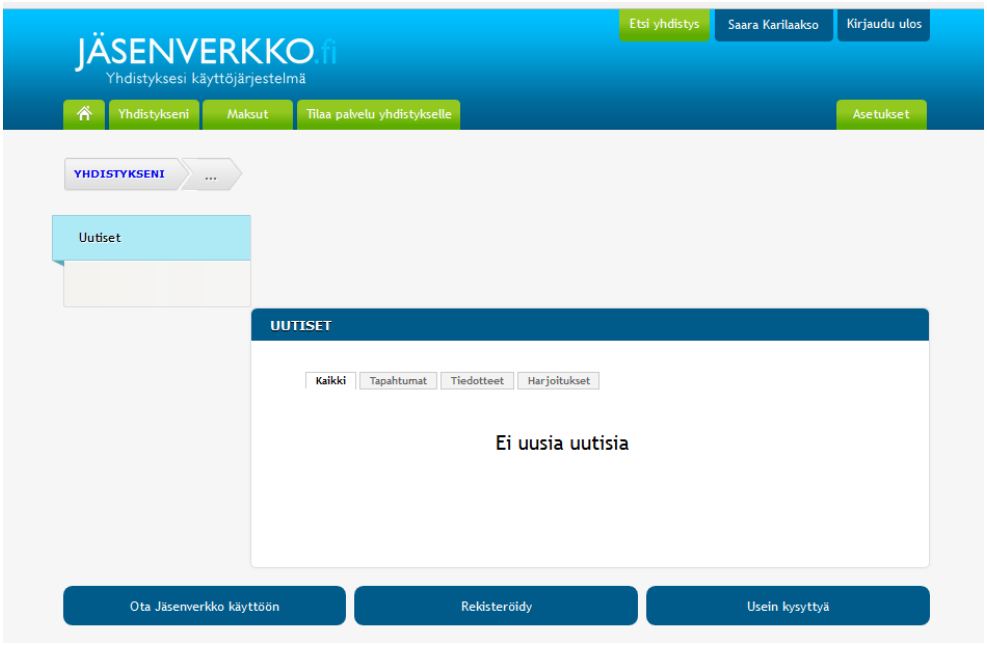
4.1 Mikä on Jäsenverkko?

Jäsenverkko on Yoso Oy:n omistama ja kehittämä verkkopalvelu. Sovelluksen ideana on, että yritys tai ryhmä voi tehdä ryhmän palveluun. Ryhmän kautta jäsenet voivat maksaa jäsenmaksuja, ilmoittautua tapahtumiin, lukea tiedotteita ja päivittää omia yhteystietoja.

Ryhmän järjestää yksi henkilö, joka voi hallinnoida maksuja, reskontraa, ilmoittautumista, tapahtumia, yhdistyksen tietoja ja organisaatiota eri tasoilla. Sovellus on ilmainen sillä ehdolla, että ryhmät ottavat vastaan ilmoituksia lähidemokratiasta, joiden tulisi näkyä sovelluksen etusivulla. Ryhmien ei tarvitse ottaa kaikkia lähidemokratia ilmoituksia, vaan ryhmän järjestäjä päättää, mistä asioista he ottavat ilmoituksia vastaan. Ryhmät voivat siis valita heistä kiinnostavia/koskevia aiheita ja jättää muut aiheet, jotka eivät koske heitä, pois. Palvelun sivusto on Jäsenverkko.fi ja kuvassa 1, nähdään palvelun etusivu.

Kuva 1 Jäsenverkon etusivu

Palvelun jäsen voi myös kuulua moneen eri ryhmään. Siis jäsen voi kuulua vaikka oman yrityksen ryhmään, saada heidän ilmoituksia ja kuulua oman jalkapalloseuran ryhmään ja saada myös sen ilmoituksia.
Jäsenet voivat myös hoitaa lastensa asioiden hoitoa omilla tunnuksilla. Kuvassa 2 nähdään, jäsenen etusivu.



Kuva 2 Jäsenverkon jäsenen etusivu

Sovelluksen ideana on siis lyhyesti toimia ryhmien tiedotusalustana jäsenille. Vaikka sovellus on pitkällä, kuten OldTimerTimer, Jäsenverkko on vielä kehitysvaiheessa ja sivu on vasta demoalustalla.

Kuten OldTimerTimer, Jäsenverkko tarjoaa mahdollisia tapahtumia ryhmistä, johon jäsen kuuluu ja jäsen pystyy ilmoittautumaan niihin. Sovelluksilla on samanlainen perusidea tiedon tarjoamisesta. Jäsenverkko kuitenkin tarjoaa ryhmissä laajempaa tietoa tapahtuvista tapahtumista, kun taas OldTimerTimer on keskittynyt tapahtumien ilmoittamiseen.

OldTimerTimerilla ja Jäsenverkolla on siis yhteinen perusidea, vaikkakin Jäsenverkko menee pidemmälle, mutta tämän takia sovellukset kannattaisikin integroida. OldTimerTimerin avulla Jäsenverkko saa lisää mahdollisia käyttäjiä itselleen. Jäsenverkon käyttäjät saisivat täten tiedon sovelluksesta, jota voi käyttää puhelimen kautta, tapahtumien muistuttamiseen. Jäsenverkon avulla puolestaan voidaan tuoda tietoa lähidemokratian asioista OldTimerTimerin käyttäjille. Täten integraation avulla OldTimerTimerin käyttäjät saavat enemmän heitä koskevaa tietoa ja Jäsenverkko saa tarjoamaansa tietoa suuremmalle ihmisjoukolle. Jäsenverkon kautta voisi myös ilmoittaa erilaisia tiedotuksia OldTimerTimerista. Esimerkiksi jos OldTimerTimerista on löytynyt ongelma ja sitä korjataan, tai jos OldTimerTimerissa on uusi päivitys saatavilla. Integ-

raation avulla kumpaakin järjestelmää voitaisiin jakaa siis suuremmalle yleisölle ja saataisiin mahdollisesti enemmän käyttäjiä. Isompi yleisö saa myös tietoa lähidemokratia asioista.

4.2 Toimeksiantaja ja toimeksianto

Toimeksiantajana toimi Yoso Oy, joka on vuonna 2005 perustettu, Espoossa sijaitseva, sovelluskehitykseen ja asiantuntijapalveluihin erikoistunut yritys. Yoso tarjoaa liiketoimintaa tukevia asiantuntijapalveluita arkkitehtuuriratkaisuissa ja toiminnan kuvaamisessa ja kehittämisessä. He ovat erikoistuneet palvelupohjaisiin toimintamalleihin sekä niiden hallinnointiin. Yoso toteuttaa myös räätälöityjä ohjelmistoja, integraatio- sekä tunnistusratkaisuja. He auttavat ohjelmistojen hankintaprosesseissa ja kilpailutuksissa ja tarjoavat vankkaa kokemusta Open Source -järjestelmien evaluoinneissa, hankinnoissa.

Aiheen idean saatiin 2015 loppu keväästä, Digital Service Development -kurssilla, missä se oli tarjolla opinnäytetyöaiheena. Toimeksiantona oli tutustua integraatioon ja suunnitella kuinka integraatio voitaisiin toteuttaa Jäsenverkon ja OldTimerTimerin välillä. Lopputuloksena piti tuottaa valmis suunnitelma, jonka jälkeen integraatiota voitaisiin lähteä toteuttamaan.

Aihe valittiin, koska se kuulosti mielenkiintoiselta ja toista opinnäytetyötä ei ollut silloin vielä tehty. Suurin syy, miksi aihe valittiin, oli se että aikaisempaa kokemusta integraatiosta ei ollut ja haluttiin kokeilla jotain uutta.

5 JÄRJESTELMIEN INTEGROINNIN SUUNNITTELU

Tässä luvussa käsitellään itse järjestelmien integraation suunnittelua, omaa pohdiskelua, erilaiset suunnitelmat ja työmäärän arvio. Niiden lisäksi pohditaan, mikä omasta mielestä olisi paras suunnitelma vaihtoehto, jonka perusteella voitaisiin lähteä integroimaan Jäsenverkkoa ja OldTimerTimeria.

