
# Front-end ohjelmistokehittäjäkonsulttina eläkevakuutusyhtiössä

Henri Väisänen

Haaga-Helia ammattikorkeakoulu
Tradenomi, tietojenkäsittely
Amk-opinnäytetyö
2022

## Tiivistelmä

**Tekijä(t)**
Henri Väisänen

**Tutkinto**
Tradenomi, tietojenkäsittely

**Raportin/Opinnäytetyön nimi**
Front-end ohjelmistokehittäjäkonsulttina eläkevakuutusyhtiössä

**Sivu- ja liitesivumäärä**
50 + 0

Konsultin asemassa työskenteleminen ohjelmistokehittäjänä vaatii uuden oppimista ja omaksumista nopeasti. Tämä pätee erityisesti projektien alkutaipaleella, kun täytyy omaksua nopeasti ennalta kokonaan tuntemattomia teknologioita, viitekehyksiä tai toimintamalleja osana ohjelmistokehitystyötä.

Tässä päiväkirjamuotoisessa opinnäytetyössä kirjoitetaan oleellisilta osin päivittäisistä front-end ohjelmistokehittäjän työtehtävistä ja työskentelytavoista. Opinnäytetyön tekijä työskentelee kokopäiväisessä toistaiseksi voimassa olevassa työsuhteessa Tietoevryllä suomalaisen eläkevakuutusyhtiön asiakkuudessa konsultoivana front-end ohjelmistokehittäjänä. Työn päiväkirjaosuutta on kirjoitettu 7.3.2022 ja 6.5.2022 välisellä ajalla.

Opinnäytetyöhön viikkoanalyyseihin on valittu viikoittain työnkuvaan liittyvä aihe, jonka kirjallisuuteen, tutkimuksiin tai tekniseen dokumentaatioon on perehdytty sekä analysoitu eri näkökulmista. Tavoitteena on kiihdyttää tekijän oppimista projektissa käytössä olevissa teknologioissa sekä perehtyä aiheisiin ja näkökulmiin, tekijän oman sekä koko kehitystiimin työskentelyn tehostamiseksi.

Opinnäytetyön aikana tietämystä karttui paljon projektissa käytössä olevista teknologioista, kuten Angularista ja RxJS:stä. Opinnäytetyön tekijän taidot lukea alaan liittyvää kirjallisuutta, kuten tutkimuksia tai teknistä dokumentaatiota kehittyi myös paremmaksi. Lisäksi ohjelmistokehityksen alalla suomessa ja muissa maissa vallitsevat juridisesti sitovat vaatimukset, kuten saavutettavuusvaatimukset nousivat opinnäytetyöprosessin aikana esille. Yksi opinnäytetyössä tehtyjä havaintoja on esimerkiksi saavutettavuuteen liittyvät testaustavat. Niiden kustannustehokkuuteen liittyvään tutkimukseen perehtyminen ja tekijä omien havaintojen analysointi osoitti esimerkiksi ruudunlukijan ja näppäimistönavigaation erityisen nopeiksi, mutta tehokkaiksi saavutettavuustestauksen menetelmiksi.

**Asiasanat**
Angular, RxJS, Scrum, saavutettavuus, design system, konsultointi

## Sisällys

# 1 Johdanto

Tässä päiväkirjaopinnäytetyössä seurataan ja analysoidaan päivittäin työtehtäviäni ohjelmistokehityksen konsulttina. Opinnäytetyön päiväkirjaosuus sijoittuu 7.3.2022 ja 2.5.2022 väliselle ajalle. Työnantajani on Tietoevry, suuri kansainvälinen ohjelmisto- ja palveluyhtiö, jolla on sekä yksityisen, että julkisen sektorin asiakkaita ympäri maapalloa.

Työskentelen itse opinnäytetyön tekohetkellä erään suomalaisen eläkevakuutusyhtiön asiakkuudessa nuorempana front-end kehittäjänä, jossa kehitän uutta verkkoasiointialustaa. Tiimissä projektijohtaja, tuoteomistaja ja Scrum Master työskentelevät asiakasyrityksessä ja ohjelmistokehittäjät ovat konsultteja eri yrityksistä. Omassa työskentelyssä projektin tärkeimpiä teknologioita ovat Angular, TypeScript, RxJS ja Node. Lisäksi projektissa on käytössä palvelinpäässä Java ja sen Spring-viitekehys.

Tarvittavaa osaamista työtehtävässäni on hyvä ymmärrys internet-sivujen perusosista HTML:stä, CSS:stä ja JavaScriptistä. Lisäksi on tärkeää tietää riittävällä tasolla, miten tiedonsiirto toimii verkossa HTTP-protokollan kautta. Näiden lisäksi minulla on ennen nykyisessä projektissa aloittamista ollut kokemusta React-käyttöliittymäkirjastosta, mikä helpotti Angular-käyttöliittymäviitekehyksen oppimista tätä projektia varten.

Tämä projekti on ensimmäinen, jossa olen käyttänyt Angularia, TypeScriptiä ja RxJS:ää ja teknologioiden osalta odotan, että suurin osa kehittymisestä tapahtuu niissä. Tavoitteena on erityisesti kehittyä reaktiivisessa ohjelmoinnissa, asynkronisessa tiedon hakemisessa ja tapahtumakäsittelyssä RxJS:n avulla. Lisäksi odotan oppivani saavutettavuuden toteuttamisesta digitaalisissa palveluissa, konsultin ja asiakkaan välisestä kommunikoinnista ja työskentelystä ketteriä menetelmiä hyödyntävässä tiimissä. Saavutettavuus on myös projektissa erittäin tärkeää, sillä asiakasta sitoo digipalvelulaki (Aluehallintovirasto, no date a). Se vaatii tiettyjen saavutettavuusvaatimusten täyttämistä (Aluehallintovirasto, no date c, no date d). Projektissa on käytössä myös asiakasorganisaation oma design systeemi, joten myös se aihe tulee varmasti tutummaksi.

Taulukko 1. Peittomatriisi päiväkirjaopinnäytetyön tekstinsisäisistä kytköksistä

| Oman ammatillisen kehittymisen tavoitteet | Seurantaviikko |
|---|---|
| RxJS | 1, 2 |
| Angular | 1, 3 |
| Scrum | 4 |
| Saavutettavuus | 5, 6 |
| Työtyytyväisyys | 7 |
| Design system | 8 |

Työhön liittyvät keskeiset käsitteet:

Angular-käyttöliittymäviitekehys

Asiakasprojektissa on käytössä Angular-käyttöliittymäviitekehys. Angular-käyttöliittymäviitekehys on luotu TypeScriptillä ja se kokoaa yhteen useita kirjastoja esimerkiksi reititykseen, lomakkeiden hallintaan ja palvelimen kanssa kommunikointiin (Google, no date k).

TypeScript

TypeScript on Microsoftin kehittämä JavaScript ohjelmointikielen vahvasti tyypitetty ylijoukko. Se siis lisää JavaScriptin päälle toiminnallisuuksia, kuten esimerkiksi luetteltuja tietotyyppejä (Microsoft, no date).

RxJS

Reactive Extensions for JavaScript. Se on luotu helpottamaan reaktiivisen ohjelmointiparadigman toteuttamista JavaScriptillä. RxJS-kirjasto on yksi Angularin sisältämistä kirjastoista. Se on rakennettu suoraan kiinni Angularin tarjoamaan HttpClient-palveluluokkaan niin, että kaikki sillä tehdyt HTTP-kutsut palauttavat aina RxJS:n tarjoaman tarkkailtava tyyppisen olion (Google, no date f, no date i; Cory Rylan, 2017; Sergi Mansilla, 2018).

Observable

RxJS:ssä observable tai suomeksi tarkkailtava on tietotyyppi, joka kuvaa tulevien arvojen virtaa, joita voidaan niiden saapuessa käsitellä ja joihin voidaan reagoida (Google, no date n).

HTTP

Hypertext Transfer Protocol. Se on esimerkiksi verkkoselainten ja verkkopalvelinten välisen kommunikaation mahdollistava protokolla. HTTP-kutsussa asiakasohjelmisto tekee pyynnön palvelimelle ja odottaa palvelimelta vastausta (Mozilla, no date b).

Koodikatselmointi

Koodikatselmoinnit tarkoittavat koodin tarkistusta jonkun muun, kuin sen kirjoittajan toimesta. Asiakasorganisaatiossa kehittäjät työskentelevät omissa kehityshaaroissa, joihin tehdyt koodimuutokset käyvät läpi koodikatselmoinnin ennen kuin muutokset voidaan siirtää päähaaraan. Koodin käy läpi ainakin yksi kehittäjä. Asiakasorganisaatiossa on linjattu, että koodikatselmoinnissa annetut kommentit tai huomautukset eivät välttämättä tarkoita, että koodiin täytyy tehdä muutoksia. Tarkoituksena on madaltaa dialogin käymisen kynnystä (Sadowski *et al.*, 2018).

Käännösavain

Käännösavaimilla lokalisoidaan sovelluksia. Käännösavain on merkkijono, jolla haetaan oikea tekstiresurssi käyttäjän valitsemalle kielelle. Esimerkiksi kuvitteellisella ”lähetä”-nappulalla voisi olla käännösavain ”application_button_send”, joka saa käyttäjän valitseman kielen mukaisen arvon. Esimerkiksi ”Lähetä hakemus” jos kieleksi olisi valittu suomi (Lokalise, no date).

Saavutettavuus

Saavutettavuus tarkoittaa digitaalisten palvelujen kontekstissa sitä, että kaikilla ihmisillä olisi yhtäläiset mahdollisuudet käyttää digitaalisia palveluita mahdollisista erilaisista rajoitteista riippumatta (Aluehallintovirasto, no date e).

Design system

Design system tai design systeemi on joukko standardeja, jotka määrittelevät visuaaliset ja tekniset ohjeet organisaation brändin toteuttamiseen käyttöliittymissä. Yleisesti ottaen design systeemit sisältävät ainakin dokumentaation standardisoiduista visuaalisista tyyleistä, käyttöliittymäkomponenttikirjaston sekä käyttöliittymäkomponenttikirjaston teknisen dokumentaation (Fessenden, 2021).

API

Application programming interface, eli ohjelmointirajapinta. Ohjelmointirajapinta on joukko määrittelyjä, sille miten rajapinnan ulkoiset palvelut voivat kommunikoida rajapinnan toteuttavan ohjelmiston kanssa (Red Hat, no date).

Story points

Story points eli tarinapisteet ovat abstrakti yksikkö, jolla arvioidaan käyttäjätarinan toteuttamiseen vaadittava vaiva. Asiakkaan kehitystiimeissä on käytössä tarinapisteet käyttäjätarinoiden vaatiman vaivan arvioinnissa (Radigan, no date).

# 2 Lähtötilanteen kuvaus

Olen työskennellyt nykyisessä asiakasprojektissa ennen opinnäytetyöprojektin aloittamista noin kuusi kuukautta. Sinä aikana työskentelyni on jo kehittynyt sille tasolle, että päivittäiset tehtävät sujuvat ilman suurempaa ulkoista ohjausta.

## 2.1 Oman nykyisen työn analysointi

Työtehtäviini kuuluu web-ohjelmointi käyttäen Angular-käyttöliittymäviitekehystä ja sen sisältämiä teknologioita, kuten TypeScript ja RxJS. Työtehtävistä selviytymiseen tarvitaan ainakin hyvä tietämys HTML:stä, CSS:stä ja JavaScriptistä. Kokemus jostain JavaScript käyttöliittymäkehyksestä auttaa omaksumaan Angularin nopeammin. On myös tärkeää ymmärtää riittävällä tasolla, miten tiedonsiirto HTTP-protokollan kautta toimii.

Osaamiseni on hankittu käymällä koulutusohjelmaani sisältyvät oleelliset opintojaksot, kuten ohjelmistoprojektit 1 ja 2 sekä ohjelmistokehityksen teknologiat. Lisäksi olen suorittanut joitain koulutusohjelman ulkopuolisia opintoja, esimerkiksi Helsingin Yliopiston Ohjelmoinnin syventävän jatkokurssin. Lisäksi olen kerryttänyt tietämystä seuraamalla säännöllisesti alan asiantuntijoiden luomaa opetussisältöä eri sosiaalisen median palveluissa.

Saamani palautteen perusteella ammatillinen kehittymiseni on lähtenyt hyvin vahvasti liikkeelle. Selviydyn päivittäisistä työtehtävistä pääasiassa itsenäisesti, ilman suurempaa ulkoista ohjaamista.

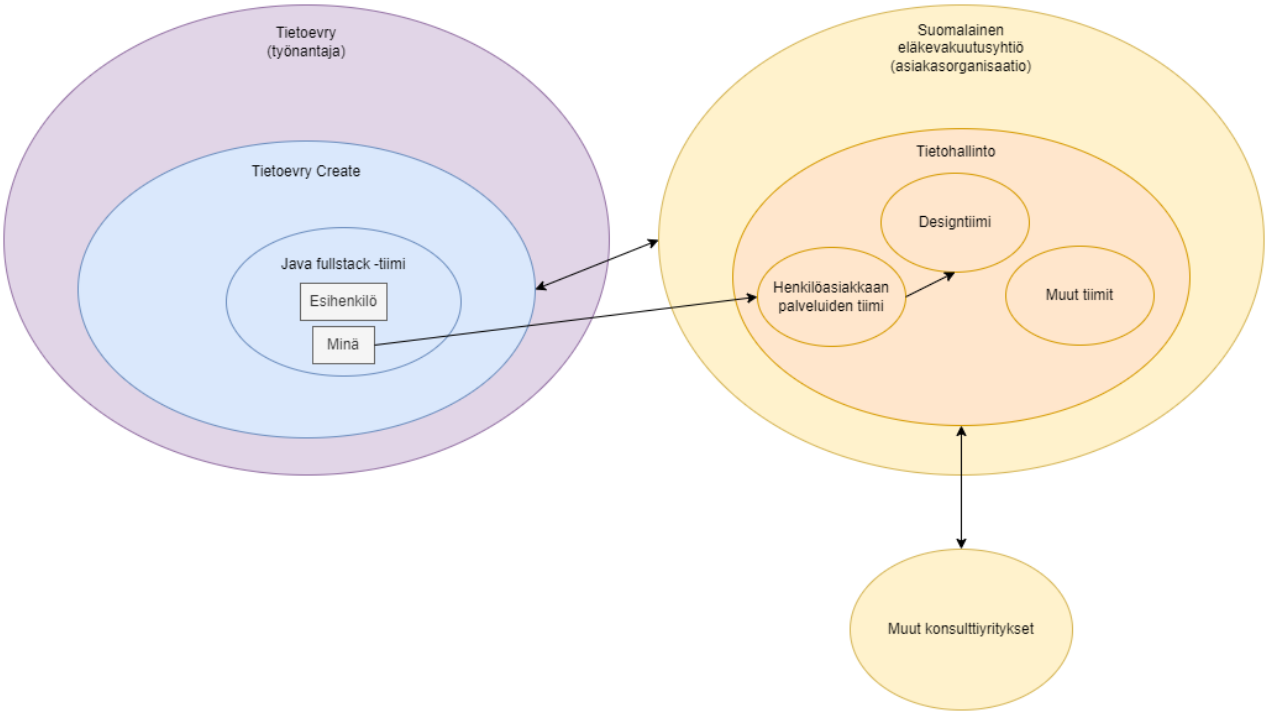
Jatkossa koen tärkeäksi panostaa projekteissa tärkeiden uusien teknologien nopeaan omaksumiseen ja parhaiden toimintatapojen sisäistämiseen. Tällä hetkellä se tarkoittaa esimerkiksi reaktiivisen ohjelmointiparadigman parempaa sisäistämistä ja RxJS-ohjelmointikirjaston laajempaa hallitsemista. Angular-käyttöliittymäviitekehyksessä yksi tärkeä toimintatapa on hyödyntää aina mahdollisimman reaktiivista syntaksia, kun tuodaan palvelimelta tietoa näkyviin käyttöliittymälle. Tähän Angular tarjoaa paljon työkaluja, joiden osaaminen on tärkeää. Osaan näistä työkaluista paneudun myös opinnäytetyön viikkoanalyyseissa.

Annetuista tasoista, aloitteleva toimija, taitava suoriutuja ja kokenut asiantuntija, koen, että omaa osaamistasoani kuvaa parhaiten taitava suoriutuja. Tätä perustelen sillä, että päivittäiset työtehtävät sujuvat itsenäisesti ja olen kehittänyt monia kehitettävän ohjelmiston osia itsenäisesti ja ne ovat edistyneet odotetusti. Olen myös saanut asiakkaan organisaatiosta erinomaista palautetta.

## 2.2 Sidosryhmien esittely

Projektissa sidosryhmiä ovat työnantajani Tietoevry ja asiakasorganisaationa toimiva suomalainen eläkevakuutusyhtiö X. Asiakasorganisaatio X:n sisällä sidosryhminä on oma eli henkilöasiakkaan palveluita kehittävä tiimi, palvelun muiden osien omat kehitystiimit sekä designtiimi. Toimin designtiimissä henkilöasiakkaan palveluiden tiimin edustajana. Se tarkoittaa, että osallistun designtiimin viikoittaiseen palaveriin ja saan toteutettavaksi design systeemin jatkokehittämiseen liittyviä tehtäviä.

Lisäksi sidosryhminä toimii asiakasorganisaation X käyttämät freelancer-konsultit ja muut konsultointiyritykset ja joiden työntekijöitä työskentelee myös minun kanssani samassa tiimissä. Sidosryhmän muodostavat myös asiakasorganisaation verkkopalvelua käyttävät loppukäyttäjät, kuten yrittäjät, joilla on asiakasorganisaatio X:n kautta hankittu henkilökohtainen yrittäjän eläkevakuutus, yritykset, jotka tarjoavat työntekijöilleen eläkevakuutuksen asiakasorganisaatio X:n kautta, sekä erilaisia eläkkeitä hakevat tai omaa eläkekertymää tarkistavat yksityishenkilöt.




Kuvio 1: Sidosryhmät

## 2.3 Työpaikan vuorovaikutustilanteet

Työskentely tapahtuu omalta osalta suurimmaksi osaksi kotoa, mutta välillä myös asiakkaan omalla toimistolla tai Tietoevryn toimistolla. Tietoevryn oma ohjeistus on, että työntekijät saavat jatkossakin työskennellä joko kotona tai toimistolla niin sanotun hybridimallin mukaisesti (Vinje,

2021). Konsulttien asiakasorganisaatioissa voi kuitenkin olla erilaisia ohjeistuksia, jolloin niitä täytyy noudattaa. Omassa asiakasorganisaatiossa ei ole toistaiseksi ollut konsulteille ohjeistusta palata toimistolle.

Asiakasorganisaatiossa päivittäisiä vuorovaikutustilanteita ovat ketterän ohjelmistokehityksen seremoniat, kuten päivittäispalaverit, sprintin retrospektiivi ja sprintin suunnittelu. Lisäksi keskustelua käydään Slack-viestintäsovelluksessa. Suurimman osan ollessa nyt ja todennäköisesti jatkossa etätöissä, voi viestintä vain sähköisten välineiden kautta välillä hidastaa hieman työskentelyä, mutta pääasiassa viestintä sujuu etätöissäkin ongelmitta. Toimistolla työskennellessä vuorovaikutustilanteita muodostavat lisäksi erilaiset kahvipöytä- ja lounaskeskustelut. Nämä saattavat myös olla tilanteita, jotka toisaalta hidastavat toimistotyöskentelyä suhteessa etätöihin. Mahdollisia haasteita vuorovaikutustilanteissa saattavat olla esimerkiksi lisätietoja vaativien toiminnallisuuksien havaitseminen riittävän aikaisessa vaiheessa, jotta ei jää tekemisessä tyhjän päälle.

# 3 Seurantajakson raportointi viikkoanalyyseineen

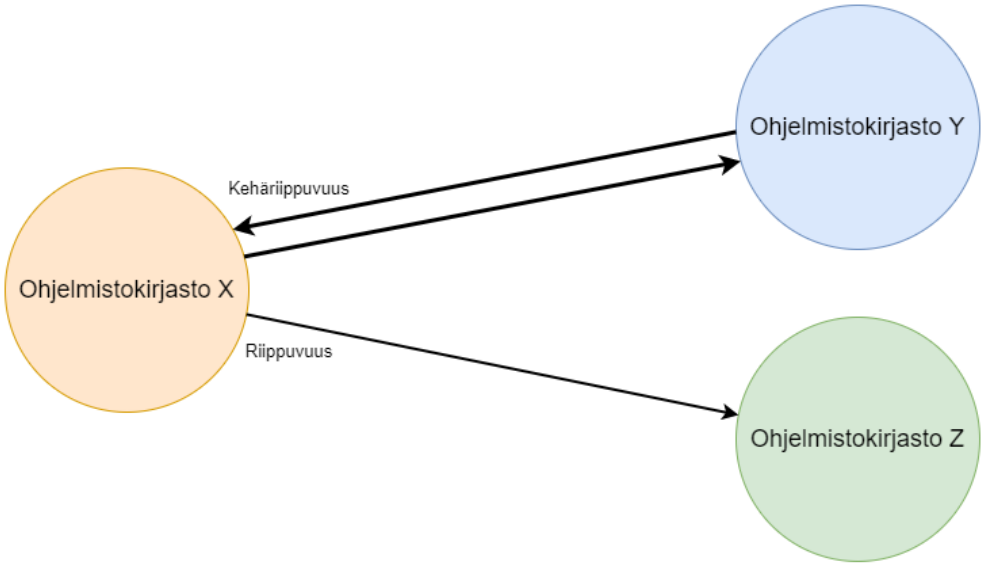
Tässä kappaleessa kerron päivittäisistä työskentelyyn liittyvistä oleellisista aiheista. Lisäksi joka viikon lopuksi on viikkoanalyysi, jossa käsittelen jotain projektiin tai työnkuvaani liittyvää aihetta.

## 3.1 Seurantaviikko 1

Maanantai 7.3.2022

Tänään tavoitteena on aluksi katsoa, että olen varmasti huomioinut viime viikolla luomassani koodikatselmoinnissa nousseet asiat, ennen kuin yhdistän oman kehityshaarani takaisin päähaaraan. Lisäksi aloitan tekemään muutosta kehittämääni viesti- ja asiakirjapalveluun, joka tällä hetkellä hakee viestien ja asiakirjojen metatiedot yhtä aikaa. Tämä tulisi muuttaa niin, että se hakisi ne toisistaan asynkronisesti erillisillä HTTP-kutsuilla. Tämä sen takia, että liitetiedostojen metatietojen lataamisessa kestää yleensä 6–7 sekuntia ja halutaan, että käyttäjä voi nähdä huomattavasti nopeammin ladattavat viestien metatiedot jo ennen sitä.

Päivittäispalaverin jälkeen jäätiin keskustelemaan yksityiskohdista liittyen aiemmin mainitsemaani viesti- ja asiakirjapalveluun. Havaitsin palvelun omassa palveluluokassa kehäriippuvuus virheen. Kehäriippuvuus tarkoittaa tilannetta, missä kaksi ohjelmistoa tai sen osaa ovat toisistaan riippuvaisia (Al-Mutawa *et al.*, 2014). Korjasin ongelman siirtämällä ongelman aiheuttaneet riippuvuudet hierarkiassa alempana olevasta kirjastosta ylempään ja yleiskäyttöisempään.




Kuvio 2: Kehäriippuvuus (Al-Mutawa *et al.*, 2014)

Sain viestien ja liitetiedostojen haun eriytettyä toisistaan melko vaivatta, kuten oletin. Se tosin toi ilmi joitain asioita käyttöliittymällä, mitkä täytyy selvittää seuraavana työpäivänä. Käyttöliittymällä viestit ja asiakirjat ovat siis eri välilehdillä. Täytyy päättää, halutaanko, että välilehteä kuvaavassa käyttöliittymäkomponentissa näkyisi latausanimaatio. Toinen vaihtoehto olisi, että käyttäjä pääsee navigoimaan asiakirjavälilehdelle niin, että välilehden sisällön tilalla on latausikoni, kunnes tiedot on onnistuneesti ladattu.

Tiistai 8.3.2022

Tavoitteena selvittää mikä on myös eilisessä päiväkirjamerkinnässä mainittu latausikonien haluttu toimintatapa. Lisäksi täytyy selvittää minkä toiminnallisuuden otan toteutettavaksi sen jälkeen ja mahdollisesti aloittaa sitä.

Latausikonin näyttäminen oli yksinkertaista eilen tehtyjen HTTP-kutsujen muutoksen jälkeen. Kuuntelemalla asiakirja HTTP-kutsun sisältävän RxJS tarkkailtavan tilaa, pystyin Angularin tarjoamalla async-putkella ja ngIf-direktiivillä näyttämään latausikonin sillä välin, kun tarkkailtavan sisältämä kutsu ei ollut valmis ja sen valmistuessa näyttämään halutut sisällöt. Nyt alkuun latausikoni näkyy sovelluksen asiakirja puolen välilehdessä niin, että välilehti on ”disabled” tilassa eli sille välilehdelle ei tietojen latauksen ollessa kesken pääse ollenkaan. Huomisen päivittäispalaverissa keskustellaan lisää, mitä halutaan latauksen aikana olevan käyttäjälle näkyvissä.

Lisäksi tutustuin tänään siihen, mitä muutoksia tarvitsee tehdä, jotta saan päivitetyn liitekomponentin käyttöön viesti- ja asiakirjapalvelun uusi viesti- ja lähetä liite -sivuilla. Liitekomponentti päivitettiin niin, että jatkossa sillä liitetyt liitteet lähetetään palvelimelle sitä mukaan, kun liitteitä lähetetään. Liite komponentti sallii usean liitteen valitsemisen kerralla, mutta palvelin pystyy vastaanottamaan vain yhden liitteen per HTTP-kutsu. Tästä johtuen koodia täytyy päivittää niin, että se lähettää liitteet yksitellen, vaikka niitä valittaisiin kerralla useampi kuin yksi. Tällä hetkellä kirjoittamani koodi ei vielä huomioi tilannetta, jossa käyttäjä valitsee kerralla useamman kuin yhden liitteen.

Keskiviikko 9.3.2022

Tänään on designtiimin viikkopalaveri. Siellä suunnitellaan yhtenäistä ratkaisua, miten lataustila näkyy viesti- ja asiakirjapalvelun käyttöliittymällä, kun välilehtikomponentin sisältöä ladataan. Olen siis oman tiimini edustaja designtiimissä eli osallistun designtiimin viikkopalaveriin ja teen välillä joitain designtiimin tehtäviä liittyen käytössä olevaan yhteiseen käyttöliittymäkirjastoon. Mikäli sovittu ratkaisu eroaa eilen tekemästäni, joudun tekemään siihen pieniä muutoksia. Lisäksi ajan salliessa aion paneutua taas päivitetyn liitekomponentin käyttöönottoon.

Lataustilan osalta päätettiin, että asiakirjalistan päällä näkyy latausanimaatio ja käyttäjä päästetään navigoimaan asiakirjavälilehdelle. Näin käyttäjä tietää paremmin, minkä asian latausta hän tarkalleen ottaen odottaa.

Päivällä oli myös kokous, jossa jaettiin viestisovellukseen liittyviä tehtäviä minulle ja toisen tiimin konsultille. Ehdin myös aloittaa oman koodini yhteensovittamisen uuden liitekomponentin kanssa. Se vaati komponentin palauttaman liitelistan liitteiden lähettämistä yksitellen JavaScript listojen tarjoaman forEach-funktion avulla. Se siis kutsuu jokaiselle listan tietoalkiolle forEach-funktion parametrina ottavan vastakutsufunktion (Mozilla, no date a). Liitteiden järjestyksellä on väliä, joten tarkistin, että forEach käy ennustettavissa olevassa järjestyksessä saamansa listan kohdat. Uusi liitekomponentti noudattaa vielä aiempaa paremmin yhden vastuun ohjesääntöä (single responsibility principle), se siis nyt keskittyy vielä aiempaa paremmin pelkästään tiedon näyttämiseen ja mahdollisimman vähän liitteiden tilan hallintaan (Wang and Zhou, 2012). Tästä johtuen joudun nyt pitämään oman komponenttini koodissa listaa onnistuneesti lähetetyistä liitteistä ja tarjota ne oikeassa muodossa takaisin liitekomponentille, jotta se osaa näyttää listan onnistuneesti liitettyjä liitteitä käyttöliittymällä. Tämän toteuttaminen jäi kuitenkin tänään vielä kesken. Liitteiden lähettäminen siis onnistuu jo, mutta huomenna täytyy vielä saada liitteet oikeassa muodossa talteen omaan komponenttiin, jotta liitekomponentti toimii halutusti.

Torstai 10.3.2022

Päivän tavoite on jatkaa mihin eilen jäin ja saada liitekomponentti toimimaan lähetä liite -sivun komponentin kanssa. Osittain samaa logiikkaa voi hyödyntää myös viestin lähetys sivulla, joten jatkan siihen, kun saan lähetä liite -sivun toimimaan. Tänään on lisäksi Tietoevryn uuden Create organisaation software engineering yksikön aloitustapaaminen sekä projektin sprintin katselmointi.

Sprintin katselmoinnissa totesimme, että monia sprintin tavoitteita saatiin toteutettua, mutta sprintille oli myös tullut odottamattomasti lisää tehtävää, mikä esti tiimiä saamasta sprintin kaikkia tavoitteita tehtyä. Kokoustamisen lisäksi ehdin myös koodata ja liitekomponentti toimii nyt lähetä liite -sivun komponentin kanssa yhteen. Käytännössä lähetä liite -sivu pitää nyt kirjaa onnistuneesti lähetetyistä komponenteista ja virheiden sattuessa käyttäjälle näytetään heräteviesti käyttöliittymän alalaidassa. Pohdin myös mahdollisuuksia hajauttaa osaa lähetä liite -sivun koodista erilliseen Angular-palveluun, mikäli se olisi riittävän irrallista itse komponentin logiikasta ja olisi mahdollisuus välttää päällekkäistä koodia useassa eri komponentissa viesti- ja asiakirjapalvelussa. Työkaverin kanssa käydyn keskustelun pohjalta lähdin hajottamaan jo kirjoitettuja funktioita vielä pienemmiksi palasiksi, mikä helpotti luettavuutta, mutta samalla huomasin, että niitä ei välttämättä ole kannattavaa siirtää Angular-palveluun. Funktiot ovat riippuvaisia monista käyttävän komponentin omista muuttujista, että niiden kaikkien antaminen funktion attribuutteina ei luultavasti ole kannattavaa.

Perjantai 11.3.2022

Tänään aion ottaa uuden liitekomponentin käyttöön myös uusi viesti –sivulla. Seremonioiden osalta tänään on päivittäispalaverin sijasta sprintin suunnittelu.

Tämä päivä menikin seremonioiden ulkopuolella vielä lähetä liite -sivun parissa. Eilinen koodi oli jäänyt vielä vähän keskeneräiseksi lähinnä yhden käännösavaimen osalta, ja sen jälkeen laitoin koodin katselmointiin ja korjasin siellä tulleen yhden huomion. Se liittyi if else-rakenteessa olleeseen return-lausekkeeseen, joka oli tässä tapauksessa turha, sillä kaikki tapaukset tuli käsiteltyä if else -rakenteen sisällä eikä funktion tarvinnut refaktoroinnin jälkeen enää palauttaa mitään arvoa. Jos koodi siis joutuu ensimmäisen if-lausekkeen sisälle, funktion ajaminen lopetetaan siihen. Jos koodi menee else-lausekkeen sisälle, se ajaa sen sisäisen koodin ja jatkaa vielä funktion loppuosan koodiin. Lisäksi keskustelin tänään palvelinkehittäjän kanssa, minkälainen API-kutsu tehdään viestin varsinaista lähettämistä varten.

Liitteiden liittäminen toimii siis niin, että ensimmäisen liitteen kohdalla sovellus luo uuden viestiketjun ja viestin sen sisälle, jonka tunnisteet se palauttaa frontend-sovellukselle, joka sitten lähettää liitetyt liitteet saman tien palvelimelle. Nämä liitteet eivät kuitenkaan lähde palvelimelta eteenpäin ennen kuin käyttäjä painaa käyttöliittymällä lähetä nappia. Suunnittelimme siis, minkälainen API-kutsu tehdään, kun lähetä painiketta painetaan ja palvelin voi alkaa prosessoida liitteitä eteenpäin. Sovittiin ratkaisu, jonka avulla samaa API-kutsua voi käyttää lähetä liite -sivulla sekä uusi viesti -sivulla niin, että API osoitteeseen voi joko tehdä tyhjän POST-kutsun, jos viestiin tulee pelkkiä liitteitä (lähetä liite -sivu) tai POST-kutsun, jonka mukana tulee viestin tekstisisältö tekstijonona (uusi viesti –sivu)

Viikkoanalyysi

Tämän viikon viikkoanalyysissa tarkastelen mitä työkaluja Angular tarjoaa yhdessä RxJS:n kanssa reaktiivisen ohjelmointiparadigman toteuttamiseen ja tarkalleen ottaen tarkkailtava tietotyyppiin käärittyjen HTTP-kutsujen hallintaan, sekä mitkä ovat parhaat toimintatavat niiden käyttämiseen Angularissa.