5.1 Integroitavat prosessit

Integraation suunnittelu aloitettiin tutkimalla, mitä prosesseja järjestelmissä on. OldTimerTimer on tehty yksinkertaiseksi tapahtumien tarjoajaksi, joten siinä on vähemmän prosesseja kuin Jäsenverkossa. Tämä ei kuitenkaan vaikuttanut integraation suunnitteluun. Tarkoituksena oli etsiä mahdollisimman samanlaisia prosesseja, jotka kannattaisi integroida keskenään. Kummassakin järjestelmässä on monta erilaista prosessia, mutta niistä kuitenkin onnistuttiin löytämään samankaltaisia prosesseja.

Yhteiset prosessit olivat tapahtumasta ilmoittaminen ja ilmoittautuminen tapahtumaan. Jäsenverkossa pääsee katsomaan tapahtumia niille tarkoitetussa välilehdessä ja OldTimerTimerissa pääsee hakemaan lähiseudun tapahtumia. Molemmissa oli tämän jälkeen mahdollista ilmoittautua tapahtumaan. Näiden kahden prosessin integroiminen olisi käyttäjälle suotuisa, koska jos käyttäjä ilmoittautuu toisessa järjestelmässä tapahtumaan, hän näkisi sen molemmissa järjestelmissä. Tapahtumien ilmoittamisessa käyttäjä näkisi kaikki mahdolliset hänelle tarjottavat tapahtumat ilman, että hänen pitäisi mennä katsomaan toista järjestelmää. Näille integroitaville prosesseille alettiin sitten miettiä, mikä herättäisi integraation liikkeelle.

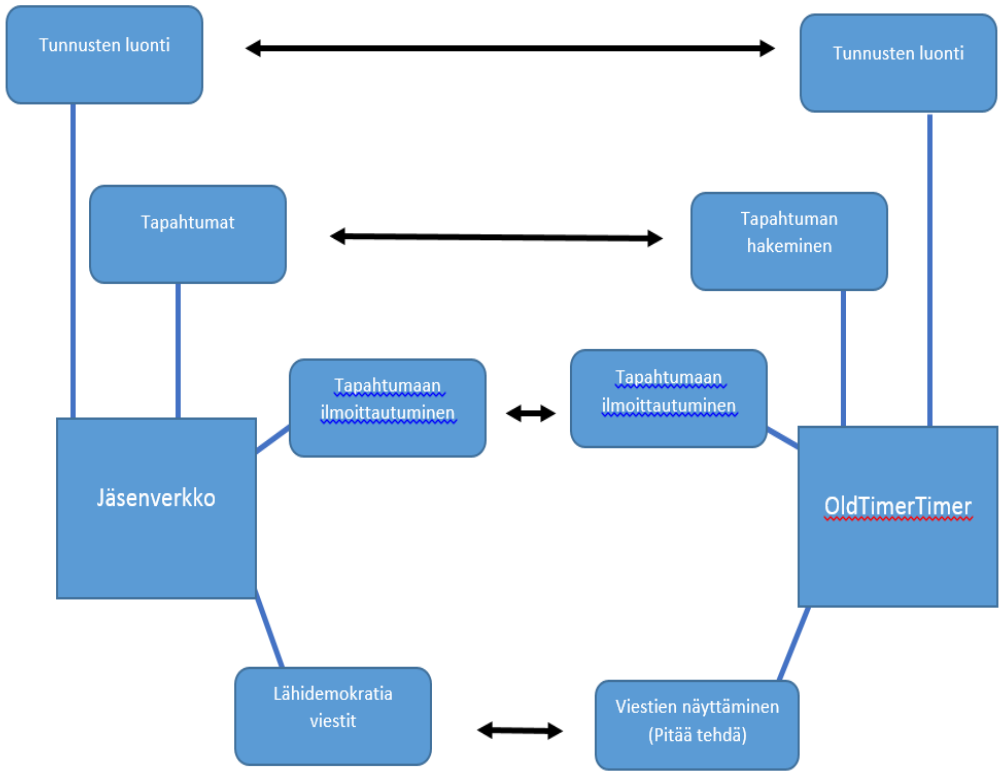
Tapahtumien ilmoituksen integraatio herätettäisiin liikkeelle kun OldTimerTimerissa haettaisiin tapahtumia tai Jäsenverkossa mentäisiin katsomaan tapahtumia. Kun tapahtumia mennään katsomaan, järjestelmät kaipaisivat mahdollisia tapahtumia toisesta järjestelmästä. Ilmoittautumisen integraatio herättäisiin liikkeelle puolestaan silloin kun jommassakummassa järjestelmässä ilmoittaudutaan tapahtumaan. Kun käyttäjä, jommassakummassa järjestelmässä, menee katsomaan tapahtumia, johon on ilmoittautunut, niin tämä järjestelmä tulee tarvitsemaan toisen järjestelmän ilmoittautumistiedot.

Rekisteröityminen on myös molemmissa järjestelmissä. Sitä ei alun perin koettu järkeväksi prosessiksi integroida, mutta ajatus muuttui kun saatiin idea toisesta opinnäytetyöstä (Kava 2014). Rekisteröitymistä ei koettu ideankaan jälkeen järkeväksi toteuttaa siten, että jos toisessa järjestelmässä tehtäisiin tunnus, niin tunnus tehtäisiin toiseen järjestelmään. Ideana oli, että ainoastaan silloin lähetetään tietoa tunnuksista kun OldTimerTimeriin tehdään uusi tunnus. Tämän avulla saataisiin Jäsenverkkoa tarjottua laajemmalle yleisölle. Tunnusten

integraation herätettäisiin liikkeelle kun OldTimerTimerissa tehtäisiin uusi tunnus. Integraation kautta Jäsenverkko tulee kaipaamaan OldTimerTimeriin luotuja uusia tunnuksia, jotta tämä voi tehdä niistä samalaiset tunnukset.

Yhteisten prosessien lisäksi mietittiin, mitkä muut prosessit voitaisiin myös integroida. Muista prosesseista ei korostunut prosessia, mikä olisi hyvä olla mukana myös toisessa järjestelmässä. Prosesseissa ei ollut paljon sellaisia, mitä voisi tuoda integraation avulla toiseen järjestelmään niin, että se ei tekisi järjestelmästä sekavaa. Lähidemokratia-viestit olivat kuitenkin sellainen asia, mikä nousi usein ylös suunnitelmia miettiessä. OldTimerTimerissa ei ole varsinaista paikkaa viesteille, mutta ne olisi hyvä tuoda siihen. Viestit saataisiin suuremmalle yleisölle helpommin, ja vaikka käyttäjä ei tulisikaan käyttämään Jäsenverkkoa, hän saisi tietää lähidemokratia asioista helpommin. Lähidemokratia viestien integraatio herätettäisiin liikkeelle kun Jäsenverkkoon lisätään viestejä. OldTimerTimer tulisi tarvitsemaan Jäsenverkon lähidemokratia viestejä kun viestejä mennään katsomaan.

Prosesseista saatiin siis eriteltyä neljä prosessia, jotka kannattaisi integroida keskenään. Kuviossa 6 nähdään integroitavat prosessit.



Kuvio 6 Integroitavat prosessit

Tämän jälkeen alettiin miettiä tarkemmin, mitä kaikkea tietoa kulkisi järjestelmien välillä ja missä muodossa. Ennen kuin tähän mentiin, suunniteltiin kuitenkin, miten järjestelmät käyttäytyisivät integraation jälkeen. Olisivatko järjestelmät riippuvaisia toisistaan vai olisivatko ne itsenäisiä?

Huolimatta siitä, mitä integraatio tuo sovelluksille, järjestelmien kannalta olisi järkevämpää, etteivät järjestelmät tule toisistaan riippuvaisia vaan pysyisivät itsenäisinä. Syynä, miksi näin päädyttiin, on se, että OldTimerTimer on opiskelijoiden kehittämä järjestelmä ja sen kehityksestä ei ole varmaa tietoa kuinka se jatkuu. OldTimerTimeria tullaan kehittämään vain silloin kun, joku kiinnostuu sen kehityksestä.