Angularin sisältämä RxJS-kirjasto sisältää paljon erilaisia apufunktioita, joilla käsitellä tarkkailtava tyyppistä dataa. Tarkkailtavan sisältämät arvot, kuten HTTP-kutsun palauttama data, ei palaudu ennen kuin tarkkailtavan subscribe-funktiota kutsutaan (Google, no date h). Yleisin tarve erityisesti HTTP-kutsujen tapauksessa on, että HTTP-kutsun palauttama data halutaan sellaisenaan suoraan näkyviin käyttöliittymälle. Tämä onnistuu subscribe-funktion avulla ja se vastaa samaa asiaa, kuin

JavaScriptin omissa promiseissa then-funktiota, jossa käsitellään promisen palauttama data. Olenkin huomannut, että kun on kehittäjänä tottunut käyttämään JavaScriptin promiseja, voi helposti olla hyödyntämättä RxJS tarkkailtavan mahdollistamia ominaisuuksia.

Angular Universityn blogipostauksessa “Angular Reactive Templates with ngIf and the Async Pipe” (2022), käsitellään erilaisia tapoja mitä Angular tarjoaa reaktiivisen ohjelmointiparadigman toteuttamiseen Angular-komponenttien HTML-sivupohjatiedostoissa. Tilanteissa, joissa halutaan tarkkailtavan palauttama data suoraan käyttöliittymälle, on parasta olla kutsumatta subsrcibe-funktiota itse ja jättää tarkkailtavan käsittely Angularin tarjoaman async-putken varaan. Angular-putket ovat erityisiä luokkia, jotka määrittelevät funktion, joka ottaa dataa sisään ja muuttaa sen eri muotoon näytettäväksi käyttöliittymällä (Google, no date e, no date j). Angularin async-putki hoitaa itsenäisesti tarkkailtavan subscribe ja unsubscribe-funktioiden kutsumisen ja (Google, no date c). Unsubscribe-funktion kutsu on erityisen tärkeää pitkäikäisten tarkkailtava olioiden kanssa muistivuodon välttämiseksi. Käytännössä voi ajatella, että async-putki avaa tarkkailtavan sisältämän datan luettavaan muotoon, esimerkiksi JavaScript olioksi. Alla oleva koodi asettaisi HTML:n <p> tagien sisälle kuvitteellisen personDataObservablen sisällön.

```
1         <p>{{ personDataObservable | async | json }}</p>
```

Json-putki on toinen Angularin tarjoama putki, joka muuttaa minkä vain arvon, tässä tapauksessa async-putken avaaman tarkkailtavan sisältämän JavaScript olion ja muuttaa sen json muotoiseksi merkkijonoksi (Google, no date g). Sitä tarvitaan, jotta JavaScript olion arvo voidaan näyttää käyttöliittymällä. Json eli JavaScript Object Notation on tiedonsiirtoformaatti, jolla voi kuvailla kaikille ohjelmointikielille yhteisiä universaaleja tietorakenteita (*JSON*, no date). Nämä tietorakenteet ovat listat sekä oliot.

Yleensä todennäköisesti halutaan tarkkailtavan sisältävän JavaScript olion eri kentät eri kohtiin käyttöliittymällä. Alla oleva koodi näyttää, miten se on mahdollista aiemman esimerkin syntaksia hyödyntämällä.

```
1         <p>{{ (personDataObservable | async |)?.firstName }}</p>
2         <p>{{ (personDataObservable | async |)?.lastName }}</p>
3         <p>{{ (personDataObservable | async |)?.email }}</p>
```

Näin saadaan tarkkailtavan sisältämän JavaScript olion sisältämien erillisten kenttien arvot avattua käyttöliittymälle näkyviin omiin kohtiinsa. Hyvää tässä on se, että on käytetty async-putkea, jotta voidaan olla huolehtimatta muistivuodoista. Tämä ei kuitenkaan ole suositeltava tapa käyttää

async-putkea, sillä nyt käyttäjän etunimeä, sukunimeä ja sähköpostia varten tehdään erilliset HTTP-kutsut, kun async-putkea käytetään erikseen jokaisessa kentässä. Seuraavaksi katsotaan, miten sen voisi välttää.

```
1         <div *ngIf=”personDataObservable | async as personData”>
2             <p>{{ (personData.firstName }}</p>
3             <p>{{ (personData.lastName }}</p>
4             <p>{{ (personData.email }}</p>
5         </div>
```

Tässä ympäröitiin aiempi HTML-koodi HTML:n  tageilla, jonka kanssa käytettiin Angularin ngIf-direktiiviä, joka näyttää direktiivin sisältämän elementin vain, jos sen sisältämä ehtolause palauttaa truthy arvon. Truthy arvo tarkoittaa siis arvoa, jonka JavaScript katsoo totuusarvoa odottavassa kontekstissa vastaavan arvoa true (Mozilla, no date c). Lisäksi ngIf tukee ”as” syntaksia, jolla saadaan ehtolauseen sisällä olevan async-putken palauttama JavaScript olio tallennettua HTML-elementin lapsielementeillä näkyvään muuttujaan ”personData”. Tämä mahdollistaa sen, että tarkkailtavaa kutsutaan vain kerran, ilman, että sovelluksen reaktiivisuus kärsisi. Itse esimerkiksi Angular polkuni alussa tallensin aina tarkkailtavan palauttaman JavaScript olion Angular-komponentin TypeScript koodissa muuttujaan, mikä tarkoitti monta riviä huonommin luettavissa olevaa vähemmän reaktiivista koodia, joka voisi pitkäikäisen tarkkailtavan kanssa vaarantaa sovelluksen muistivuodoille.

### 3.2 Seurantaviikko 2

Maanantai 14.3.2022

Tänään kehitän testipalvelimelle toiminnallisuuden, joka näyttää käyttöliittymän koodin näkökulmasta samalta, kuin perjantaina palvelinkehittäjän kanssa sovitut API-kutsut viestin lähettämiselle ja liitteiden poistamiselle. Sitten teen Angular sovelluksen viesti- ja asiakirjapalvelun omaan palveluluokkaan uudet metodit, joilla kutsutaan näitä uusia rajapintoja. Lähetä liite -sivulla täytyy sitten vielä ottaa käyttöön nuo uudet metodit, jotta koko sivu tulee sitten toimimaan oikeaa palvelinta vasten odotetusti. Tämän jälkeen viime viikolla lähetä liite -sivulla käyttöön otettu liitekomponentti täytyy integroida uusi viesti- ja viestiketju -sivuille sekä viimeistellä viestin lähettämisen flow niillä sivuilla. Tänään on myös sprintin jälkeinen demotilaisuus, jossa asiakasorganisaatio pääsee näkemään uusia tuotoksia. Minulla ei tällä kertaa ole siellä näytettävää.

Tänään huomattiin vielä pieniä epäjohdonmukaisuuksia viesti- ja asiakirjapalvelun rajapinnassa. Pidettiin palvelinkehittäjän kanssa pikapalaveri ja nyt pitäisi taas olla suunnitelmat selvät. Refaktoroin aiempia rajapinnan tarjoamia metodeja ja niitä käyttävän Angular-palveluluokan metodeja. Nyt

Angular-sovellus voi käyttää siis kaikissa viestin lähettämistilanteissa samoja metodeja oli kyse sitten viestiketjuun vastaamisesta, liitteiden lähettämisestä tai uuden viestin lähettämisestä. Aiemmin olisi ollut tilanteita, joissa Angular sovelluksella ei ole tiedossa sen lähettämän viestin tunnistetta, kuten viestiketjuun vastatessa. Refaktoroitiin siis niin, että kaikki viestien ja liitteiden lähettämisen metodit vaativat sekä viestin, että viestiketjun tunnisteet, mutta Angular sovellus hakee uuden viestin tai viestiketjun tunnisteet tarpeen mukaan tilanteissa, joissa jompikumpi ei ole vielä tiedossa. Tämä yksinkertaisti rajapintaa huomattavasti.

Tiistai 15.3.2022

Tänään otan käyttöön uudet metodit rajapinnan käyttämiseen lähetä liite -sivun komponentissa sekä aloitan uusi viesti -sivulla refaktoroinnin uutta liitekomponenttia varten.

Päivällä tulikin vielä yksi viesti ja asiakirjapalveluun liittyvä suunnittelupalaveri, jossa oli nyt kaikki kehittäjäosapuolet paikalla. Palvelussa yhdistyy useita aiemmin erillisinä olleita palveluita, joten katsottiin, miten esimerkiksi tietotyyppejä voitaisiin yhdistää niin, että ne sopivat kaikkien taustapalveluiden kanssa yhteen. Paransin testipalvelimelle useita metodeja niin, että testipalvelin vastaa nyt tarkemmin oikeaa tietokantaa vasten pyörivää myöhemmin tuotantoon menevää palvelinta. Käytännössä se siis pitää nyt välimuistissa uudet viestit, viestiketjut sekä liitteet eikä vain vastaa staattisesti Angular sovelluksen odottamalla tavalla. Eiliset rajapintauudistukset tekivät tästä riittävän nopeaa toteuttaa myös testipalvelimelle. Sitten refaktoroin Angular-sovelluksen viestiketju-sivun koodia toimimaan uuden rajapintalogiikan kanssa ja sainkin sen toimimaan. Töitä tosin jäi vielä siihenkin siltä osin, että käytössä ollut lataustilaa indikoiva tarkkailtava ei enää toiminut. Siitä jatkan huomenna.

Keskiviikko 16.3.2022

Tänään suunnitelmana jatkaa uuden rajapintalogiikan integroimista Angular sovelluksessa.

Eilen mainitsemani latausindikaattorin toimimaan saaminen vaati hieman temppuilua entuudestaan tuntemattomien RxJS-operaattorien kanssa, mutta tässä varmasti oppi paljon. RxJS:n tarjoama switchMap-operaattori mahdollisti uuden rajapintalogiikan kanssa sen, että indikaattoritarkkailtava indikoi jälleen lataustilaa oikein. Lisäksi työtä aiheutti se, miten uudet viestit saa viestin lähetyksen jälkeen reaktiivisesti ladattua niin, että indikaattoritarkkailtava valmistuisi vasta, kun uusimmat viestitkin on ladattu.

Torstai 17.3.2022

Tänään luon viestin lähetys statusta indikoivan tarkkailtavan myös lähetä liite -sivulle. Lisäksi ehdin mahdollisesti aloittaa uuden liitekomponentin sovittamista viestiketjun -sivun vastaa viestiin -kohtaan.  Seremonioiden osalta tänään on päivittäispalaverin lisäksi avoimet asiat.

Avoimia asioita ei pidetty, koska ei ollut käsiteltäviä asioita. Sain viestin lähetys statuksen tarkkailtavan luotua lähetä liite -sivulle. Se vaati jälleen uusien RxJS-operaattorien tutkimista ja loppujen lopuksi RxJS:n forkJoin-operaattori oli sellainen, joka vastasi tarpeeseen. Sillä sain tehtyä listan HTTP-kutsuja, jotka tallensin uuteen tarkkailtavaan, joka sitten sen subscribe-funktiota kutsuessa suorittaa kaikki sen sisälle tallennetut HTTP-kutsut kauniisti järjestyksessä ja palauttaa niiden arvot listana, kun kaikki ne ovat valmiina.

Perjantai 18.3.2022

Tänään aloitan integroimaan uutta liitekomponenttia viestiketju -sivun vastauskenttään.

Se luultavasti tarkoittaa myös jonkin verran koodin uudelleen organisointia palveluluokkiin, kaiken uudelleenkäytettävän koodin osalta.

Tänään oli todella tuottoisa päivä. Sain integroitua uuden liitekomponentin viestiketjusivulle ja toteutettua latausanimaation liitteen latautuessa palvelimelle. Tässä nopeutti se, että “lähetä liite” -sivulla opitut tekniikat olivat nyt hyvin hallussa. Kuten aamulla arvelin, koodia sai myös jonkin verran jaettua sivujen kesken palveluluokan avulla.

Lisäksi täydensin aiemmin tehtyä toiminnallisuutta viestiketjusivulla, missä viestin kohdalla näkyy kunkin viestin mukana tulleet liitteet ja liitteen tyyppi. Nyt liitteet on järjestelty liitetyypeittäin kunkin viestin kohdalla.

Viikkoanalyysi

Tällä viikolla töissä käytin useaan kertaan erilaisia RxJS:n tarjoamia operaattoreita. Operaattorit ovat funktioita, jotka käsittelevät tarkkailtavan palauttamaa dataa ja palauttavat uuden tarkkailtavan (Troncone, no date; Sergi Mansilla, 2018). Operaattorit voi sijoittaa RxJS:ssä tarkkailtavan pipe-funktion sisälle, joka kutsuu operaattoreita järjestyksessä aina tarkkailtavan saadessa uuden arvon.

Törmäsin tällä viikolla monesti tilanteisiin, joissa tuntui tarpeelliselta kutsua tarkkailtavan subscribe-funktion vastakutsufunktion sisällä toisen tarkkailtavan subscribe-funktiota. Tämä mahdollistaa aiemmin valmistuneen tarkkailtavan palauttaman datan käyttämisen osana seuraavan tarkkailtavan operaattoreiden parametreja. Subscribe-funktion kutsuminen subscribe-funktion sisällä on kuitenkin monien kehittäjien mielestä huono tapa, jota tulisi välttää (Gallardo, no date; Billiet, 2018;

Vardanyan, 2019). Perusteena sille käytetään usein sitä, että se altistaa sivuvaikutuksille muualla koodissa ja ei ole reaktiivisen paradigman mukaista.

Olin yllättynyt, miten pystyin tekemään pipe-funktion virran sisällä asioita, joihin luulin tarvitsevani esimerkiksi monta subscribe-funktion kutsua ja virran ulkopuolisia ehtolauseita. Tässä aion siis käydä läpi RxJS:n switchMap-operaattoria, jota käytin tällä viikolla ja miten se vaikutti koodin luettavuuteen ja pituuteen.

SwitchMap mahdollistaa lähdetarkkailtavan palauttaman arvon lähettämisen sisäiselle tarkkailijalle. Se voi eliminoida tarpeen tehdä subscribe-funktion kutsu subcribe-funktion takaisinkutsufunktion sisällä. Esimerkiksi HTTP-kutsuja tehdessä tälle voi olla tarve, jos ensimmäisen HTTP-kutsun palauttamaa arvoa tarvitaan seuraavan HTTP-kutsun tekemiseen. Toinen tapaus missä switchMap-operaattori voi olla hyödyllinen on tilanne, jossa ensimmäisen HTTP-kutsun täytyy valmistua ennen kuin toinen HTTP-kutsu tehdään. Esimerkiksi viestisovelluksessa yhdellä HTTP-kutsulla lähetetään viesti palvelimelle ja toisella HTTP-kutsulla haetaan uusimmat viestit palvelimelta. Jälkimmäistä kutsua ei kuitenkaan kannata tehdä ennen kuin ensimmäinen kutsu on mennyt läpi, jotta uudella viestillä päivitetty lista viestejä palautuu käyttöliittymälle. SwitchMap-funktiolla voidaan siis luoda uusi tarkkailtava, joka tekee useita asioita yhdellä subscribe-funktion kutsulla. Esimerkkinä pseudokoodi, jossa ei käytetä switchMap-operaattoria ja koodi, jossa sitä käytetään.

Ilman switchMap-funktiota.

```
1         function postMessage() {
2         postMessage(content).subscribe(() => {
3                 getMessageThread(threadId).subscribe((messages) => {
4         this.messageThread = messages;
5         })
6         })}
```

SwitchMap-funktio käytössä.

```
1          function postMessage() {
2          return postMessage(content).pipe(
3                  switchMap(getMessageThread(threadId))
4                   )
5          }
```

Jälkimmäisessä esimerkissä olen lisäksi tallentanut switchMap-funktion avulla luodun uuden tarkkailtavan funktion palautusarvoksi, joka yhdessä Angularin async-putken kanssa mahdollistaa datan viemisen käyttöliittymälle ilman, että meidän täytyy kertaakaan kutsua itse subscribe-funktiota.

Ensimmäisessä esimerkissä tarkkailtavia käsitellään kuin perinteisiä promise pohjaisia HTTP-kutsuja ja tallennetaan HTTP-kutsun palauttamat arvot JavaScript-luokan muuttujiin.

### 3.3 Seurantaviikko 3

Maanantai 21.3.2022

Tänään katson kuntoon perjantaina tekemäni koodikatselmoinnin kommentit läpi ja teen mahdollisia korjauksia. Lisäksi mahdollisesti aloitan liitteiden ja viestin lähetys logiikkaa uusi viesti -sivulla.

Tämä päivä meni tuon koodikatselmoinnin läpikäymisessä. Keskustelin lisäksi toisen kehittäjän kanssa millä tavalla estetään tietyille sivuille navigoiminen riippuen tunnistautumismenetelmästä. Yksimielisesti todettiin, että olisi fiksua tehdä yhteinen toteutus tätä varten luomalla ”route guard” -luokka, joka toteuttaa Angularin tarjoaman rajapinnan canActivate. *Route guard* tai suomeksi reitinvartija nimellä kutsutaan luokkia, jotka toteuttavat jonkin Angularin tarjoamista sivun suojaamiseen luoduista rajapinnoista. Niiden avulla voi varmistaa sivulle navigoidessa, että annetut ehdot täyttyvät (Google, no date b; Chenkie, 2017).

Tiistai 22.3.2022

Tänään aion toteuttaa vielä viimeiset pari korjausehdotusta koodikatselmoinnista ja aloittaa eilen mainitun reitinvartijan. Jos siinä ei kestä kauaa aloitan myös uusi viesti -sivun muutokset liittyen liitekomponenttiin ja datan lähetykseen.

Sain koodikatselmoinnin ehdotukset tehtyä ja koodin yhdistettyä kehityshaaraan. Ehdin myös aloittaa reitinvartijan toteuttamista, mutta se jäi vielä kesken.

Keskiviikko 23.3.2022

Tänään teen reitinvartijan loppuun ja sen jälkeen aloitan uusi viesti -sivun tiedon lähetystä ja liitekomponentin päivitystä uuteen. Tänään on lisäksi designtiimin viikkopalaveri.

Viikkopalaverissa keskusteliin yhteisistä asettelua helpottavista CSS-luokista, jotka tulevat käyttöliittymäkirjaston viimeisimmässä versiossa mukana. Sain reitinvartijan tehtyä onnistuneesti. Se hakee sisään kirjautuneen käyttäjän tiedot, tarkistaa tunnistautumistavan. Jos se ei vastaa käytetyn komponentin reitin tiedoissa olevaa odotettua tunnistautumistapaa (esimerkiksi tupastunnistautuminen), niin reitinvartija ohjaa verkkopalvelun etusivulle. Toteutin reitinvartijan vielä niin, että sille voi myös määritellä reitin tiedoissa reitin, jolle se navigoi käyttäjän etusivun sijasta. Pyrin tekemään reitinvartijasta mahdollisimman yleiskäyttöisen. Tämä toteutuu käytännössä niin, että reitinvartijaa käyttävä sovellus voi itse määritellä mitä tunnistautumistapaa reitinvartija odottaa. Lisäksi käyttävä

sovellus voi itse määrittää reitin, minne reitinvartija ohjaa oikeuksien puuttuessa. Reitinvartijan lisäksi ehdin aloittamaan uusi viesti -sivun refaktorointia.

Torstai 24.3.2022

Tänään edistän eilen kesken jäänyttä uusi viesti -sivun refaktorointia. Lisäksi tänään on avoimet asiat niminen palaveri. Siellä käydään läpi lisää käyttäjätarinoita ja annetaan niille tarinapisteitä. Tarinapisteet ovat abstrakti arvio työn ajan tarpeesta (Radigan, no date). Ne eivät kuitenkaan ole verrattavissa mihinkään aikamääreeseen, kuten työpäiviin tai tunteihin.

Sain uusi viesti -sivun eilen valmiiksi ja laitettua sen koodikatselmointiin. Lisäksi sain reitinvartijan koodikatselmoinnista läpi ja yhdistettyä koodimuutokset kehityshaaraan.

Perjantai 25.3.2022

Tänään hoidin uusi viesti -sivun koodikatselmoinnissa tulleet huomiot kuntoon. Lisäksi ehdin aloittaa hieman sähköisen vanhuuseläkehakemuksen virheiden selvittelyä.

Viikkoanalyysi

Viime viikon viikkoanalyysissä kirjoitin RxJS:n switchMap-operaattorista ja tällä viikolla ajattelin kirjoittaa withLatestFrom nimisestä operaattorista. Se on suhteellisen yksinkertainen ja nimikin kertoo melko selkeästi mitä se tekee. Sille voi antaa attribuuteiksi yhden tai useampia tarkkailtavia. Se palauttaa niiden viimeisimmät arvot yhdessä lähdetarkkailtavan arvon kanssa aina, kun lähdetarkkailtava palauttaa arvon (RxJS Team, no date). Se mahdollisti kirjoittamani koodin yksinkertaistamisen tavalla mitä en osannut ajatella mahdolliseksi. Siltä toki monesti tuntuu, kun ei vielä osaa esimerkiksi jotain uutta teknologiaa läpikotaisin.

Oli siis tilanne, jossa minun täytyi hakea lista mahdollisia viestiketjun aiheita, joista käyttäjä voi valita. Tästä valinnasta riippuen, käyttäjälle tulee valittavaksi liitettävien liitteiden tyypeiksi vain tietyt valinnat, jotka liittyvät valittuun aiheeseen. Teknisemmällä tasolla ajateltuna minulla oli lista viestiketjun aiheita ja olio, joka sisälsi kutakin aihetta vastaavan listan liitteiden tyyppejä. Nuo tietorakenteet olivat kuitenkin tarkkailtavien sisällä. Alla pseudokoodia havainnollistamaan tietorakenteet.

```
conversationTopics = [aihe1, aihe2, aihe3, ...];

attachmentTypesPerTopic = {
   ”aihe1”: [tyyppi1, tyyppi2, ...],
   ”aihe2”: [...],
   ”aihe3”: [...]
};
```

Alun perin olin tallentanut tarkkailtavaan HTTP-kutsun palauttaman listan viestiketjun aiheita ja erilliseen tarkkailtavaan olion, joka sisälsi aiheita vastaavat listat liitteiden tyyppejä. Lisäksi käytin viestiketjun valintaan pudotusvalikkokäyttöliittymänkomponentin tarjoamaa tarkkailtavaa, joka palauttaa aina käyttäjän valitseman arvon (Google, no date d). Käytännössä sain kaiken tarvittavan tiedon pudotusvalikon tarkkailtavalta ja liitetyypit aiheittain sisältävän tarkkailtavan avulla. WithLatestFrom-operaattori mahdollisti sen, että pystyin tekemään kaiken tarvitsemani logiikan yhdellä tarkkailtavalla. Lisäksi käytin paria JavaScript listoistakin tuttua funktiota, map ja filter. Seuraavaksi abstraktoitu versio alkuperäisestä tilanteesta pseudokoodilla. Koodissa dollarimerkki kuvastaa muuttujaa, joka on tyypiltään tarkkailtava.

```
attachmentTypesPerTopic = getAttachmentTypesPerTopic()
.subscribe((types) => types);

this.form.get(‘category’) // Kategoriavalinnan käyttöliittymäkomponentti
        .subscribe((selectedCategory) => {
                this.selectedTopicCategories =
                attachmentTypesPerTopic[selectedCategory];
});
```

Seuraavaksi katsotaan, miten voisin tehdä tämän saman reaktiivisemmin.

```
attachmentTypesPerTopic$ = getAttachmentTypesPerTopic();

this.selectedTypes$ = this.form.get(‘category’).valueChanges.pipe(
        withLatestFrom(attachmentTypesPerTopic$),
        filter(notNilFilter),// Suodatetaan nolla-arvot varmuuden vuoksi
        map(([selectedCategory, typesPerTopic]) => {
                return Object.keys(typesPerTopic)
                .filter((key) => key === selectedCategory)
                .map((key) => typesPerTopic[key])
})
);
```

En päässyt eroon vain sisäisestä subscribe-funktion kutsusta, vaan pystyin jättämään lopullisen tiedon liitteiden tyypeistä tarkkailtavan sisään. Sitten Angularin async-putki hoitaa subscribe-funktion kutsut sivupohjatiedostossa. Näin tehdessä ei tarvitse huolehtia esimerkiksi mahdollisista muistivuodoista. Tässä tapauksessa vaara olisikin itseasiassa todellinen. Tämä johtuu siitä, että Angularin AbstractControl-luokan valueChanges-attribuutin tarkkailtava on ääretön tarkkailtava (Google, no date a; Sean G. Wright, 2016; Taylor, 2019). Sillä ei siis ole loppupistettä toisin kuin HTTP-kutsujen palauttamilla tarkkailtavilla. Tätä viikkoanalyysia kirjoittaessa huomasin, miten nopeaa olen esimerkiksi RxJS:n ymmärtämisessä kehittynyt. Huomasin kirjoittaessa myös vielä edelleen parempia tapoja toteuttaa samoja toiminnallisuuksia, jotka otin tämän viikkoanalyysin pseudokoodissa käyttöön ja joita voin taas seuraavalla viikolla hyödyntää töissä.

### 3.4 Seurantaviikko 4

Maanantai 4.4.2022

Tänään palasin sairaslomalta ja tarkoituksen alkaa ratkoa kesken jäänyttä sähköisen vanhuuseläkehakemuksen virhelistaa.

Päivittäispalaverissa sovittiin, että eläkehakemuksien virheiden selvittely siirretään eräälle toiselle kehittäjälle. Minä otin siis tänään työn alle viestiketjujen aiheiden haun palvelimelta, uuden viestiketjun luontia varten. Sain luotua testipalvelimelle uuden osoitteen, josta hakea viestiketjun aiheet. Lisäksi tein Angular-sovelluksen viesti- ja asiakirjapalvelun palveluluokkaan uuden metodin, jolla kutsutaan rajapinnan uutta osoitetta. Tänään oli lisäksi demotilaisuus, jossa esiteltiin viimeisimmän sprintin tuotoksia.

Tiistai 5.4.2022

Tänään tavoitteena edistää viestiketjun aiheiden valintaa uuden rajapinnan kautta.

Sain toteutettua lähetä viesti -sivulle logiikan, joka käyttää uutta lähdettä viestin aiheiden hakemiseen. Samalla uudistin hieman vanhaa viestin aiheisiin liittyvää koodia, pitämällä arvoja tarkkailtavien sisällä niin kauan, että niitä käytetään sivupohjatiedostossa, missä vaiheessa pystyn käyttämään Angularin async-putkea tarkkailtavan hallintaan.

Keskiviikko 6.4.2022

Tavoitteena on saattaa loppuun viestin aiheiden rajapintahaun muutostyö. Käytännössä täytyy viedä haettu tieto vielä sivupohjatiedostolle, jotta se päätyy käyttöliittymälle ja toteuttaa sama logiikka lähetä liite -sivulle.

Sain tehtyä viestin aiheet kokonaisuuden loppuun. Lisäksi tutustuin eläkehakemusten etusivuun liittyviin töihin, joihin jatkan seuraavina päivinä.

Torstai 7.4.2022

Tänään tarkastelen mahdollisuuksia ajaa kaikkia projektin Angular sovelluksen sisältämiä mikrosovelluksia yhtäaikaisesti, jotta voin testata eläkehakemusten etusivulla olevien linkkien toimivuutta käytännössä.

Tämä päivä meni tutkiessa eläkehakemusten etusivulle tehtäviä töitä, joissa oli joitain epäselvyyksiä sekä kaikkien mikrosovellusten rinnakaisajoa kokeillessa. Sovittiin huomiselle ylimääräinen palaveri, jossa käydään eläkehakemusten etusivua vielä tarkemmin läpi.

Perjantai 8.4.2022

Tänään eläkehakemusten etusivuun liittyvä ylimääräinen palaveri. Sitten alan työstämään eläkehakemusten etusivuun liittyviä toiminnallisuuksia.

Eläkehakemusten etusivuun liittyvät asiat selkeytyivät ja sain siihen liittyviä töitä aluilleen. Vaikuttaa siltä, että käyttöliittymäkirjastosta ei löydy täysin samanlaista komponenttia, mikä sivulle on suunniteltu. Se täytyy luultavasti toteuttaa manuaalisesti, mutta kyseessä onneksi yksinkertainen infolaatikkokomponentti, joten erityisen aikaa vievää se ei ole.

Viikkoanalyysi

Tällä viikolla tuntui, että työt lähtivät hieman hitaasti liikkeelle sairasloman jälkeen. Suhteellisen yksinkertaistenkin ohjelmointiin liittyvien työtehtävien valmiiksi saattamisessa oli hitautta. Ratkoin tätä ongelmaa pilkkomalla työtehtäviä niin pieniksi paloiksi kuin oli mahdollista ja pyysin perjantaille läpikäyntipalaverin uudesta työkokonaisuudesta. Ylimääräisten palavereiden sopiminen on tiimissämme yleinen käytäntö, jos ilmenee sellaisia ongelmia, joihin olisi hyvä saada useampien tiimiläisten mielipide tai jos asia koskee useita tiimiläisiä. Pääasiassa tällaiset asiat kuitenkin pyritään hoitamaan asiakasprojektissa käytettävien ketterien menetelmien mukaisissa seremonioissa.

Asiakasprojektissa on käytössä Scrum, jota on mukautettu asiakasyrityksen tarpeisiin. Scrum on viitekehys, joka tarjoaa ihmisille, tiimeille ja yrityksille kevyen tavan toteuttaa projekteja (Schwaber and Sutherland, 2020). Työskentelen henkilöasiakkaan palveluiden tiimissä, jossa on viisi kehittäjää, tuoteomistaja, Scrum Master ja projektijohtaja. Olen itse yksi kehittäjistä. Toteutamme sprintin kehitysjonolta tehtäviä. Tuoteomistaja vastaa itse tuotteesta ja näkemyksestä. Tiimimme Scrum Master tukee tiimiä Scrumin periaatteiden noudattamisessa ja johtaa seremonioita. Käsitykseni

mukaan projektijohtaja pitää omalta osaltaan huolen siitä, että projekti etenee aikataulussa. Projektijohtaja myös toimii rajapintana liiketoiminnan määritelmien ja kehittäjien välillä. Projektijohtaja ei ole osa Scrumin mukaista tiimirakennetta (Schwaber and Sutherland, 2020).

Virallisen Scrum-oppaan mukaan Scrum-tiimiin kuuluvat kehittäjät, tuoteomistaja sekä Scrum Master (Schwaber and Sutherland, 2020). Kehittäjien tehtävänä on toteuttaa ominaisuuksia tietyn ajanjakson sisällä, suunnitella kyseisen ajanjakson sisältö sekä huomioida tuloksen laatu ja tarvittaessa mukauttaa suunnitelmia. Meidän tiimissämme toteutettavat tehtävät valitsee tuoteomistaja ja nämä tehtävät sitten jaetaan kehittäjille. Käytännössä siis meidän tiimissämme kehitystyön jaksojen suunnittelutyö ei kokonaisuudessaan ole kehittäjien vastuulla, joka on Scrum-oppaan vastaista.

Tuoteomistaja on vastuussa tiimin tuottamasta arvosta (Schwaber and Sutherland, 2020). Tuoteomistajan tehtävänä on valmistella ja järjestellä kehitysjonoa sekä viestiä sen sisällöstä ja pitää se läpinäkyvänä. Myös tavoitteen viestiminen ja laatiminen on tuoteomistajan vastuulla. Meidän tiimissämme tuoteomistajan rooli on jäänyt itselleni hieman epäselväksi. En esimerkiksi ole varma siitä, hoitaako kehitysjonosta viestimistä ja kehitysjonon valmistelua tuoteomistaja. Tämä ei Scrum-oppaan perusteella ole tarkoituksenmukaista.

Scrum Masterin työtehtävät ovat monipuoliset ja hän on yhteistyössä niin koko tiimin kuin tuoteomistajan ja organisaationkin kanssa (Schwaber and Sutherland, 2020). Scrum Master vastaa tiimin tehokkuudesta ja toimii esimerkillisenä johtajana. Scrum-tiimin osalta Scrum Masterilla on tärkeä rooli tiimin kehitystyön esteiden poistamisessa, itseohjautuvuuteen valmentamisessa ja käyttäjille arvokkaiden asioiden korostamisessa. Lisäksi Scrum Master huolehtii, että tehtävät tulevat tehtyä valmiin määritelmän mukaisesti. Scrum Master toimii myös tuoteomistajan tukena ja organisaation suhteen Scrum Master tukee organisaatiota Scrumin toteutukseen liittyen.