5.2 Siirrettävät tiedot ja niiden muoto

Enemmän aikaa kului tietojen muodon suunnitteluun kuin itse tietoihin, joita siirretään järjestelmien välillä. Nämä tiedot oikeastaan tuli jo aikaisemmin mainittua. Tapahtumien ilmoittautumisista tietenkin lähtee ilmoittautumistieto toiseen järjestelmään. Itse uusista tapahtumisista lähtee tapahtumatieto toiseen järjestelmään. Jäsenverkosta lähidemokratia viestit lähetetään OldTimerTimeriin.

Tunnusten luomisessa kuitenkin piti pysähtyä miettimään, mitä kaikkea tietoa voidaan lähettää järjestelmien välillä. Tieto ei saa olla arkaluonteista. Tieto lähetetään internetin kautta, eikä se koskaan takaa täydellistä turvallisuutta, vaikka integraatioon tehtäisiin erinomainen tietoturva. OldTimerTimerissa tunnuksen luomisessa ei kuitenkaan kysellä, mitään erittäin henkilökohtaista, josta hakkerit voisivat saada jotain irti. OldTimerTimerin tunnuksiin liittyvät tiedot voidaan siis lähettää sellaisinaan, kunhan tunnusten luontiin ei lisätä henkilötunnuksen kysymistä tai mitään muuta kovin henkilökohtaista tietoa. Tämä on kuitenkin hyvin epätodennäköistä OldTimerTimerin kannalta, mutta se oli asia, mikä piti ottaa huomioon.

Kysymykseksi nousi, mitä kukin järjestelmä omistaa siirrettävistä tiedoista. Tämä kysymys ei ollut helppo kysymys ja siihen ei kunnolla osattu edes vastata. Päätös joka valittiin, oli että molemmat järjestelmät omistaisivat omat tietonsa. Jäsenverkko omistaisi lähidemokratiaviestit, jotka tuotaisiin näytille OldTimerTimeriin. Molemmat järjestelmät omistaisivat omat tapahtumansa, jotka jaettaisiin molempien kesken.

Tiedostomuoto oli asia, mitä mietittiin pitkän aikaa opinnäytetyötä tehdessä. Tiedonmuoto pitäisi olla sellainen, että molemmat verkkopalvelu, Jäsenverkko, ja android-sovellus, OldTimerTimer, voisi käyttää tiedostoja. Tätä varten lähdettiin katsomaan erilaisia avoimen standardin tiedostomuotoja. Tämänlaisia tiedostomuotoja olivat XML ja JSON. Muitakin mahdollisia tiedostomuotoja kuten YAML mietittiin, mutta XML ja JSON olivat ne tiedostomuodot, jotka tulivat useasti mieleen suunnitelmia tehdessä. Näiden kahden muodon välillä

pyörittiin kauan, yrittäen miettiä kumpi olisi järkevämpi käyttää. Kysymyksenä oli myös tietoturva molempien tietostomuotojen kannalta.

Selvä päätös, kumpaa muotoa käytettäisiin, lopulta tuli itse asiassa toisesta opinnäytetyöstä. Opinnäytetyötä tehdessä OldTimerTimeriin tehtiin samalla aikaa toista opinnäytetyötä, jota käytiin kuuntelemassa loppuseminaarissa. Opinnäytetyössä tehtiin REST-arkkitehtuurin mukainen web service, jossa päädyttiin käyttämään XML:lää JSON:in sijaan, koska JSON:in tukemista ei todettu OldTimerTimerin kannalta elintärkeäksi (Jussilainen 2015). Tämä ei tarkoittanut sitä, että JSON-tietomuotoa ei kannattaisi käyttää, mutta opinnäytetyön päätös nosti XML:n käyttöideaa korkeammaksi kuin JSON:in. Lopulta päädyttiin siis siihen, että tiedot järjestelmien välillä siirrettäisiin XML muodossa.

5.3 Suunnittelu, integraatiotapojen vertailua ja suunnitelmat

Suunnitelmien teko aloitettiin integraatiotapoihin tutustumisen jälkeen. Kun integroitavat prosessit oli päätetty, alettiin tekemään ensimmäisiä suunnitelmia. Tietojenmuotoa mietittiin samalla kun suunnitelmia aloitettiin piirtämään. Aluksi suunnitelmia piirrettiin käsin paperille, sillä se oli nopeampaa kuin tietokoneella piirtäminen. Lisäksi tekijällä oli tietokone ongelmia opinnäytetyön alkupuolella. Kaavioista tuli usein sekavia, ja kukaan muu ei olisi saanut niistä selvää kuin itse tekijä. Nämä piirrokset olivat vain mallina lopullisia suunnitelmakuvioita varten. Niiden avulla yritettiin välttää mahdollisia suuria muutoksia kun suunnitelmat piirrettäisiin tietokoneella. Integraatiosuunnitelmat päätettiin suunnitella jokaisella neljällä tavalla, johon oli tutustuttu. Täten saataisiin isompi kuva, mikä olisi paras tapa toteuttaa integraatio ja mikä ei olisi niin hyvä idea. Suunnitelmia pohtiessa mielipiteet eri integraatiotavoista muuttuivat ja toivat esille erilaisia kysymyksiä.

Point-to-point-integraatio oli ensimmäinen suunnitelma, josta suunnittelu aloitettiin. Aluksi point-to-point-integraatiota pidettiin mahdollisena tekniikkana, jolla integraatio suoritettaisiin sen yksinkertaisuuden takia. Ajanmittaa kuitenkin tämä tunne muuttui. Suunnitelma haluttiin pitää mahdollisimman selvänä, joten yhteyksiä yritettiin muodostaa mahdollisimman vähän. Nämä suunnitelmat kuitenkin heitettiin roskiin kun aloitettiin miettiä, että onko integraatio järkevää toteuttaa point-to-point-tekniikalla.

Point-to-point voi olla yksinkertainen ja helppo tapa toteuttaa, mutta sen avulla ei välttämättä saada kaikkea mahdollista hyötyä irti OldTimerTimerista ja Jäsenverkosta. Syynä on sen yksinkertaisuus, joka jo kerrottiin luvussa 2.5.1. Jos Jäsenverkkoon tai OldTimerTimeriin halutaan liittää, joku uusi järjestelmä, sen lisääminen ei välttämättä ole niin helppoa kuin muissa integraatio ratkaisuissa.

Integraatio voidaan siis toteuttaa point-to-point-tavalla, mutta silloin järjestelmien välillä pitää olla monta eri yhteyttä. Paras tapa käyttää point-to-point-tekniikkaa on silloin kun järjestelmien välillä on vain yksi yhteys. Jos halutaan

toteuttaa integraatio siten, että se täyttää aiemmin kerrotut tehtävät, järjestelmien välillä pitää olla ainakin kuusi yhteyttä. Tämän takia point-to-point-integraatiota ei todettu järkeväksi tavaksi toteuttaa, mutta ideoita oli ja niistä luotiin yksi kokonainen suunnitelma.

Hub-and-spoke-suunnitelma aloitettiin samaan aikaan kuin point-to-point-suunnitelman suunnittelu ja sen avulla alettiin miettimään web serviceä. Hub-and-spoke oli toinen menetelmä, joka jo heti alussa oli sellainen, mikä tuntui mahdolliselta integraatioratkaisulta. Sen viehätyksenä oli se, että hubiin muiden järjestelmien lisääminen jälkeenpäin ei olisi mahdotonta tai ei pitäisi olla hankalaa. Sillä olisi myös mahdollista toteuttaa kaikki integroitavat kohdat, joita oli ajateltu.

Ensimmäisenä tutkittiin, miten tieto liikkuisi hubin sisällä, mutta sitten kysymykseksi tuli, mitenkä tiedot lähetettäisiin hubiin. Ensimmäisessä versiossa ajateltiin, että järjestelmät ottaisivat suoran yhteyden hubiin internetin kautta, mutta kysymykseksi tuli olisiko järjestelmillä mahdollisuus tehdä niin. OldTimerTimerista ei ainakaan löytynyt mahdollista tapaa tehdä niin tietoa hakiessa, ja nopeasti valinta siirtyi web serviceen. Web serviceä oli jo alussa ajateltu käyttää apuna tietojen lähettämisessä järjestelmien välillä, jos kohdejärjestelmä ei vastaa. Kysymykseksi tuli sitten, millä tavalla tiedot tuotaisiin hubista web serviceen. Eli miten web service toteutettaisiin (luku 5.4), käyttäen SOAPia vai RESTiä. Lopulta sekä SOAP, että REST merkattiin suunnitelmaan mahdolliseksi vaihtoehdoksi.