Meidän tiimissämme valmiin määritelmä sisältää muun muassa yksikkötestauksen sekä käyttöliittymän osalta päästä päähän -testauksen. Meidän projektissamme aikataulu on kuitenkin määritelty niin tiukaksi, että Scrum Masterin tehtäviinkin kuuluvan valmiin määritelmän noudattamisen seuraaminen ei ole ollut mahdollista liian tiukan aikataulun vuoksi. Tiimimme Scrum Master pitää erittäin hyvin yllä positiivista ja avointa ilmapiiriä, joka on Scrumin periaatteiden mukaan tärkeä osa Scrum Masterin työtä (Schwaber and Sutherland, 2020). Lisäksi tiimimme Scrum Master on aktiivisesti yhteydessä kehittäjiin selvittääkseen, onko kehitystyössä esteitä ja ehdottaa ratkaisuja kehitystyön sujuvoittamiseksi.

Scrum-tiimin tulee kattaa kaikki osaaminen, joka tarvitaan arvon tuottamiseen (Schwaber and Sutherland, 2020). Tiimin tulee myös olla itseohjautuva ja kooltaan riittävän pieni, jotta kommunikaatio

sujuu hyvin. Scrum-tiimi on vastuussa kaikesta tuotteeseen liittyvästä, kuten ylläpidosta, tuotannosta, laadun varmistamisesta ja kehittämisestä. Tiimissämme vastataan pääasiassa kehittämisestä ja muut Scrum-tiimin vastuualueet jäävät vajaiksi. Tiimissä laadun varmistuksen osalta esimerkiksi käyttöliittymän testaus ei ole kokonaisvaltaista ja testaus tapahtuu käsittääkseni suurilta osin tiimin ulkopuolella, joka on Scrumin periaatteiden vastaista.

Scrum-tiimiin kuuluvat olennaisena osana tiimin jäsenten lisäksi Scrumin tapahtumat. Sprintti on yksi Scrumin tapahtumista, joka on pituudeltaan vakio ajanjakso, jonka aikana edistetään kehitysjonon tehtäviä (Schwaber and Sutherland, 2020). Sprintti sisältää itsessään myös Scrum tapahtumia. Sprintin sisältämiä tapahtumia ovat sprintin suunnittelu, päivittäispalaverit, sprintin katselmointi ja sprintin retrospektiivi. Meidän tiimissämme sprintin tapahtumia kutsutaan seremonioiksi.

Sprintin alussa pidetään sprintin suunnittelu, johon osallistuu koko tiimi (Schwaber and Sutherland, 2020). Sprintin suunnittelussa määritellään sprintin tavoite, valitaan tehtävät, jotka tehdään sprintin aikana sekä suunnitellaan näiden tehtävien toteutus. Kuukauden pituisen sprintin suunnittelupalaverin maksimipituus on kahdeksan tuntia. Tiimissämme sprintin pituus on kolme viikkoa ja kalenterivaraus sprintin suunnittelulle on kaksi ja puoli tuntia. Tämä on siis vielä Scrumin virallisen oppaan mukainen pituus. Meidän tiimissämme kuitenkin suunnitteluun aikataulutettu kaksi ja puolituntia ei useinkaan riitä. Tiimissämme on käytössä lisäksi avoimet asiat niminen palaveri. Se pidetään periaatteessa vain tarvittaessa, mutta lähes aina siellä joudutaan jatkamaan sprintin suunnittelua keskellä sprinttiä. Se ei ole Scrumin periaatteiden mukaista.

Päivittäispalavereissa tarkastellaan sprintin etenemistä (Schwaber and Sutherland, 2020). Näiden palavereiden yhteydessä on myös mahdollista muuttaa kehitysjonoa. Päivittäispalaveri saa kestää maksimissaan 15 minuuttia ja niiden tarkoituksena on tuoda esiin kehitystyön mahdolliset esteet ja parantaa vuorovaikutusta. Meidän tiimissämme päivittäispalaveri on aikataulutettu kestämään 15 minuuttia, mutta lähes aina palaverin todellinen pituus on lähempänä 30 minuuttia tai jopa 45 minuuttia. Oman näkemykseni mukaan palavereiden venyminen johtuu yleensä kehitysjonon tehtävien määrittelyn ja suunnittelun puutteellisuudesta.

Sprintin katselmoinnissa katsotaan, mitä sprintin aikana on toteutettu (Schwaber and Sutherland, 2020). Toteutunut työ esitellään myös olennaisille sidosryhmille. Meidän tiimissämme katselmointi on tiimin sisäinen palaveri, jossa käydään läpi, mitä sprintin aikana on saatu aikaiseksi. Lisäksi asiakasyrityksessä järjestetään sprintin alussa edellistä sprinttiä koskeva demotilaisuus, jossa useat asiakasyrityksen kehitystiimit esittelevät sprintin aikaansaannokset liiketoiminnalle. Scrum-oppaan mukainen katselmointi on siis käytännössä meillä kahdessa osassa.

Sprintin retrospektiivissä tarkastellaan kulunutta sprinttiä ja tarkoituksena on parantaa tiimin työtapoja (Schwaber and Sutherland, 2020). Meidän tiimissämme retrospektiivi järjestetään joka toisella sprintillä. Käytännössä joka toisella sprintillä tiimin työskentely jää siis reflektoimatta. Sprintin retrospektiivin tarkoituksena on myös tehdä tarvittaessa muutoksia työskentelyyn tiimin havaintojen pohjalta. Käytännössä projektin aikataulu estää toteuttamasta parannuksia havaittuihin ongelmiin tiimin työskentelyn tehostamiseksi.

Tiimissämme siis noudatetaan Scrum-viitekehystä joiltain osin, mutta siihen on tehty myös muutoksia. Scrumin tarkemmalla noudattamisella voisi olla vaikutusta työskentelyyn ja se voisi mahdollistaa tehokkaamman kehitystyön. Sitäkin isommaksi ongelmaksi koen, että liian tiukan aikataulun vuoksi tarvittavien parannusten tekemiseen ei ole aikaa, vaikka retrospektiiveissä tiimi käy tuottoisaa keskustelua havainnoista työskentelyn ongelmakohdista ja niiden ratkaisuista. Tiimissämme kuitenkin selkeästi noudatetaan ketterää kehitystapaa, joka on saanut paljon vaikutteita Scrumista.

Meidän Scrum-tiimimme on sopivan kokoinen, mutta asiakasorganisaatiossa on monta muutakin kehitystiimiä. Tällaisissa tilanteissa on havaittu tehokkaammaksi alkaa toteuttaa lisäksi niin sanottua Scrumien Scrumia (Mundra, Misra and Dhawale, 2013). Scrumien Scrum (Scrum of Scrums, SoS) tarkoittaa tiimiä, joka muodostuu useasta kehitystiimistä (Spanner, no date; Mundra, Misra and Dhawale, 2013). Sen on havaittu hyödyttävän erityisesti suurissa projekteissa, jossa Scrumien Scrum muodostetaan useasta samaa tuotetta kehittävästä tiimistä. Tällainen malli voisi oikein toteutettuna olla mahdollisesti hyödyksi myös asiakasorganisaatiossa.

Scrumia noudattavilla ohjelmistokehitystiimeillä on havaittu olevan korrelaatiota paremman tuottavuuden kanssa (Cardozo *et al.*, 2010). Lisäksi suurissa useista tiimeistä muodostuvissa projekteissa Scrumien Scrumin on havaittu olevan tehokkaampi, kuin pelkät yksittäiset itsenäiset Scrum tiimit (Mundra, Misra and Dhawale, 2013). Toisaalta monissa Scrumin tehokkuutta mittaavissa tutkimuksissa on hyväksytty mukaan Scrumia vapaammin soveltavia tiimejä sekä tiimejä, jotka yhdistelevät Scrumiin muita ketteriä menetelmiä (Cardozo *et al.*, 2010). Tämän takia ei voida vetää johtopäätöstä, että Scrumin mahdollisimman tarkka noudattaminen toteuttaisi aina parhaan lopputuloksen.

### 3.5 Seurantaviikko 5

Maanantai 11.4.2022

Viikonlopun aikana minulle oli tullut sähköpostiin ilmoituksia virheistä staging-ympäristössä. Staging ympäristö on palvelinympäristö, joka muistuttaa mahdollisimman paljon tuotantoympäristöä (Luukkainen and Ilves, 2021). Tänään tarkoituksen selvitellä niitä sekä tehdä eläkehakemusten etusivuun liittyviä tehtäviä alta pois.

Selvitin tänään yhden virheilmoituksen. Se aiheutui siitä, että staging-ympäristössä Angular-sovelluksen palvelinrajapinnalta vastaanottamasta JSON-objektista puuttui joitain Angular-sovelluksen odottamia kenttiä. Pidettiin lyhyt puhelu palvelinkehittäjän kanssa ja sovittiin ratkaisuksi, että hän lisää nämä kentät rajapintakutsun vastaukseen. Toinen vaihtoehto olisi ollut kirjoittaa näiden kenttien arvot suoraan käyttöliittymäkoodiin kiinni. Todettiin kuitenkin, että olisi nurinkurista, että osa tarvittavasta tiedosta ei tulisikaan palvelimelta vaan olisi niin sanotusti kova koodattu käyttöliittymäsovellukseen. Etenin myös eläkehakemusten etusivuun liittyvissä tehtävissä eteenpäin hyvää tahtia. Työviikko lähti tosi mukavasti liikkeelle.

Tiistai 12.4.2022

Tänään toivon saavani eläkehakemuksien etusivun tehtävät lähes valmiiksi. Lisäksi iltapäivällä on tiedossa asiakkaan saavutettavuuskoulutus liittyen saavutettavien verkkopalvelujen kehittämiseen.

Etenin jälleen hyvin eteenpäin eläkehakemuksien etusivun tehtävien parissa. Lisäksi asiakkaan saavutettavuuskoulutus osoittautui erittäin mielenkiintoiseksi ja tarkoituksena onkin vielä tänä vuonna jatkaa saavutettavuusasioissa kehittymistä käymällä työnantajan tarjoamia koulutuksia.

Keskiviikko 13.4.2022

Tavoitteena saada eläkehakemuksien etusivun tehtävät valmiiksi ja mahdollisesti aloittaa tutkimaan siihen liittyviä virheen selvitys tehtäviä. Eilisestä saavutettavuuskoulutuksesta sain myös ideoita joihinkin saavutettavuusparannuksiin eläkehakemuksien etusivulle, joita aion toteuttaa.

Sain eläkehakemuksien etusivun tehtävät valmiiksi, virheiden selvittelyt tosin jäävät vielä huomiselle, kuten vähän oletinkin. Saavutettavuusparannukset ja muutostyöt eläkehakemusten etusivulla on nyt valmiita koodikatselmointiin.

Torstai 14.4.2022

Tänään toivon saavani eläkehakemukset etusivun muutokset koodikatselmoinnista läpi ja asennukseen staging-ympäristöön. Sitten selvittelen samalla sivulla nostettuja ilmoituksia virheistä. Jos ne ovat nopeasti alta pois saatan hypätä tänään vielä takaisin viesti- ja asiakirjapalvelun pariin.

Eläkehakemuksien etusivun koodikatselmoinnissa ei noussut esiin mitään ongelmia, joten sain muutokset vietyä staging-ympäristöön. Asennuksen valmistuttua huomasin tosin, että sivu ei toiminut asennuksen jälkeen staging-ympäristössä enää ollenkaan. Tänään pidettiin lisäksi pieni palaveri projektipäällikön kanssa viesti- ja asiakirjapalveluun liittyvistä tehtäväjonossa seuraavaksi olevista tehtävistä. Aloitin myös viesti- ja asiakirjapalvelun etusivuun liittyvää tehtävää. Siinä täytyy muuttaa sivun ingressiteksti muuttumaan käyttäjän tunnistautumistavan ja käyttäjäroolin mukaan.

Viikkoanalyysi

Viikon mielenkiintoisin työtehtävä liittyi varmastikin tiistain saavutettavuuskoulutukseen ja sen jälkeisiin ideoihin saavutettavuusparannuksista silloin työn alla olleelle eläkehakemusten etusivulle. Olenkin asettanut saavutettavien verkkopalveluiden kehityksen jo entuudestaan tämän vuoden oppimistavoitteisiini, joita asetetaan Tietoevryllä vuosittain osana MyGrowth-prosessia (Tietoevry, 2022b, 2022a). MyGrowth-prosessissa asetetaan vuosittain henkilökohtaisia ja ammatillisia kehittymis- ja oppimistavoitteita, joiden toteuttamista Tietoevry tukee eri tavoin.

WCAG (Web Content Accessibility Guidelines) tarjoaa standardin verkkosisältöjen saavutettavuudelle (W3C, no date e). WCAG selittää, miten verkkosisällön saa saavutettavaksi toimintarajoitteisille ja niille, joilla esimerkiksi on jokin vamma. Saavutettavuus tarkoittaa siis sitä, että verkkopalvelut ja -sisällöt pyritään toteuttamaan niin, että erilaisten ihmisten on helppoa käyttää palveluita (Aluehallintovirasto, no date e). Yhdenvertaisuus lisääntyy saavutettavuuden myötä. WCAG 2.1 standardi eli tällä hetkellä viimeisin WCAG-standardi määrittelee 13 ylätason ohjetta saavutettavuuden toteuttamiseen (Aluehallintovirasto, no date d; W3C, no date e). Näiden 13 ylätason ohjeen alla on 78 kriteeriä, jotka on jaettu yhden, kahden ja kolmen A:n vaatimustasoon. Kolmen A:n vaatimustaso on näistä vaativin, mutta sekään ei takaa täydellistä saavutettavuutta. WCAG-kriteerien ajatuksena onkin vain luoda digitaalisten palveluiden saavutettavuuden minimitaso, joka mahdollistaisi mahdollisimman monille verkkopalvelujen käyttämisen.

Esimerkkejä WCAG:n vaatimuksista on videoiden tekstitys, joka on vaatimus A-tason täyttämiseksi (Aluehallintovirasto, no date d). AA-tason vaatimus taas on esimerkiksi ääniselitteen tarjoaminen videosisällölle. Tiukimman, eli AAA-tason vaatimus on sisällön löytyminen myös esimerkiksi viittomakielisinä videoina. Myös tekstin ymmärrettävyyden parantaminen on AAA-tason kriteeri.

Digitaalisten palvelujen saavutettavuus on erittäin tärkeää monille rajoitteisille ihmisille ja se on hyödyllistä myös ei-rajoitteisille. Arvioiden mukaan suomalaisista yli miljoona eli noin viidesosa tarvitsee saavutettavia palveluita (Aluehallintovirasto, no date b). Aluehallintoviraston mukaan on paljon ryhmiä, joita ei huomioida riittävästi tai ollenkaan digitaalisia palveluita suunniteltaessa. Tällaisia ihmisryhmiä ovat esimerkiksi henkilöt, joilla on heikentynyt näkö tai näkövamma, erilaiset kognitiiviset rajoitteet tai esimerkiksi heikko suomen kielen taito. Lisäksi kaikkiin ihmisiin vaikuttavat tilapäisesti mahdollisia ongelmia aiheuttavat tilat ja ilmiöt, kuten esimerkiksi sivun luettavuutta ruudulla heikentävä kirkas auringonvalo. Saavutettavuus ei siis ole pelkästään teknistä toteutusta, jolla mahdollistetaan esimerkiksi verkkopalvelun yhteensopivuus ruudunlukijan kanssa. Se sisältää myös normaaleja hyvien käyttöliittymä- ja käyttökokemuskäytänteiden omaksumista ja käytäntöön panemista.

Digipalvelulaki antaa vaatimuksia saavutettavuudesta julkiselle sektorille ja osalle yksityisen ja kolmannen sektorin organisaatioista (Aluehallintovirasto, no date c, no date a). Myös eläkevakuutusyhtiöt ovat Suomessa organisaatioita, joita digipalvelulaki ja sen saavutettavuusvaatimukset koskevat. Näin ollen myös asiakasprojektissani saavutettavuusasiat tulee siis ottaa huomioon. Digipalvelulaki velvoittaa noudattamaan A- ja AA-tason kriteereitä.

Omassa työssäni saavutettavuus ja digipalvelulain vaatimukset ovat tulleet vahvemmin esiin tiimimme tekemisessä vasta nyt projektin loppumetreillä. Saavutettavuuden perusasiat ovat kuitenkin lähtökohtaisesti kunnossa, koska projektissa on käytössä Googlen ylläpitämän Angular Material -käyttöliittymäkirjaston johdannainen (Google, no date l). Saavutettavuuden perusasiat ovat siis jo lähtötilanteessa kunnossa, koska Google on Angular Material -käyttöliittymäkirjastossa ottanut jokaisessa komponentissa saavutettavuuden huomioon ja jokaisen komponentin dokumentaatiossa onkin oma osuus saavutettavuudelle. Angular Material -käyttöliittymäkirjaston komponentteja on siis muokattu asiakkaan tarpeisiin esimerkiksi asiakkaan brändin mukaiseksi.

Vielä kuitenkin jää joitain tapauksia mitä valmiiksi saavutettavilla komponenteilla ei voi ottaa huomioon. Esimerkiksi tällä viikolla eläkehakemuksien etusivua kehittäessä oli tilanne, missä eri eläkehakemusten yläotsikoiden ja ingressitekstin alla oli nappula, jonka tekstinä on ”siirry hakemukselle”. Ruudunlukijan käyttäjälle tilanne voi olla hämmentävä, koska ruudunlukija lukee monen eri nappulan kohdalla vain ”siirry hakemukselle” ilman mitään lisätietoja. Tässä tapauksessa päädyin parantamaan saavutettavuutta vielä manuaalisesti. Käytännössä ohjelmoin nappulalle ruudunlukijalle näkyvän selitteen, joka saa ruudunlukijan lukemaan nappulan kohdalla ensin nappulan sisältämän tekstin eli ”siirry hakemukselle” ja sitten yläotsikon tekstin, jonka alla nappula sijaitsi. Kokonaisuudessaan ruudunlukija saattaisi siis lukea parannuksen jälkeen esimerkiksi ”siirry hakemukselle vanhuuseläkehakemus”.

Saavutettavuuteen liittyy monia haasteita (Abuaddous, Jali and Basir, 2016). Haasteet voivat liittyä standardeihin ja ohjeisiin, verkkosisällön suunnitteluun ja kehittämiseen sekä saavutettavuuden arviointiin. Erilaisia ohjeita ja sääntelyä on paljon, ja ne poikkeavat esimerkiksi maiden välillä. Myös saman maan sisällä eri sektoreita ja aloja voi koskea erilaiset määräykset. Haasteena voidaan myös nähdä, että saavutettavuusstandardien toteutus vaatii tietyntasoisen ymmärryksen teknologisen osaamisen lisäksi itse saavutettavuudesta. Lisäksi saavutettavuusstandardien seuraaminen saattaa olla tehotonta yrityksen kannalta, sillä jopa isoissa yrityksissä resursseista esimerkiksi aika, raha ja henkilöstö voivat muodostua esteeksi sille, että saavutettavuuteen panostettaisiin riittävästi.

Verkkosisällön suunnittelun ja kehittämisen aikana voi esiintyä useita haasteita (Abuaddous, Jali and Basir, 2016). Ohjelmistokehittäjien keskuudessa ei välttämättä ole tarpeeksi tietotaitoa saavu-

tettavuuteen liittyen. Myös motivaation puute saavutettavuuteen liittyen voi olla haaste, elleivät kehittäjät ole saavutettavuudesta riittävän tietoisia. Saavutettavuutta ei lisäksi usein sisällytetä koulutukseen (Abuaddous, Jali and Basir, 2016). Kehittäjillä ei tällöin ole tarvittavaa osaamista saavutettavuudesta, mikä voi johtaa saavutettavuuden laiminlyöntiin. Jos saavutettavuusasiat tuotaisiin kehittäjien tietoisuuteen jo koulutuksen alkuvaiheessa, se voisi positiivisesti vaikuttaa kehittäjien motivaatioon toteuttaa saavutettavia verkkopalveluita.

Myös saavutettavuuden arviointiin ja testaamiseen liittyy haasteita (Abuaddous, Jali and Basir, 2016). Tähän on kehitetty useita menetelmiä ja niitä on tutkittu laajasti. Eri menetelmien välillä eroja ovat esimerkiksi käyttötarkoitus sekä vaadittavat resurssit. Saavutettavuuden arviointia voi tehdä hyödyntäen automaatiota tai käyttäjälähtöistä testausta. Automaattisia työkaluja käyttäessä aikaa voi kulua huomattavasti sopivan työkalun etsimiseen ja valintaan ja usein lisäksi tarvitaan myös manuaalista tarkastelua. Näiden tuloksien analysoiminen vaatii lisäksi kokemusta, jotta tulokset ovat laadukkaita. Työkalusta riippuen tulokset voivat vaihdella sen mukaan, miten automaattista arviointia tekevä työkalu tulkitsee saavutettavuusstandardeja. Käyttäjälähtöisessä testauksessa haasteena on muun muassa se, että se vaatii paljon aikaa ja tavallisella ohjelmistokehittäjällä ei välttämättä ole tarvittavaa osaamista tai mahdollisuuksia toteuttaa tällaista testausta. Suurilla joukoilla testaaminen on myös kallista.

Näitä haasteita voi ratkoa esimerkiksi asettamalla tiimiin saavutettavuusasiantuntija (Abuaddous, Jali and Basir, 2016). Lisäksi haasteisiin voi puuttua vaikuttamalla kehitystiimin ajattelutapaan muun muassa painottamalla saavutettavuuden merkitystä jo ohjelmistokehityksen koulutuksen aikana. Yksinkertaiset ja helposti lähestyttävät ohjeistukset saavutettavuuden toteuttamiseen voivat kannustaa ohjelmistokehittäjiä panostamaan enemmän saavutettavuuteen.

Omalla kohdallani saavutettavuuden ajatteleminen osana verkkopalvelujen kehittämistä on tullut vasta aivan opintojen loppusuoralla ja työuran alussa. Opinnoissa saavutettavuudesta ei juurikaan ollut puhetta. Mahdollinen syy voi olla siinä, että saavutettavuus on noussut erityisen tärkeään asemaan vasta vuonna 2019 ja asiaan on herätty vasta sen jälkeen. Silloin uusi digipalvelulaki alkoi vaatia julkisia palveluita toteuttamaan vähintään WCAG A- ja AA-tason kriteerit (Valtiovarainministeriö, 2019).

Nykyään koen saavutettavuuden erittäin mielenkiintoiseksi kehittämisen osa-alueeksi, ja tavoitteena on kouluttautua sillä osa-alueella lisää. Toistaiseksi saavutettavan verkkopalvelun kehittäminen on kuitenkin tuntunut suhteellisen haastavalta, sillä valmiita taustatietoja ei juurikaan ole ollut. Työpaikalla yleinen mielipide onkin, että huonosti toteutettu saavutettavuus on isompi paha kuin ei saavutettavuutta. Tämä huomio tuotiin esille myös tämän viikkoisessa saavutettavuuskoulutuksessa.

Asiakasprojektissa tunnistan, että liian tiukka aikataulu yhdistettynä liian pieniin resursseihin, hankaloittaa huomattavasti, ellei täysin estä saavutettavuuden huomioimista kehitystyössä. Osalla projektin kehittäjistä ei välttämättä myöskään ole tietoa saavutettavuudesta tai se on hyvin vähäistä. Koko projektin ja kehitystiimin näkökulmasta tämän viikkoinen saavutettavuuskoulutus varmasti omalta osaltaan toi saavutettavuuden enemmän esille tärkeänä kehitystyön osa-alueena. Se myös varmasti opetti jonkin verran projektin kehittäjille uusia teknisiä taitoja saavutettavuuden toteuttamiseen. Käytännössä kuitenkin valmista saavutettavuusosaajaa ei synny yhden koulutuksen seurauksena ja siihen pääseminen vaatisi todennäköisesti konsulteista muodostuvassa kehitystiimissä kehittäjiltä oma-aloitteisuutta.

### 3.6 Seurantaviikko 6

Tiistai 19.4.2022

Tänään on eläkevakuutusyhtiön verkkopalveluita kehittävien tiimien yhteinen viikoittaispalaveri. Kehitystyön osalta tavoitteena tehdä viesti- ja asiakirjapalvelun etusivulle tunnistautumistavasta riippuvainen ingressiteksti. Jos se tapahtuu helposti, alan toteuttamaan myös eläkevakuutusyhtiön henkilökunnan ominaisuuksia viesti- ja asiakirjapalveluun.

Sain tehtyä muuttuvan ingressitekstin valmiiksi ja luotua sille koodikatselmoinnin, joka on vielä katselmoimatta. Otin myös työn alle ominaisuuden, joka luo henkilökunnan roolissa palvelua käyttävälle viestiin automaattisesti allekirjoituksen käyttäjän etu- ja sukunimellä. Lähdin aluksi selvittelemään, miten pystyn sisällyttämään tekstikenttään tekstiä heti sivun latautuessa. Se ei ollut itselleni entuudestaan tuttua. Kävi ilmi, että viestin tekstikenttään sidotun Angularin FormControl-olion alkuarvo kirjoittaa yli tekstikenttään pelkällä HTML:llä kirjoitetun tekstin (Google, no date d). Allekirjoitus teksti täytyy siis asettaa tekstikentän arvoksi ohjelmallisesti komponentin TypeScript koodin puolella. Sen toteuttaminen jäi vielä alkutekijöihin.

Keskiviikko 20.4.2022

Tänään minulta jää päivittäispalaveri välistä. Jatkan eilistä allekirjoitusominaisuutta ja toivon saavani edistettyä myös koodikatselmoinnissa olevaa ehdollisesti muuttuvan ingressitekstin ominaisuutta.

Sain koodikatselmointiin huomioita ja keskustelin toisen frontend-kehittäjän kanssa ja ilmeni, että yksi kolmesta ehdosta ingressitekstissä oli hieman väärin. Ingressitekstin pitää siis olla eri TUPAS järjestelmän kautta tunnistautuneille, käyttäjätunnuksella tunnistautuneille ja käyttäjälle, jolla on oikeudet sekä henkilö- että yrittäjäasiakkaan palveluihin. Viimeisimmän ehdon toteutumisen olin tar-

kistanut koodissa hieman väärin. Piti hieman selvitellä Slack-viestintäsovelluksen välityksellä, miten se kuuluisi toteuttaa. Siihen löytyi onneksi äkkiä vastaus ja sain toiminnallisuuden korjattua. Lisäksi sain allekirjoitusominaisuudesta yksinkertaisen version toteutettua. Se toimii muuten, mutta sitä täytyy vielä päivittää niin, että allekirjoituksen kieli vaihtuu palvelussa valitun kielen mukaan.

Viikkoanalyysi

Projektin kannalta saavutettava lopputulos on erittäin tärkeää. Tämä johtuu siitä, että asiakasta sitoo digipalvelulaki, mistä kirjoitin myös edellisessä viikkoanalyysissä. Saavutettavuus näkyy omassa työssä niin, että täytyy tietää miten saavutettavuutta voi parantaa kehittäjän näkökulmasta katsottuna. Tämän viikon viikkoanalyysissä paneudun siihen, miten saavutettavuutta testataan ja kehitetään.

Asiakasorganisaatiossa saavutettavuuden huomioiminen on alkujaan lähtenyt saavutettavuusauditoinnin tilaamisesta ulkoiselta palveluntarjoajalta. Saavutettavuusauditointi tilattiin, koska uusi digipalvelulaki oli tulossa voimaan. Palveluntarjoajan tekemä saavutettavuusauditointi paljasti asiakkaan verkkopalveluissa piilleet saavutettavuus ongelmat. Auditointi ja sen tuottamat havainnot olivat kuitenkin välttämättömiä. Ilman sitä asiakasorganisaatiossa ei olisi voitu tietää mitä ja miten paljon tehtävää WCAG:n A- ja AA-tason kriteerien saavuttaminen vaatisi. Saavutettavuusauditoinnit ovat myös jääneet osaksi asiakkaan verkkopalveluiden kehittämistä ja niitä tehdään säännöllisesti uudestaan. Näin on mahdollista seurata kehitystä ja tietää missä mennään. Se on myös tuonut ilmi sen, että mahdolliset saavutettavuusparannukset eivät lopu nopeasti kesken. Auditoinneissa löytyvät ongelmat vain muuttuvat pikkuhiljaa pienemmiksi ja pienemmiksi.

Tehokkainta on testata saavutettavuutta jo kehityksen aikaisissa vaiheissa (W3C, no date b). Silloin ilmenneet ongelmat ovat helpointa ja halvinta korjata. Tämän mahdollistamiseksi kehittäjän on tärkeää tietää tapoja, miten voi itse testata saavutettavuutta jo kehityksen aikana. Tätä työtä helpottamaan on olemassa paljon erilaisia työkaluja.

En ole itse tietoinen, että asiakasorganisaatiossa olisi varsinaisia virallisia dokumentoituja ohjeita saavutettavuuden testaamiseen, mutta kehitettävien palveluiden ja ominaisuuksien vaaditaan olevan saavutettavia. Testaamisen helpottamiseksi olen itse havainnut hyödylliseksi navigoida kehitettävää verkkosivua myös näppäimistöllä ja kiinnittää huomiota pääseekö kaikkiin nappeihin ja toimintoihin käsiksi pelkällä näppäimistöllä. Lisäksi on hyvä kiinnittää huomiota siihen, liikkuuko kohdistin näppäimistöä käyttäessä oikein eli yleensä normaalissa lukemisjärjestyksessä (W3C, no date a). Kohdistettu elementti pitäisi myös visuaalisesti erottua selkeästi, jotta on helppo nähdä

missä kohdistus liikkuu. Elementtien sisällä liikkuminen tulisi tapahtua nuolinäppäimistöllä ja valinnan tapahtua välilyönnillä tai rivinvaihdolla. Jo WCAG 2:n A-taso vaatii näppäimistönavigoinnin mahdollistamista (W3C, no date c).

Tiimeissä, joissa ei vielä tehdä ollenkaan saavutettavuustestausta, on helppoa ja edullista aloittaa se käyttämällä heikentynyttä näköä simuloivia silmälaseja ja automaattisia tarkistusohjelmia (Bai, Mork and Stray, 2017). Heikentyneen näön simulointi ja tarkistusohjelmat ovat kehittäjälle helppoja saavutettavuustestaamisen tapoja, joiden käyttämisessä on vain pieni kynnys. Heikentynyttä näköä simuloivat lasit auttavat kehittäjää kokemaan saavutettavuusongelmat omin silmin.

Automaattitestauksella voi löytää saavutettavuusongelmia helposti ja nopeasti. Automaattitestiohjelmistot tarkastelevat sovelluksen koodia ja etsii yleisiä teknisiä virheitä, jotka esimerkiksi rikkovat toiminnan ruudunlukijoiden kanssa. Automaattitestauksessa on myös se hyvä puoli, että jotkut automaattitestiohjelmistot voidaan integroida suoraan kehittäjän kehitysympäristöön, jolloin ongelmat ilmenevät reaaliajassa sovellusta luodessa. Näitä saavutettavuus testauksen menetelmiä voi hyödyntää jo kehityksen alusta alkaen. Esimerkki tällaisesta on HTML:n otsikko elementtien oikea oppinen hierarkia.

Myöhemmissä kehitysvaiheissa kannattaa ottaa saavutettavuustestauksen osaksi myös WCAG-kriteerien läpikäynti, sovelluksen käyttäminen ruudunlukijalla ja käyttäjäprofiilitestaus (Bai, Mork and Stray, 2017). Ne ovat menetelminä kalliimpia, mutta niillä löytää myös lisää saavutettavuusongelmia. WCAG-kriteerit testausmenetelmänä tarkoittaa käytännössä kriteerien läpikäymistä manuaalisesti ja tarkistamalla toteutuuko ne. Sitä voi tehdä esimerkiksi käyttäjätarinatasolla. WCAG-kriteerien heikkous testausmenetelmänä on, että monet kehittäjät pitävät niitä liian kuormittavina käyttää säännöllisesti osana saavutettavuustestausta.

Ruudunlukija on sovellus, joka muuttaa ruudulla olevaa tekstiä puheeksi käyttäjälle puhesyntetisaattorin avulla. Esimerkiksi näkörajoitteiset voivat käyttää ruudunlukijaa apuna tietokoneen, tabletin tai älypuhelimen käyttämiseksi. Ruudunlukija vaatii kehittäjältä jonkin aikaa opettelua ja totuttelua, mutta ruudunlukijalla voi havaita menetelmistä kaikkein eniten toiminnon suorittamisen estäviä ongelmia. Näin on varsinkin, jos ruudunlukijaa ei ole käytetty apuna sovelluksen kehityksen alusta lähtien.