Bus-integraatiotavan suunnittelu alkoi hitaammin kuin point-to-point- ja hub-and-spoke-tapojen. Syynä oli se, ettei sen käsitystä aluksi heti tajuttu ja siihen tutustuminen vaati hieman enemmän aikaa. Suurin syy hämmennykselle oli se, ettei busin ja hub-and-spokein eroavaisuuksien hahmottaminen ollut aluksi helppoa.

Alussa bus-integraatiota ei pidetty kovinkaan isona integraatioratkaisuna, mutta kun suunnitelmassa päästiin eteenpäin, huomattiin, että se saattaisi olla ehkä parempi vaihtoehto kuin hub-and-spoke. Tieto kulkisi hyvin samantapaisesti kuin hub-and-spokessa, mutta järjestelmien ei tarvitsisi tietää toisesta järjestelmästä mitään. Tämä helpottaisi integraation toteutusta siten, ettei tarvitsisi olla niin huolissaan järjestelmien yhdistyskohdista. Busiin olisi myös helppo lisätä uusia järjestelmiä, jos niin tahdotaan tulevaisuudessa, ja integraatio yritysten välillä olisi helpompaa kuin hub-and-spokessa.

Tietoa ei myöskään käsitellä yhdessä paikassa vaan jokaiselle tiedolle on määritetty oma toiminto, jossa se käsitellään. Tämä jakaa takkaa, joka mahdollistaa, ettei kaikkea tarvitse koodata yhteen hubiin. Hubin rasituskin vähentyisi ja ei tarvitsisi olla niin huolissaan kuinka hub kestää rasituksen, vaikkakin sitä ei tietenkään saa unohtaa.

SOA-integraatiotapaa aloitettiin suunnittelemaan viimeisenä. Idea SOA-suunnitelmaan saatiin aiemmasta tehdystä opinnäytetyöstä (Kekäläinen 2010).

Integraatiotavan omaksuminen kesti paljon enemmän kuin muut tavat. Integraatiomenetelmän idean valaisi opettajan lyhyt selitys SOAsta, mitä ei lukemalla saatu niin helposti selvitettyä. Sen perusideana on siis, että on palvelin, jolta järjestelmä hakee haluamaansa tietoa. Ratkaisuun olisi helppo lisätä uusia järjestelmiä, jos myöhemmin tulee tarve, ja järjestelmien yhdistäminen ei olisi hankalaa.

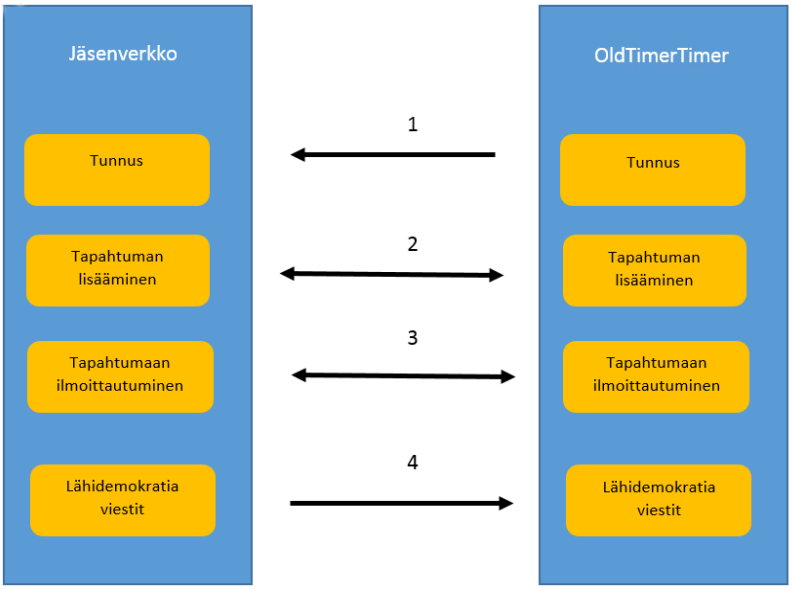
Toteutuksessa pitäisi tehdä kaksi eri SOAta rinnakkain, sillä järjestelmä jolta haetaan tietoa, ei voi hakea toisesta järjestelmästä tietoa samalla yhteydellä. Järjestelmä, joka toimii palvelimena voi vain lähettää tietoa. Tämän takia pitäisi luoda kaksi eri SOAta. Tämä ei kuitenkaan tulisi olemaan ongelma.

Idea, että käytetään kahta SOAta tuli nopeasti, mutta sitten tuli kysymykseksi, miten tieto käsiteltäisiin kun järjestelmä lähettää sen. Kun tätä alettiin miettimään, kävi ilmi, että bus-integraatiota voidaan hyödyntää SOAssa. SOAhan voisi toteuttaa samantyylisen ratkaisun kuin bus-integraatio tai käyttää bus-integraatiota, jota oli suunniteltu jo aikaisemmin. Lopulta pohtimisen jälkeen jo suunniteltua bus-integraatiota päätettiin käyttää tässä integraatiosuunnitelmassa.

Lopputulokseksi saatiin siis valmistettua neljä erilaista integraatiosuunnitelmaa. Näistä suunnitelmista piirrettiin sitten lopulliset kuviot.

5.3.1 Point-to-point

Kuviossa 7 nähdään aikaansaatu suunnitelma. Siinä tieto lähetettäisiin järjestelmien välillä kahden web servicen avulla.



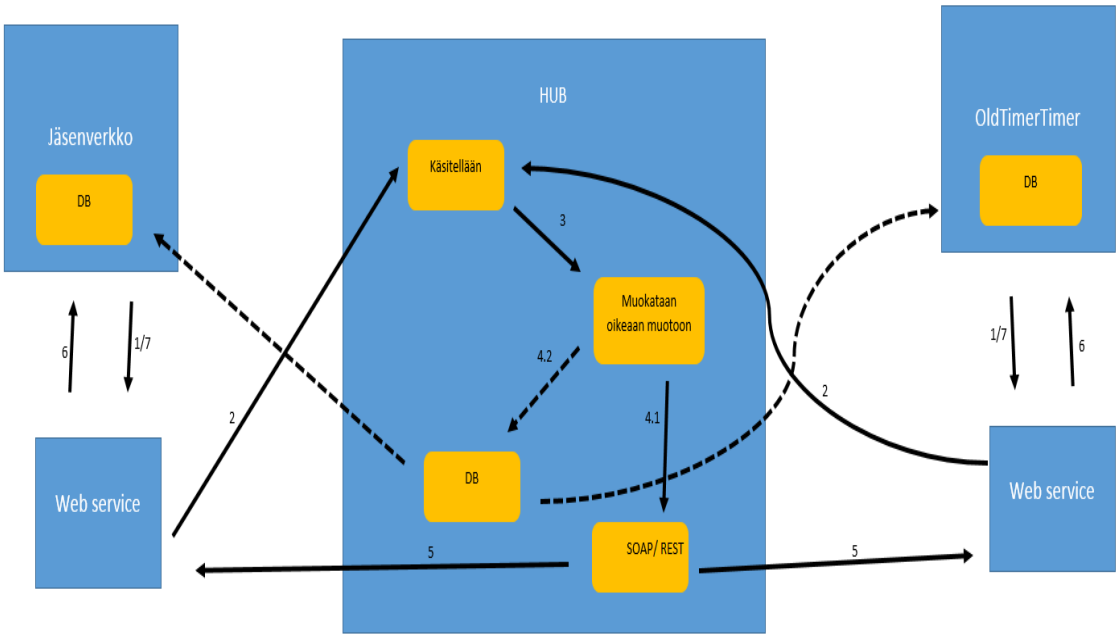
Kuvio 7 Point-to-point-integraatiosuunnitelma

Kuvioon 7 merkityt kohdat:

1. OldTimerTimeriin tehdään tunnus, jonka tiedot lähetetään sille luodulle web servicelle. Web service lähettää tiedot Jäsenverkon web servicelle ja siten Jäsenverkolle, jossa luodaan tunnus samoilla tiedoilla tietokantaan.
2. Toiseen järjestelmään lisätään tapahtuma, josta lähetetään tiedot toisen järjestelmän tietokantaan. Järjestelmien välillä tässä kohtaa on kaksi eri yhteyttä.
3. Toisessa järjestelmässä ilmoittaudutaan tapahtumaan, josta lähetetään tiedot toiseen järjestelmään. Järjestelmien välillä tässä kohtaa on kaksi eri yhteyttä.
4. Jäsenverkosta lähetetään lähidemokratia viestit OldTimerTimeriin, jossa ne tuodaan esille.