Olen itse kokenut ruudunlukijan erityisen hyödylliseksi työkaluksi. Ruudunlukijalla verkkosivun käyttäminen avaa toimintarajoitteettomana kehittäjänä aivan uuden maailman verkkosivujen selaamiseen. Se soveltuu mielestäni kehityksen kaikkiin vaiheisiin, kunhan kehittäjä kerran oppii ruudunlukijan käytön riittävällä tasolla. Sen jälkeen sen käyttämiselle on suhteellisen matala kynnys.

Käyttäjäprofiilitestaus tarkoittaa sitä, että testaaja eläytyy määritellyn käyttäjäprofiilin mukaiseksi käyttäjäksi, jolla on jokin toimintarajoite (Bai, Mork and Stray, 2017). Tämä vaatii testaajalta tietämystä tarpeellisista toimintarajoitteista ja siitä millä kaikilla eri tavoilla ne voivat vaikuttaa käyttäjän toimintaan. Käyttäjäprofiilitestausta kannattaa käyttää harvemmin kuin edellä mainittuja testaustapoja. Se on kalliimpaa ja sen liika toistaminen saman henkilön toimesta voi aiheuttaa vinoumia testituloksiin. Jos tiimissä on kokemusta käyttäjäprofiilitestauksesta tai sen vaatimaan koulutukseen voi investoida, se on erittäin hyvä tapa testata saavutettavuutta.

Projektitasolla on hyvä valita aiemmista testaustavoista vähintään kaksi (Bai, Mork and Stray, 2017). Itse arvioisin, että jos kaksi aiempaa testaustapaa on projektitasolla minimitaso, niin kehittäjänä voi auttaa saavutettavuuden toteutumista erittäin paljon ottamalla käyttöön esimerkiksi ruudunlukijan ja kehitysympäristöön integroituvan automaattitestiohjelmiston osaksi omaa työskentelyä. Itsellä ruudunlukija on jo ollut satunnaisesti käytössä. Se on auttanut havaitsemaan puutteita saavutettavuudessa. Olen esimerkiksi havainnut tilanteen, jossa oli tarpeellista ohjelmoida ruudunlukijan lukemaan nappulan sisällön lisäksi otsikko, jonka alla se oli.

Suurin osa yhteensopivuudesta ruudunlukijoiden kanssa tulee siitä, että seuraa parhaita käytänteitä HTML-sivupohjatiedostoja luodessa. Puhutaan myös semanttisen HTML:n kirjoittamisesta (Wunder, no date). Se tarkoittaa, että käytetään mahdollisimman kuvaavia HTML-elementtejä. Semanttisia HTML-elementtejä ovat esimerkiksi <nav> ja <p>. Ei-semanttisia HTML-elementtejä ovat esimerkiksi  ja .

Joissain tapauksissa on tarve kuitenkin antaa ruudunlukijalle itse tieto komponentin toiminnasta. Näin täytyy tehdä erityisesti monimutkaisten ja interaktiivisten, eri tarpeisiin mukautettujen komponenttien tapauksissa. Voi tulla tarve käyttää paljon ei-semanttisia HTML-elementtejä. Näissä tapauksissa ruudunlukijaa voi auttaa ymmärtämään omaa, mukautettua komponenttia oikein käyttämällä WAI-ARIA nimistä standardia (Mozilla, no date d; W3C, no date d). Kehittäjälle se näyttäytyy mahdollisuutena lisätä HTML-elementeille attribuutteja, jotka kertovat ruudunlukijalle komponentin roolista, ominaisuuksista ja tilasta.

Tällaisissa komponenteissa on kuitenkin aina se heikkous, että käyttäjän ei voida olettaa tietävän miten mukautettu komponentti toimii (Mozilla, no date d). Esimerkiksi pudotusvalikot toimivat eri käyttöjärjestelmillä eri tavoilla eli eri käyttöjärjestelmien käyttäjät ovat tottuneet erilaisiin toteutuksiin. Jos kuitenkin päättää tehdä oman pudotusvalikon, niin täytyy valita jokin yksi tapa, miten pudotusvalikko toimii kaikille käyttäjille.

Saavutettavuustestaus on siis tärkeä ottaa osaksi kehitysprosessia jo alusta lähtien. Kehittäjänä kannattaa omalta osaltaan varmistaa saavutettavuuden huomioiminen, ottamalla käyttöön joitain matalan kynnyksen saavutettavuustestausmenetelmiä.

### 3.7 Seurantaviikko 7

Torstai 28.4.2022

Tänään tavoitteena palauttaa mieleen ennen sairaslomaa kesken jääneet tehtävät ja saattaa ne loppuun. Pidämme myös palaverin, jossa käsitellään mitä viesti- ja asiakirjapalvelu vielä vaatii, jotta se saadaan valmiiksi.

Sain tänään viimeisteltyä ennen sairaslomaa kesken jääneet tehtävät. Lisäksi viesti- ja asiakirjapalvelun kokous selkeytti seuraavia askeleita. Tietoevryllä oli lisäksi tiimipalaveri, jossa käsiteltiin esimerkiksi kehittäjien mieleen nousseita ajatuksia ja kysymyksiä alkuvuonna tapahtuneesta organisaatiomuutoksesta.

Perjantai 29.4.2022

Tänään tavoitteena saada viestin lähetys -ominaisuus toimimaan staging-ympäristössä Angular-sovelluksen osalta. Lisäksi minulla on tänään Tietoevryllä haastattelu jotain markkinointi juttua varten.

Viestin lähetyksessä onkin vielä useita asioita, jotka täytyy yhtenäistää palvelimen kanssa. Nyt loppumetreillä on tullut siis vähän hankaluuksia siitä, että etukäteen ei ole ollut tietoa siitä mitä pitäisi ottaa huomioon, kun henkilö- ja yrittäjän eläkevakuutus -asiakkaan viestipalvelut yhdistetään projektin aikana. Nyt nämä asiat täytyy siis suunnitella tehdessä.

Viikkoanalyysi

Tällä viikolla tiimillämme Tietoevryllä oli tiimipalaveri, jossa keskusteltiin organisaatiomuutoksesta ja työtyytyväisyydestä. Sen johdosta kirjoitan viikkoanalyysiin tutkimuksista ohjelmistokehittäjien työtyytyväisyyteen liittyen sekä siihen vaikuttavista tekijöistä. Lisäksi pohdin sitä, miten ne vertautuvat omiin kokemuksiini.

Yksi isoimmista kehittäjien kokemaan työtyytyväisyyteen vaikuttavista tekijöistä on hyvin toimivat tekniset työkalut ja prosessit (Graziotin and Fagerholm, 2019; Sadowski and Zimmermann, 2019; Storey *et al.*, 2021). Tämä on asia, jonka voin itsekin allekirjoittaa. Työskentely mielekkyyteen vaikuttaa erittäin negatiivisesti, jos esimerkiksi koodin integraatio ja tuotantoonvienti ei ole kunnolla automatisoitu. Myös kehitysympäristö ja siihen liittyvien ohjelmistojen olisi hyvä olla asianmukaisia

projektin koko ja käytössä olevat teknologiat huomioon ottaen. Muita tärkeiksi koettuja asioita ohjelmistokehittäjien työssä ovat esimerkiksi tyytyväisyys esihenkilöön, koettu tuottavuus, tiimin kulttuuri ja palkitseminen. Niillä on myös melko korkea korrelaatio todelliseen koettuun tyytyväisyyteen.

Yleistäen suurin osa kehittäjistä on melko tyytyväisiä omaan työhönsä (Graziotin and Fagerholm, 2019; Sadowski and Zimmermann, 2019). Yleisimpiä syitä onnettomuuteen kehittäjien keskuudessa on jumittuminen ongelmanratkaisussa ja aikataulupaineet. Myös teknisiin työkaluihin ja prosesseihin liittyvät ongelmat tulevat usein esiin onnettomuutta lisäävinä tekijöinä. Lisäksi huonoon koodiin ja huonoihin ohjelmointikäytäntöihin törmääminen aiheuttaa useille onnettomuutta. Mielenkiintoista on myös, että kehittäjät useimmiten kertovat työtyytyväisyyteen negatiivisesti vaikuttaviksi asioiksi sellaisia asioita, joihin tiimien vetäjillä ja esihenkilöillä on mahdollisuuksia vaikuttaa.

Kehittäjien tyytyväisyyttä on liiketoiminnallisesti kannattavaa parantaa, sillä tyytyväisyyden puute aiheuttaa monia rahassa mitattavia ongelmia (Graziotin and Fagerholm, 2019; Sadowski and Zimmermann, 2019). Tällaisia ongelmia ovat esimerkiksi matalampi tuottavuus, aikataulusta myöhästyminen ja huonontunut koodin laatu. Onnettomat kehittäjät myös raportoivat töistä etääntymistä, kuten työtehtävien välttelyä. Tutkimustulokset myös viittaavat siihen suuntaan, että kehittäjien onnellisuus todella korreloi tuottavuuden kanssa. Tutkimuksiin tosin tietysti liittyy se ongelma, että onnellisuutta mitatessa kaikkia muuttujia on mahdotonta pitää vakiona. Käyttämieni lähteiden perusteella asiasta kuitenkin vaikuttaa olevan melko vahva tieteellinen konsensus.

Onnellisuus vaikuttaa myös moniin kehittäjien pehmeisiin taitoihin. Onnettomat kehittäjät kommunikoivat huonommin ja työskentelevät vähemmän organisoidusti (Graziotin and Fagerholm, 2019). Onnelliset kehittäjät taas tekevät parempaa yhteistyötä muiden kehittäjien kanssa, jakavat avoimesti tietoa ja auttavat muita kehittäjiä ratkaisemaan heidän omia ongelmiaan.

Onnettomat kehittäjät myös kärsivät useammin mielenterveyden häiriöistä. Tunteet, kuten ahdistus, stressi, huono itseluottamus, tunne masentuneisuudesta ja pelko tuomituksi tulemisesta olivat yleisempiä onnettomuutta tuntevilla kehittäjillä.

Itse olen huomannut, että jos ohjelmoidessa törmää huonoon ja sotkuiseen koodiin se aiheuttaa toivottomuuden tunnetta. Se tuntuu siltä, että vaikka itse tekisi kuinka hyvää työtä, niin koko projektin toimintatapoihin ei pysty lähes mitenkään vaikuttamaan. Lisäksi esimerkiksi kohtuuttoman tiukka aikataulu, testien ajamisen ja tuotantoonviemisen prosessin puutteet aiheuttavat itselleni ylimääräistä stressiä. Lyhyellä tähtäimellä nämä ongelmat eivät toki ole kasvanut itselle isoiksi ongelmiksi. Kuitenkin jos minulle tarjottaisiin uutta projektia tai työpaikkaa, jossa tietäisin, että samoja ongelmia ei esiinny, voisi kiinnostus olla suurta.

Scrum-viitekehystä soveltavissa projekteissa olisi mielestäni äärimmäisen tärkeää tarttua esimerkiksi sprintin retrospektiivipalavereissa kehittäjien esille tuomiin ongelmiin välittömästi konkreettisin toimin, jotta kehittäjille ei ala kasautua toivottomuuden tunnetta. Sellainen toimintatapa voisi merkittävästi vähentää koettuja ongelmia. On siis myös työnantajien ja konsultteja palkkaavien yrityksien etu, että näitä kehittäjien kokemia ongelmia pyritään korjaamaan. Pelkästään se, että kehittäjien vaihtuvuus olisi mahdollisimman vähäistä, toisi jo itsessään hyötyjä (Sadowski and Zimmermann, 2019). Sen lisäksi hyötyjä olisivat tietenkin jo aiemmin mainitut asiat, kuten parantunut tuottavuus ja toimivampi yhteistyö kehittäjien kesken (Graziotin and Fagerholm, 2019).

### 3.8 Seurantaviikko 8

Maanantai 2.5.2022

Tavoitteena hoitaa pari tehtävää liittyen virheisiin, joita tapahtuu staging-ympäristössä. Lisäksi viesti- ja asiakirjapalvelun yhtenäistämistä viimeisimpien palvelin muutosten kanssa.

Sain virhetilanteet ratkottua. Kyseessä oli tilanne, jossa käyttöliittymällä näytettiin kohdat tiettyjen eläkkeiden aikaisimmalle mahdolliselle hakemuspäivämäärälle, vaikka nykyisellä käyttäjällä näitä päivämääriä ei ole. Lisäksi aiemmin mainittu päivämäärä ja aikaisin eläkkeen maksupäivämäärä oli käyttöliittymällä väärässä järjestyksessä, joka johtui alun perin epäselvistä JSON-kenttien nimistä. Sovittiin jo aiemmin niiden uudelleen nimeämisesti ymmärrettävämpään muotoon, jotta kentän tarkoitusta ei tarvitse jatkossa arvailla.

Tiistai 3.5.2022

Tänään täytyy keskustella viesti- ja asiakirjapalvelua kehittävän palvelinkehittäjän kanssa, jotta saataisiin esimerkiksi viestin lähetys -toiminnallisuus testattavaksi mahdollisimman pian staging-ympäristöön. Siihen liittyviä muutostöitä siis Angular-sovelluksen puolella tänään.

Sain paljon aikaiseksi sen jälkeen, kun sovittiin palvelinkehittäjän kanssa tarvittavista muutoksista. Nyt Angular-sovelluksen pitäisi kommunikoida paremmin yhteen muuttuneen palvelinympäristön kanssa, mutta tätä ei ole vielä testattu staging-ympäristössä.

Keskiviikko 4.5.2022

Tänään jälleen keskustelua palvelinkehittäjän kanssa ja Angular-sovelluksen ja Java-palvelimen mukauttamista toisiinsa sopiviksi. Tavoitteena saada viestin lähetyksen toiminnallisuus mahdollisimman pian testattavaksi. Lisäksi tänään designtiimin viikkopalaveri, jossa käytiin läpi seuraavassa design systeemin komponenttikirjaston versiossa tulevia parannuksia ja korjauksia. Kirjoitan design systeemeistä lisää viikkoanalyysissä.

Jälleen hyvää konkreettista edistystä viesti- ja asiakirjapalvelussa. Saatiin vihdoin viestejä siirtymään Angular-sovellukselta palvelimelle. Siinä sivussa olen myös korjaillut huomaamiani virheitä palvelussa. Seuraavaksi täytyy vielä luoda mekanismi, jolla tunnistaa eri asiakastyyppien viestiketjut toisistaan. Asiakastyypeillä tarkoitan tässä tapauksessa henkilöasiakkaita ja YEL- eli yrittäjän eläkevakuutus -asiakkaita.

Torstai 5.5.2022

Tänään heti aamusta sovittiin palvelinkehittäjän kanssa tavasta, miten erotellaan eri asiakastyyppien viestiketjut, jotta palvelin voi ohjata viestit oikeaan paikkaan. Sovittiin, että palvelin lisää tietokannasta tulevaan viestiketjun tunnisteeseen ylimääräisen merkin, joka kertoo kumman asiakastyypin viestiketju, on kyseessä. Suurin osa muutoksista on luultavasti palvelimen päässä, mutta Angular-sovelluksessa täytyy muuttaa viestiketjun tunnisteen tietotyyppi numerosta merkkijonoksi ja korjata mahdolliset sivuvaikutukset viesti- ja asiakirjapalvelussa. Lisäksi teen testipalvelimelle joitain muutoksia, jotta sillä voi testata Angular-sovellusta tuotantoa paremmin vastaavaa dataa vasten.

Taas hyvää edistystä viesti- ja asiakirjapalvelun kanssa. Nyt toistaiseksi viestin lähettämisen mahdollistamiseksi täytyy muutoksia enää tehdä vain palvelimen päässä. Olen aloittanut uutta tehtävää, joka on mahdollistaa asiakkaan henkilökunnalle viestiketjujen sulkemisen uusilta viesteiltä. Lisäksi korjasin useita latausindikaattoreita, jotka olivat tähän asti kadonneet ennen kuin operaatio, jota latausindikaattori odottaa on valmis.

Perjantai 6.5.2022

Tänään mahdollisia lyhyitä palavereita palvelin kehittäjän kanssa, jos on tarvetta muutoksilla Angular-sovelluksen päässä. Muuten teen tänään viestiketjun sulkemisen ominaisuutta.

Sain viestiketjun sulkemistoiminnallisuuden onnistuneesti toteutettua. Se oli hieman odotettua yksinkertaisempaa.

Viikkoanalyysi

Tällä viikolla, kuten joka viikko oli jälleen designtiimin viikkopalaveri. Tällä kertaa käsiteltiin design systeemiin perustuvan komponenttikirjaston seuraavan version tuomia parannuksia. Tästä heräsi idea ottaa viimeisen viikkoanalyysin aiheeksi design systeemit. Design systeemille ei ole muuta vakiintunutta suomenkielistä termiä.