5.3.2 Hub-and-spoke

Kuviossa 8 nähdään lopullinen aikaan saatu hub-and-spoke-integraatiosuunnitelma. Tämä on yksi vaihtoehto kuinka hub-and-spoke-integraatio voitaisiin toteuttaa.



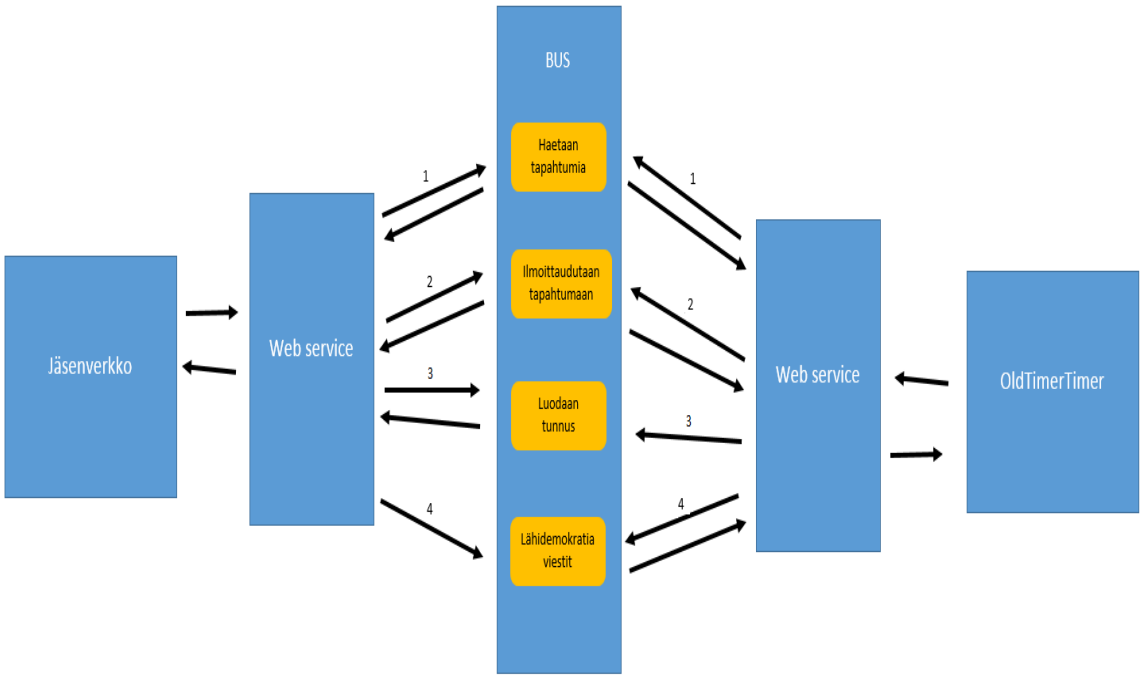
Kuvio 8 Hub-and-spoke-integraatiosuunnitelma

Kuvioon 8 merkityt vaiheet:

1. Järjestelmä on hakemassa/lähettämässä tietoa. Järjestelmä lähettää komennon/tiedon web serviceen.
2. Web service ottaa yhteyden hubiin, jossa komento/tieto käsitellään, kummalle järjestelmälle komento/tieto lähetetään.
3. Komento/tieto lähetetään muokattavaksi oikeaan muotoon, jossa tarvittaessa muokataan komento/tieto kohdejärjestelmälle sopivaksi
4.
   4.1. Komento/tieto lähetetään lähetettäväksi SOAPin tai RESTin avulla, jos yhteyttä tietokantaan ei suoraan saada.
   4.2. Komento/tieto lähetetään suoraan kohdejärjestelmän käsiteltäväksi/tietokantaan.
5. Komento/tieto lähetetään SOAPin tai RESTin avulla web serviceen, jossa se odottaa kunnes se saa yhteyden tietokantaan.
6. Komento/tieto lähetetään kohdejärjestelmän käsiteltäväksi ja sen tietokantaan.
7. Kohdejärjestelmä vastaa kyselyyn, lähettämällä tiedon web serviceen (1 kohta)

5.3.3 Bus

Kuviossa 9 nähdään bus-integraatiolla aikaan saatu integraatiosuunnitelma. Tämä on yksi vaihtoehto kuinka integraatio voitaisiin toteuttaa.



Kuvio 9 Bus-integraatiosuunnitelma

Kuvioon 9 merkityt kohdat:

1. Järjestelmä on lähettämässä/hakemassa tiedon uudesta tapahtumasta. Järjestelmä lähettää tiedon web serviceen, joka lähettää sen bus-tietoväylään. Tieto käsitellään, onko tallennettava tieto vai komento, jolla haetaan tietoa. Tietoa lähettäessä tieto lähetetään tallennettavaksi busiin. Tarvittaessa tieto muutetaan oikeaan muotoon ennen kuin se tallennetaan. Tietoa haettaessa komento käsitellään busissa, josta lähetetään tapahtumien tiedot web servicelle ja sitä kautta järjestelmälle takaisin, jossa samoilla tiedoilla lisätään uusi tapahtuma.

2. Järjestelmä on lähettämässä/hakemassa tiedon tapahtumaan ilmoittautumisesta. Järjestelmä lähettää tiedon web serviceen, joka lähettää sen bus-tietoväylään. Tieto käsitellään, onko tallennettava tieto vai komento, jolla haetaan tietoa. Tietoa lähettäessä tieto lähetetään tallennettavaksi busiin. Tarvittaessa tieto muutetaan oikeaan muotoon ennen kuin se tallennetaan. Tietoa haettaessa komento käsitellään busissa, josta lähetetään tiedot ilmoittautumisista web servicelle ja sitä kautta järjestelmälle takaisin, jossa samoilla tiedoilla lisätään ilmoittautuminen.

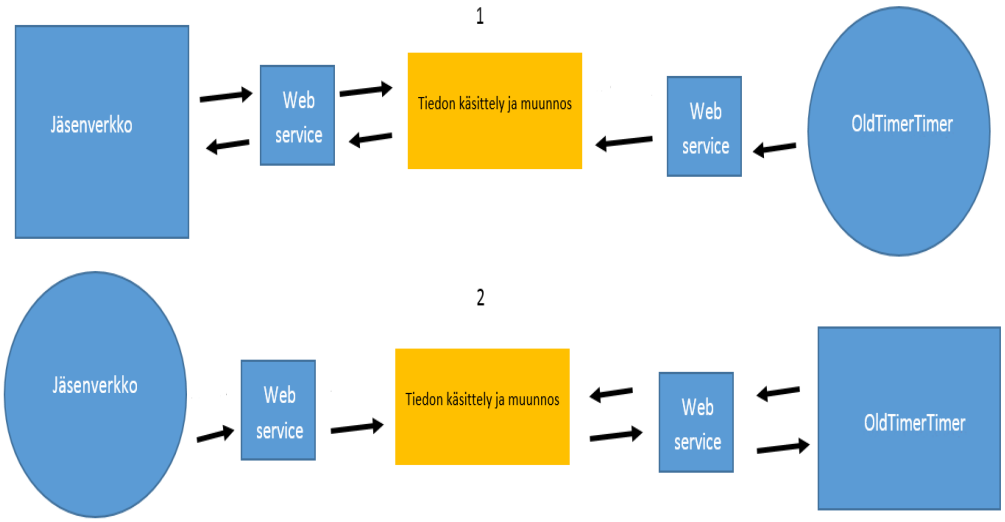
3. OldTimerTimer on lähettämässä tiedon uudesta tunnuksesta. Järjestelmä lähettää tiedon web serviceen, joka lähettää sen bus-tietoväylään. Tieto käsitellään, onko tallennettava tieto vai komento, jolla haetaan tietoa. Tässä tilanteessa tallennetaan ja tieto lähetetään tallennettavaksi busiin. Tarvittaessa tieto muutetaan oikeaan muotoon ennen kuin se tallennetaan. Jäsenverkko on hakemassa tietoa tunnuksista. Järjestelmä lähettää komennon web serviceen, joka lähettää sen busiin. Komento käsitellään busissa, josta lähetetään tunnusten tiedot web

servicelle ja sitä kautta Jäsenverkolle. Jäsenverkossa luodaan samoilla tiedoilla tunnus tai jos käyttäjällä on jo tunnus, niin kyseiset tilit yhdistetään.

4. Jäsenverkko on lähettämässä tiedon lähidemokratia viestistä. Järjestelmä lähettää tiedon web serviceen, joka lähettää sen bus-tietoväylään. Tieto käsitellään, onko tallennettava tieto vai komento, jolla haetaan tietoa. Tässä tilanteessa tallennetaan ja tieto lähetetään tallennettavaksi busiin. Tarvittaessa tieto muutetaan oikeaan muotoon ennen kuin se tallennetaan. OldTimerTimer on hakemassa tietoa lähidemokratia viesteistä. Järjestelmä lähettää komennon web serviceen, joka lähettää sen busiin. Komento käsitellään busissa, josta lähetetään viestien tiedot web servicelle ja sitä kautta OldTimerTimerille, jossa viestit tuodaan esille, niille tehdylle paikalle.

5.3.4 SOA

Kuviossa 10 nähdään SOA-integraatiolla aikaansaatu integraatiosuunnitelma. Tiedon käsittelemiseen käytetään aikaisempaa bus-integraatiosuunnitelmaa.



Kuvio 10 SOA-integraatiosuunnitelma

Kuvioon 10 merkityt kohdat:

1. Ensimmäinen SOA, jossa OldTimerTimerista haetaan tietoja. Keskellä hyödynnetään bus-integraatiota. OldTimerTimer lähettää tietoja tunnuksista, tapahtumista ja ilmoittautumista, jotka Jäsenverkko käy hakemassa busista.
2. Toinen SOA, jossa Jäsenverkosta haetaan tietoja. Keskellä hyödynnetään Bus-integraatiota. Jäsenverkko lähettää tietoja lähidemokratia viesteistä, tapahtumista ja ilmoittautumista, jotka OldTimerTimer käy hakemassa busista.

5.4 SOAP vai REST

Kuten luvussa 2.6 kerrottiin SOAPilla ja RESTillä on hyviä ja huonoja puolia. Tässä opinnäytetyössä ei tehty päätöstä kumpaa tapaa integraation toteutuksessa käytetään. Asiaa pohdittiin siltä osin, mitä opinnäytetyön tekijä itse lähtisi käyttämään ja miksi.

Molemmat ovat hyvää vaihtoehtoja, mutta REST osoittautui vaihtoehdoksi, jota opinnäytetyön tekijä itse lähtisi käyttämään. Syy tähän on tämän opinnäytetyön aikana toteutunut toinen opinnäytetyö, joka liittyi OldTimerTimeriin (Jussilainen 2015). Opinnäytetyössä toteutettiin REST-pohjainen web service, mitä voisi mahdollisesti hyödyntää Jäsenverkon ja OldTimerTimerin integraatiossa. Jos sen hyödyntäminen ei ole mahdollista niin siitä voidaan katsoa mallia. Toiset syyt, miksi opinnäytetyön tekijä tekisi integraation RESTillä, ovat sen joustavuus, helppous ja tuki monelle kielelle. REST on myös halvempi vaihtoehto toisin kuin SOAP, josta voi tulla äkkiä hyvinkin kallis.

RESTin käyttö integraation toteutuksessa on kuitenkin tämän opinnäytetyön tekijän ehdotus. Lopullista päätöstä ei haluttu tehdä sen takia, että opinnäytetyön tekijä ei itse tee integraatiota ja integraation tekijä voi haluta käyttää puolestaan SOAPia.

5.5 Synkronisuus, reaaliaika ja yhtenä ryhmänä vai jako eri ryhmiin

Suunnitelmien valmistuessa alettiin pohtimaan olisiko integraatio synkroninen. Ideaalina olisi, että kaikki tiedot olisi saman tien saatavissa toisessa järjestelmässä. Tämä kuitenkin huonontaisi ja hidastaisi järjestelmien toimintaa, varsinkin jos toiseen järjestelmään ei saataisi yhteyttä. Kysymykseksi tuli sitten, mitkä asiat täytyisi olla synkronisia, jotta integraatio toimisi halutulla tavalla.

Integraation ei tarvitsisi olla kokonaan synkronoitu, sillä kaikki kohdat eivät todellakaan tarvitse sitä, kuten lähidemokratia viestit, mutta tunnuksen luonti on tehtävä synkronisesti. Jos tunnusta ei ole luotu myös Jäsenverkkoon niin haluttuja tietoja ei saada Jäsenverkolta ja haluttuja tietoja ei myöskään mene Jäsenverkkoon. Tunnusten luominen täytyy olla synkroninen, jotta tiedon siirtyminen järjestelmien välillä olisi mahdollista ja toimisi halutulla tavalla.

Muiden prosessien ei siis tarvitse olla synkronisia, mutta kuinka usein niitä pitäisi sitten päivittää. Tätä lähdettiin miettimään käyttäjän näkökulmassa. Oletetaan, että tiedot päivitettäisiin kerran päivässä, aamuyöstä, järjestelmiin. Jos käyttäjä ilmoittautuisi Jäsenverkossa tapahtumaan, joka olisi samana päivänä kuin ilmoittautumispäivä, niin OldTimerTimer ei muistuttaisi siitä ajallaan. Tämä johtuisi siitä, että tieto ilmoittautumisesta ei olisi päivittynyt OldTimerTimeriin, koska tietojen päivitys tapahtuisi vain kerran päivässä. Tapahtumien

ilmoittautumisien päivittäminen pitäisi, saatavuuden kannalta, päivittää usein ja säännöllisesti. Itse tapahtumienkin pitäisi myös päivittyä säännöllisesti, jotta käyttäjällä olisi aina mahdollisuus olla tietoinen saatavista tapahtumista. Niiden ei tarvitsisi päivittyä niin usein kuin ilmoittautumisien, ehkä kerran päivässä. Ainoa, joka ei tarvitsisi kovin tiheää päivittämistä, olisi lähidemokratia viestit. Niiden päivittäminen riippuu kokonaan siitä, kuinka usein viestejä lisätään. Päivittyminen voi tapahtua, joka toinen päivä tai kerran viikossa.

Integraation ei siis tarvitse olla kokonaan synkroninen, mutta sitten kysymykseksi tuli integraation reaaliaika. Suunnitelmissa piti lähteä miettimään täytyisikö integraation olla reaaliaikainen. Vastaus kysymykseen tuli nopeasti kun lähdettiin miettimään OldTimerTimerissa olevaa tapahtumien muistutusta. Integraation pitäisi olla siis reaaliajassa, mutta kuinka tarkka sen pitäisi olla. Muutaman minuutin heitto ei haittaisi OldTimerTimerin toimintaa, mutta liian pitkä ero se ei kuitenkaan saisi olla. Integraatiossa ei tarvitse siis pyrkiä minuutin tarkkaan reaaliaikaan, mutta jotta OldTimerTimerin muistutus toimisi, niin kuin se halutaan, pitäisi reaaliaika saada viiden minuutin sisälle. Viiden minuutin viive ei olisi pahaksi. Reaaliaika kannattaisi olla pehmeää kovan sijaan, jotta integraatiossa ei kaaduttaisi mahdollisten myöhästymisien takia. Maailma ei kaadu, vaikka järjestelmät eivät aina voi pitää kiinni reaaliajasta.

Suunnitelmien saatua lähes valmiiksi piti alkaa pohtia, miten OldTimerTimer lisättäisiin Jäsenverkkoon. Opinnäytetyön alussa annettiin ohjeeksi, että se oli oma päätös, laitetaanko OldTimerTimer yhtenä isona joukkona Jäsenverkkoon vai jaettaisiinko se pienimpiin ryhmiin. Asiaa alettiin suunnittelemaan siten, että mietittiin, mitä lähidemokratia viestejä OldTimerTimeriin lähetettäisiin. Jos OldTimertTimer laitetaan yhtenä ryhmänä Jäsenverkkoon, kaikki tulisi saamaan samat lähidemokratia viestit.

On sellaisia lähidemokratia asioita, jotka ovat niin sanotusti kaikille, mutta on monia, jotka ovat vain tietyille ihmisille. Nuorten ei olisi järkevää saada tietoa vanhuksiin liittyvistä viesteistä. Viestit eivät kiinnostaisi nuoria kun ne eivät koske heitä. Sama asia on myös toistinpäin. Vanhukset eivät välttämättä ole kiinnostunut nuoriin liittyvistä viesteistä. OldTimerTimer voidaan lisätä Jäsenverkkoon yhtenä ryhmänä, mutta viestien valinta pitäisi olla sellaisia, jotka koskisivat sitten kaikkia tai suurinta osaa. Järkevintä olisi jakaa OldTimerTimer pienimpiin ryhmiin, mutta kysymykseksi tuli, mitkä olisivat nämä ryhmät.

Jako voitaisiin tehdä paikkakunnittain tai vaikka ammatin mukaan, mutta nämä eivät kuulostaneet niin järkeviltä vaihtoehdoilta. Jakaminen taas iän perusteella oli vaihtoehto, johon lopulta päädyttiin ja joka kuulostikin hyvältä vaihtoehdolta. Ikäryhmät piti kuitenkin sen jälkeen miettiä, joka oli hieman haasteellista. Mitkä olisivat järkevät ikäryhmät? Alle 18-vuotiaat olivat selvä yksi ryhmä, mutta 18-vuotiaasta ylöspäin oli vaikea päättää, missä vetää rajat. Lopulta päädyttiin, alle 18-vuotiaiden lisäksi, kolmeen ryhmään, jotka olivat 18-vuotiaista 29-vuotiaaseen, 30-vuotiaista 59-vuotiaaseen ja 60-vuotiaat ja vanhemmat. Jaon mahdollistamiseksi OldTimerTimeriin pitää tehdä rekisteröity-

miseen kohta, jossa kysytään käyttäjän ikä, jotta käyttäjä voidaan lisätä haluttuun ryhmään. Jakaminen on siten kannattavaa, että jokainen käyttäjä saisi omaan ikäryhmään liittyviä tietoja, mitkä voivat kiinnostaa heitä enemmän kuin, jonkun toisen ikäryhmän tiedot.

5.6 Suunnitelmien vertailu

Jokainen tehty suunnitelma lähetettiin toimeksiantajalle, mutta opinnäytetyössä haluttiin kertoa, mikä suunnitelma oli omasta mielestään paras. Suunnitelmien hyviä ja huonoja puolia mietittiin ja pohdittiin, mikä olisi järkevin tapa toteuttaa integraatio. Tämä todettiin vaikeaksi päätökseksi sillä integraatiota ei lähdetty itse toteuttamaan. Oma mielipide valitusta suunnitelmasta voisi hyvinkin muuttua jos sitä lähtisi itse toteuttamaan. Siksi suunnitelmia alettiin vertailla, minkä suunnitelman, tekijä itse, lähtisi toteuttamaan tiedossa olevilla tiedoilla.

Kuten aikaisemmin kerrottiin, point-to-point-integraatiosuunnitelmaa tehdessä (luku 5.3) koettiin jo, ettei se olisi järkevin tapa toteuttaa integraatio. Sen suunnitelma oli siis nopeasti siirretty sivuun. Hub-and-spoke-suunnitelmaa puolestaan pohdittiin kauankin. Se ei olisi huono tapa toteuttaa integraatio rakenteensa puolesta, mutta sitten alettiin miettiä mahdollisia kustannuksia. Järjestelmien käyttäjien määrä voi tulla hyvinkin suureksi, joten hubin pitäisi kestää kovaa rasitusta, joten sillä pitäisi olla paljon tehoa ja levy tilaa. Tämä voi tulla hyvinkin kalliiksi. Integraatio tulee maksamaan, mutta se voi tulla maksamaan vielä enemmän hub-and-spoke-integraation takia. Integraation mahdollistaminen yritysten välillä on myös hankalampaa hub-and-spoken kanssa toisinkuin bus- ja SOA-integraatioissa, joten itse ei koettu hub-and-spoke-tapaa parhaimmaksi vaihtoehdoksi.

Vaihtoehdoiksi jäi siis bus- ja SOA-integraatiosuunnitelmat. Kysymykseksi jäi, kummalla tavalla itse integraatio toteutettaisiin. Näiden kahden välillä ei osattu valita parasta vaihtoehtoa sillä molemmat suunnitelmat olisivat omasta mielestä parhain tapa toteuttaa integraatio. Molemmissa suunnitelmissa yritysten välinen integraatio ei ole haaste ja on helpompi toteuttaa kuin hub-and-spoke-tavassa. SOA-suunnitelmassa päädyttiin myös käyttämään bus-integraatiota mahdollisena vaihtoehtona tiedon käsittelyyn. SOA-suunnitelma nähtiin, jo sitä suunnitellessa, kiinnostavalta. Jos itse olisi lähdetty tekemään integraatiota, se olisi ollut mahdollisesti vaihtoehto, jota olisi käytetty, pelkästään oman kiinnostuksen takia. Parasta suunnitelmaa valittaessa ei kuitenkaan osattu sanoa kumpi suunnitelma olisi parempi vaihtoehto, joten päädyttiin tasapeliin. Integraatiota omasta mielestä kannattaisi lähteä tekemään, joko bus-integraatiosuunnitelmalla tai sitten SOA-suunnitelmalla.

5.7 Arvioitava työmäärä

Suunnitelmien lisäksi haluttiin arvioida mahdollista työmäärää integraatiolle. Oikeaa työmäärää oli vaikea arvioida, sillä se riippuisi kokonaan siitä kuka tekee integraation. Onko tekijä tehnyt aikaisempaa integraatiota vai ei? Tietääkö hän etukäteen jotain integraatiosta? Tietääkö hän integroitavat järjestelmät? Nämä olivat ensimmäisiä asioita, jotka tulivat mieleen työmäärää arvioidessa.

Oletetaan, ettei integraation tekijä tiedä integraatiosta yhtään mitään. Hänen pitäisi siis tutustua integraatioon, järjestelmiin, suunnitelmaan ja ohjelmiin, joilla integraatio tehdään. Tutustumiseen kuluu aikaa ennen kuin integraatiota voidaan edes aloittaa. Tutustumisen aikana tekijän pitää ottaa myös huomioon, missä tilassa järjestelmät silloin ovat ja kuinka paljon ne ovat muuttuneet sitten integraatiosuunnitelman. OldTimerTimer voi hyvinkin muuttua sen kehityksen aikana, jos siihen löydetään tekijöitä.

Itse ohjelmointiinkin pitää varata paljonkin aikaa. Tekijän ohjelmointitaidoista riippuen ohjelmoinnissa voi kulua, joko paljon tai vähän aikaa, ennen kuin saadaan tulosta aikaan. Ei kuitenkaan riitä, että tekijä saa integraation tehtyä, hänen pitää myös testata sitä. Testaukseen pitää varata paljon aikaa ja myös mahdollisiin korjauksiin, joita löydetään testauksen kautta. Tietoturva pitää myös varmistaa ja siihen pitää varata hyvin aikaa, jotta integraation tietoturva olisi hyvällä tasolla.

Lopullinen työmäärä siis riippuu integroinnin tekijästä, mutta silti yritettiin arvioida mahdollinen kuluva aika, joka tarvitaan integraation toteuttamiseen. Kuten aiemmin mainittiin, arvioiminen oli vaikeaa ja kysymykseksi tuli kuinka ahkerasti tekijä tekisi integraatiota. Tekisikö hän sitä opiskelujen tai työn ohella vai olisiko se hänen ainoa projektinsa. Lopulta päädyttiin siihen, että työhön kannattaa varata ainakin puoli vuotta, olettaen että tekijällä ei ole aikaisempaa kokemusta integroinnista. Työmäärän arvioinnissa oletetaan myös, että integroinnin tekijä työskentelee normaalin työajan puitteissa.

6 YHTEENVETO

Tässä luvussa kerrotaan yhteenveto siitä, mitä kaikkea on saatu aikaan opinnäytetyön aikana, mitä haasteita opinnäytetyön aikana ilmestyi ja integraation jatkokehitystä. Lisäksi kerrotaan integraatiota jatkavalle, mitä jatkokehityksessä pitää huomioida.

6.1 Lopputulos

Opinnäytetyöllä pyrittiin vastaamaan seuraaviin tutkimuskysymyksiin: Mitkä ovat integraatiotavat? Miten Jäsenverkko ja OldTimerTimer voidaan ja kannattaa integroida? Mikä on mahdollinen työmäärä työlle?

Opinnäytetyön aikana saatiin selvitettyä, mitä integraatio on ja tutustuttiin sen tapoihin ja siihen liittyviin erilaisiin asioihin. Integroitaviin järjestelmiin tutustuttiin ja pohdittiin, mikä olisi järkevin tapa integroida järjestelmät. Kerättyjen tietojen perusteella suunniteltiin neljä integraatiosuunnitelmaa, joissa käytettiin neljää eri integraatiotapaa, joihin oli tutustuttu. Otettiin myös kantaa integraatioon liittyviin asioihin, mikä niistä olisi paras vaihtoehto tässä integraatio tapauksessa. Niiden lisäksi onnistuttiin arvioimaan mahdollinen työmäärä itse integraation toteutukselle.

Lopputuloksena saatiin siis valmistettua neljä erilaista suunnitelmaa työnantajalle, kuinka OldTimerTimer ja Jäsenverkko voitaisiin integroida keskenänsä. Integraatio tultaisiin mahdollisesti toteuttamaan näiden tehtyjen suunnitelmien pohjilla.

6.2 Haasteet

Opinnäytetyötä tehdessä ei ollut kovin paljon haasteita ja itse opinnäytetyö ei aiheuttanut niitä, mutta muutama haaste ilmestyi opinnäytetyötä tehdessä. Ensimmäinen haaste, mikä aiheutti myös toisen haasteen, oli itse laitteisto. Opinnäytetyötä aloittaessa tekijällä hajosi hänen oma tietokoneensa, mikä ei kesällä aiheuttanut niin isoa ongelmaa. Hänelle ostettiin uusi kone, mutta tämä uusi kone temppuili koko syksyn ja sitä kuljetettiin huoltoon kerran kuukaudessa, mikä hidasti opinnäytetyön etenemistä. Tekijälle ostettiin halpa varakone, jota hän käytti kun uusi tehokkaampi kone vietiin huoltoon. Toinen haaste opinnäytetyölle oli kuitenkin ilmestynyt.

Laitteiston kokoaikainen temppuileminen aiheutti sen, että tekijän motivaatio uhkasi laskea opinnäytetyön alussa, sillä hän pelkäsi koko ajan, että tehokkaampi kone taas temppuilisi. Lisäksi varakone oli halvin hyllystä otettu tietokone eikä siinä ollut riittävästi tehoja. Tämä aiheutti sen, että tekijä odotti paremman tietokoneen palaamista ennen kuin jatkoi opinnäytetyötä. Tilanne kuitenkin muuttui kun parempi tietokone meni neljättä kertaa huoltoon, jonka jälkeen tekijä alkoi tekemään kunnolla, vaikka hitaasti, opinnäytetyötä varakoneellaan.

Kolmas haaste oli aiheen rajaus, mitä tapahtui opinnäytetyön alkuvaiheessa, kesällä ja syksyllä, kun toteuttaja alkoi pohtimaan suunnitelmia. Tekijällä ei ollut aikaisempaa kokemusta integraation ja sen suunnittelemisen kanssa. Tehtävän toimeksianto ja toimeksiantaja olivat myös antaneet aika vapaat kädet toteuttajalle ja aiheen rajaaminen oli alussa erittäin vaikeaa. Kun tekijä alkoi rakentamaan suunnitelmia, hän usein uppoutui aivan liian pitkälle, aiheuttaen itsellensä paniikin. Tietokoneen temppuileminen ei auttanut panikointia, mutta kun toteuttaja siirtyi varakoneen käyttöön, hän sai kunnolla rajattuaan aihe alueensa.

6.3 Jatkokehitys ja huomioitavaa

Järjestelmien integraation toteutus olisi seuraava vaihe jatkokehityksessä. Toteutus todennäköisesti suoritetaan toisena opinnäytetyönä, joten siinä saattaa kulua aikaa ennen kuin itse integraatio tehdään. Tämän takia integraation tekijän pitää ottaa huomioon järjestelmien tilanne. OldTimerTimeria kehitetään sitä mukaa, mitä opiskelijoilla on aikaa ja kiinnostusta asiaan. Integraation tekijän pitää siis tarkistaa, että järjestelmissä ei ole tapahtunut suuria muutoksia niiden sisällön takia. Molempien järjestelmien kehitystilanne on jo niin pitkällä, että niiden sisällön ei pitäisi kauheasti enää muuttua, mutta varmuuden vuoksi tämä pitää tarkistaa.

Järjestelmien tilanteen tarkistamisen lisäksi tietoturva on asia, joka pitää ottaa huomioon integraatiota tehdessä. Tieto liikkuu internetin kautta, joten siihen pitää kunnolla panostaa, jotta hakkerit eivät pääsisi kiinni tietoihin. Tietoturvaan pitäisi panostaa silti kunnolla, vaikka tieto kulkisikin vain yrityksen sisällä. Jotta käyttäjien tiedot olisivat turvassa, tietoturva pitää ottaa tosissaan ja käyttää aikaa sen varmistamiseen.

LÄHTEET

Apurva, T. 2013. Difference Between Synchronous and Asynchronous Messaging? Viitattu 10.11.2015
http://peoplesofttutorial.com/difference-between-synchronous-and-asynchronous-messaging/

Electronics Hub. 2015. Real Time Operating System (RTOS). Viitattu 13.11.2015
http://www.electronicshub.org/real-time-operating-system-rtos/

Hohpe, Gregor & Woolf, Bobby. 2009. Enterprise Integration Patterns. Boston: Addison Wesley. Pdf-dokumentti versio. Viitattu 23.8.2015
http://smitsdj.home.xs4all.nl/temp/Addison%20Wesley%20-%20Enterprise%20Integration%20Patterns%20-%20Designing,%20Building%20And%20Deploying%20Messaging%20Solutions%20-%20With%20Notes.pdf

Jussilainen, Mikko. 2015. Rest-pohjaisen web servicen kehittäminen. Pdf-dokumentti. Viitattu 27.12.2015
https://www.theseus.fi/bitstream/handle/10024/102841/Jussilainen_Mikko.pdf?sequence=1

Kava, Kimmo. 2014. Magento-Moodle-integraatio. Pdf-dokumentti. Viitattu 3.10.2015
https://www.theseus.fi/bitstream/handle/10024/81828/Kava_Kimmo.pdf?sequence=1

Kekäläinen, Marko. 2010. Agilefant-projektihallintatyökalun SOA-rajapintatoteutus ja integrointi Oikeat Oliot Oy:n tuntukirjanpitojärjestelmään. Pdf-dokumentti. Viitattu 5.12.2015
http://www.theseus.fi/bitstream/handle/10024/14249/Kekalainen_Marko.pdf?sequence=1

Konttaniemi, Eero. 2015. Palveluväylä. Blogi-kirjoitus. Viitattu 9.11.2015
http://esuomi.fi/palveluvaylan-toukotyot/

Linthicum, David S. 1999. Enterprise Application Integration. Boston: Addison Wesley.

Linthicum, David S. 2003. Next Generation Application Integration. Boston: Addison Wesley. Pdf-dokumentti versio. Viitattu 13.11.2015
https://aihtcsestudents.files.wordpress.com/2014/07/david_s-_linthicum_next_generation_application_ibookzz-org.pdf

Mueller, John. 2013. Understanding soap and rest basics. Blogi-kirjoitus. Viitattu 2.6.2016
http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/

Tahvanainen, Aki. 2012. Järjestelmäintegraatio terveydenhuoltoalalla. Pdf-dokumentti. Viitattu 7.9.2015
https://publications.theseus.fi/bitstream/handle/10024/53544/opinnayte-tyo_aki_tahvanainen.pdf?sequence=1