Design systeemit ovat syntyneet organisaatioiden tarpeesta luoda yhtenäistä ulkoista ja sisäistä viestintää koko organisaation laajuisesti (Koivisto, 2019; Vizard, 2020). Design systeemien yksi

hyöty on myös, ettei kehittäjien tarvitse tehdä yleisiin käyttöliittymätarpeisiin liittyvää toistuvaa työtä (Gu, 2021). Käyttöliittymäkomponentit voidaan siis koodin tasolla viimeistellä esimerkiksi saavutettavuus huomioon ottaen, eikä näitä perusasioita tarvitse tehdä aina uudestaan. Design systeemin avulla voidaan myös varmistaa, että yrityksen visuaalinen ilme on responsiivinen ja toimii joustavasti erilaisissa konteksteissa (Vizard, 2020). Näitä ovat esimerkiksi erikokoiset näytöt, eri käyttöjärjestelmät, erilaiset syöttölaitteet ja eri käyttäjäryhmät.

Eri design systeemit voivat olla suunniteltu hyvinkin erilaisista näkökulmista, mutta yleensä kaikilla on jokin sisäinen tai julkinen verkkosivu, jossa design systeemin periaatteet on dokumentoitu (Koivisto, 2019; Vizard, 2020; Gu, 2021). Lisäksi design systeemit sisältävät usein ohjeita esimerkiksi saavutettavuuteen liittyen, sekä linkkejä muihin samaan design systeemiin liittyviin materiaaleihin. Nämä muut materiaalit ovat usein kokoelma uudelleenkäytettäviä design resursseja kuten käyttöliittymäkomponentteja sekä vastaava kokoelma sovelluskehityksessä käytettäviä koodipohjaisia käyttöliittymäkomponentteja esimerkiksi Git-repositoriossa. Näitä koodipohjaisia komponentteja on usein toteutettu eri kielillä ja kirjastoilla eri alustoille sopiviksi, kuten iOS, Android ja web.

Design systeemit ovat olleet viimeisen noin vuosikymmenen ajan erittäin pinnalla esimerkiksi teknologiayrityksissä ja käyttöliittymäalalla laajemminkin (Koivisto, 2019; Vizard, 2020). Design systeemit voidaan nähdä kehittyneenä versiona yritysten graafisista ohjeistuksista. Tällaisista uudelleenkäytettävistä palasista designia esimerkiksi arkkitehtuurin alalla on kuitenkin ollut kirjallisuutta jo 1970-luvulla (Koivisto, 2019; Vizard, 2020). Lisäksi esimerkiksi painettu kirja voidaan nähdä yhtenä ensimmäisistä esimerkeistä systemaattisesti uudelleen toistettavasti käyttöliittymästä (Koivisto, 2019). Design systeemien erilaiset edeltäjät ovat siis olleet olemassa jo hyvinkin kauan ennen kuin käsite design systeemeistä on muodostunut. Sittemmin niistä on tullut valtavirtaa jo kaiken kokoisissa yrityksissä useilla eri aloilla sekä julkisissa ja kolmannen sektorin organisaatioissa (Vizard, 2020; Gu, 2021).

Asiakasyrityksessä on siis myös käytössä design systeemi, johon liittyy Figmassa dokumentoidut graafiset ohjeistukset ja design resurssit, joilla luodaan käyttöliittymistä prototyyppejä. Figma on pääasiassa verkossa käytettävä ohjelma, jolla voi luoda vektorigrafiikkaa (Figma, no date). Se on suunniteltu juuri käyttöliittymien suunnitteluun ja prototypiointiin. Tämän Figmassa olevan dokumentaation lisäksi design systeemin käyttöliittymäkirjasto, eli koodipohjaiset komponentit on julkaistu yrityksen sisäisenä NPM-pakettina (Node package manager). Tämä NPM-paketti sisältää myös dokumentaation komponenttien koodista ja niiden tarjoamista rajapinnoista, joka on selattavissa sisäisillä verkkosivulla. Asiakasyrityksen käyttämä design systeemi on siis vain sisäisessä käytössä. Se kuitenkin perustuu lähdekoodiltaan Googlen julkiseen Material Design nimiseen design systeemiin.

Monet design systeemit ovat kuitenkin suunniteltu avoimesti saatavilla olevaksi ja useiden eri tahojen käytettäviksi erilaisiin tarpeisiin (Koivisto, 2019; Vizard, 2020). Brändille spesifit design systeemit ovat kuitenkin lähtökohtaisesti suunniteltu vain tietyn brändin ja organisaation käyttöön ja ne ovat suljettu käytöstä organisaation ulkopuolisilta tahoilta. Näin on siis myös asiakasyrityksessä. Toisaalta jotkut design systeemit ovat avoimia organisaation ulkopuolisille tahoille. Esimerkkinä tästä on Spotify, jolla on ohjeistuksia Spotify brändin käyttämiseen ulkopuolisille kehittäjille, jotka integroivat Spotifyn palveluita omiin sovelluksiinsa (Spotify, no date; Koivisto, 2019). Spotify on siis tarkemmin sanottuna tehnyt julkiseksi vain tietyt rajatut osat sen käyttämästä design systeemistä, Encoresta, ja tätä osaa kutsutaan Spotifyn graafiseksi ohjeistukseksi (Spotify, no date; Koivisto, 2019; Kaiser, Posniak and Bent, 2020). On olemassa myös täysin julkisia ja vapaasti käytettäviä design systeemejä, kuten IBM:n Carbon tai Googlen Material Design, jotka ovat julkisia myös lähdekoodiltaan (Google, no date m; IBM, no date; Koivisto, 2019).

Asiakasyrityksessä design systeemiä ylläpitää koodin tasolla kehittäjät eri tiimeistä. Käytännössä kuka vain kehittäjä on tervetullut tekemään korjauksia design systeemin komponenttikirjaston koodiin, jos esimerkiksi havaitsee siinä ongelmia. Designtiimissä on kuitenkin myös kehittäjiä edustamassa eri kehitystiimeistä, jotka ovat mukana designtiimin viikkopalaverissa. Nämä edustajat, joihin lukeudun myös itse, on pääasiassa vastuussa komponenttikirjaston koodin ylläpidosta. Komponenttikirjastoon liittyvät tehtävät siis jaetaan lähtökohtaisesti viikkopalavereissa näille kehittäjille. Olen välillä ajatellut, että asiakasyrityksen designtiimillä olisi hyvä olla kehittäjiä, jotka keskittyvät täysipäiväisesti vain design systeemin komponenttikirjaston ylläpitämiseen. Nykyisellään kehittäjät, jotka ovat mukana designtiimin viikkopalaverissa, ovat usein oman tiiminsä tehtävistä niin kiireisiä, että eivät ehdi tehdä komponenttikirjastoon liittyviä tehtäviä. Lisäksi eri tiimien Scrum Masterit eivät usein itsekään tunnu haluavan antaa omien kehittäjien aikaa komponenttikirjaston ylläpitämiseen. Siihen tosin luultavasti vaikuttaa myös se, että projekti on lähestymässä tuotantoonvientiä ja kevään aikana projektissa on ollut sen takia poikkeuksellisen kiire.

Mielenkiintoista oli lukea, että Spotifylla oli kokeiltu kuvailemaani design systeemin kehittämiseen keskittynyttä tiimiä, mutta se oli koettu siellä toimimattomaksi (Kaiser, Posniak and Bent, 2020). Spotifyn artikkelissakin kuitenkin mainitaan, että useissa yrityksissä tällainen toimintatapa on todettu toimivaksi. Spotifyn tapauksessa keskitetyn kehittäjätiimin koettiin kuitenkin aiheuttavan pullonkaulan. Tämä johtui siitä, että design systeemiä piti kehittää niin monille erilaisilla alustoille, jopa älyjääkaapeille. Lisäksi Spotifylla tiimejä kannustetaan työskentelemään hyvin autonomisesti ja keskitetty tiimi design systeemille ei täysin istunut tähän toimintamalliin (Kaiser, Posniak and Bent, 2020). Spotifylle ehti kehittyä eri tiimeissä 22 erilaista design systeemiä. Sen sijasta, että olisi yritetty puskea tuota vastaan Spotify kuitenkin päätti omaksua tuon ominaisuutena heidän työskentelytavoissaan ja kehitti heidän nykyisen design systeemin Encoren (Kaiser, Posniak and Bent,

2020). Spotify kutsuu Encorea design systeemien design systeemiksi, kokoelmaksi design systeemejä, korostaen Spotifyn tiimien autonomiaa.

Spotifylla keskitetty tiimi hallinnoimaan design systeemin kehitystä ei siis ollut pidemmän päälle toimiva ratkaisu. Näistä kokemuksista lukeneena ajattelen, että ehkä tässä asiakasprojektissa voisi olla hyötyä jonkinlaisesta hybridimallista niin, että designtiimiä täydentää eri tiimeissä olevat kehittäjät, jotka mahdollisuuksien mukaan myös tekevät komponenttikirjastoon liittyviä tehtäviä. Lisäksi designtiimissä olisi kuitenkin myös esimerkiksi kaksi kehittäjää, jotka takaisivat, että komponenttikirjastoon liittyvät tehtävät eivät jää heti kiireen tullen tekemättömäksi.

# 4 Pohdinta

Olen ollut koko päiväkirjaopinnäytetyöprosessin ajan töissä samassa projektissa nuorempana front-end kehittäjänä, ja työtehtäväni ovat pysyneet lähtötilanteen kanssa samanlaisina. Tässä pohdinnassa erityisesti vertaan nykytilannettani lähtötilanteeseeni ja käyn läpi oppimiani asioita. Lopuksi pohdin, miten päiväkirjaopinnäytetyö soveltui minulle.

Päiväkirjaopinnäytetyön alussa perehdyin syvällisemmin töissä tekemiini teknologisiin ratkaisuihin. Opinnäytetyöprosessin aikana olen sisäistänyt projektissa käytössä olevia teknologioita ja oppinut niistä uusia ominaisuuksia. Teknologioihin liittyen erityisen paljon olen oppinut lisää Angularista ja RxJS:stä. Se on ollut odotettavissa, sillä ne ovat myös olleet projektin alusta lähtien itselleni vähiten tutut teknologiat. Myös TypeScript tuli prosessin aikana entistä tutummaksi. Olen oppinut paljon käteviä TypeScriptiin liittyviä toiminnallisuuksia, joista tulee olemaan hyötyä tulevaisuudessa myös muissa projekteissa.

Lähtötilanteen kartoituksen yhteydessä en vielä tiennyt, että työtehtäväni tulevat olemaan projektin kannalta hyvin kriittisiä. Tekemäni toiminnallisuudet ovat palvelun tuotantoonviennin kannalta niin oleellisia, että niiden on pakko valmistua ennen sitä tai projektin tuotantoonvienti ajallaan estyy. Menin projektiin nuorempana ohjelmistokehittäjänä, joten ajattelin vastuuni olevan aika rajallista. On erittäin motivoivaa työskennellä tietäen työn olevan hyvin oleellista projektin ajallaan valmistumisen kannalta. Mielestäni tämä kertoo myös syvästä luottamuksesta tiimin kaikkiin jäseniin.

Toisaalta mikäli projektissa olisi käytössä kehittyneemmät menetelmät, kuten jatkuva integraatio ja tuotantoonvienti, niin projekti etenisi paljon varmemmin. Jatkuva integraatio ja tuotantoonvienti olisikin ollut myös erittäin kiinnostava viikkoanalyysin aihe. Sitä aihetta kiinnostaisi tutkia jatkossa lisääkin. Esimerkiksi sen vaikutuksista projektin kustannustehokkuuteen ja aikataulussa pysymiseen voisi olla mielenkiintoista tietää lisää.

Olen onneksi onnistunut melko hyvin pitämään aikatauluun ja projektin valmistumiseen liittyvät huolet hallinnassa. Koen, että tällaiset asiat eivät ole minun päätettävissäni, joten olen vain pyrkinyt keskittymään tekemään oman työni mahdollisimman hyvin.

Analyysien kirjoittaminen tuki myös oppimista. Niitä kirjoittaessani huomasin, että kehityin ja keksin parempia teknisiä toteutuksia, kuin mitä olin töissä alun perin toteuttanut. Lisäksi huomasin analyysejä kirjoittaessani jopa joitain epäjohdonmukaisuuksia koodissani ja näin opinnäytetyön kirjoittaminen tuki niiden korjaamista. Luin analyysejä varten paljon Angularin ja RxJS:n dokumentaatiota, joten myös dokumentaation lukutaitoni kehittyi ja totuin entistä paremmin teknisen dokumentaation

lukemiseen. Analyysien kirjoittamiseen meni kuitenkin opinnäytetyöprosessin alussa joitakin viikkoja tottua. Prosessin loppua kohti sain niitä kirjoittaessani erityisen paljon lisätietoa aiheista, joihin ei välttämättä olisi tullut säännöllisesti työpäivien aikana muuten perehdyttyä.

Tietoa etsiessäni ja lähteitä lukiessani opin kiinnittämään huomion tutkimuksien ja muiden tekstien oleellisiin asioihin. Opin myös hakemaan tietoa niin, että löydän ongelmani kannalta relevanttia sisältöä. Esimerkiksi oikeiden ja aiheeseen sopivien tietokantojen käyttäminen hakujen lähteenä oli oleellista tutkimuksien ja tieteellisten artikkeleiden löytämiseksi. Pyrin tietoisesti siihen, että lähteitä olisi monipuolisesti. Lähteistä erityisesti tieteelliset artikkelit olivat itselleni tekstinä vieraampia ja niiden hyödyntäminen oli tämän vuoksi työläintä. Tutuimpia itselleni olivat erilaiset tekniset dokumentaatiot, joihin oli kiinnostavaa tutustua syvemmin tämän opinnäytetyön puitteissa.

Kirjoitin saavutettavuudesta kahdessa viikkoanalyysissä. Näiden avulla opin saavutettavuudesta paljon uutta ja ymmärsin sen, että se on tärkeä ottaa osaksi ohjelmistokehitystä mahdollisimman aikaisessa vaiheessa. Myös erilaiset saavutettavuustestauksen menetelmät tulivat tutuiksi ja näitä tulen varmasti hyödyntämään töissä tulevaisuudessa. Esimerkiksi ruudunlukijan käyttäminen ja näppäimistönavigointi ovat jo nyt säännöllisesti käytössä töitä tehdessä.

Opinnäytetyön alussa nostin esiin myös haluni kehittyä työskentelyssä ja kommunikoinnissa ketteriä menetelmiä hyödyntävässä tiimissä. Scrumin mukaisen ohjelmistokehityksen periaatteet tulivat tutummaksi, sillä perehdyin yhdessä viikkoanalyysissä Scrum-oppaaseen ja vertasin tämän oppaan toimintamallia oman tiimimme toimintaan. Kävi ilmi, että tiimimme ei noudata Scrumia täysin. Scrumiin tarkemmin perehtyminen kuitenkin tukee hyvin jokapäiväistä työtäni, sillä ymmärrän nyt paremmin esimerkiksi erilaisten roolien ja seremonioiden merkityksen. Osaan myös paremmin analysoida ja pohtia mahdollisia parannuksia toimintatapoihin. Scrumiin perehtyminen edesauttoi suoraan kehittymistäni ketterässä tiimissä työskentelyssä.

Koen päiväkirjaopinnäytetyön olleen itselleni oikea opinnäytetyömalli. Se pakotti perehtymään erilaisiin työssäni merkityksellisiin aiheisiin erityisen tarkasti ja analysoimaan omaa ja tiimin työskentelyä, sekä pohtimaan mahdollisia parempia toimintamalleja. Opinnäytetyön työmäärä säilyi kohtuullisena kokopäivätyön ohessa. Tämän ansiosta pystyin käyttämään enemmän voimavaroja opinnäytetyön tekemiseen mahdollisimman ansiokkaasti ja sain opinnäytetyön tekemisestä enemmän hyötyjä irti.

On myös joitain asioita, mitkä eivät olleet niin isossa asemassa opinnäytetyöprosessin aikana, mutta joihin haluaisin kuitenkin perehtyä jatkossa vielä enemmän. Tällaisia aiheita ovat muun muassa erilaiset ohjelmistojen ja koodin testauksen menetelmät sekä niiden automatisointi jatkuvan

integraation ja jatkuvan tuotantoonviennin menetelmin. Konkreettisemmin myös hyvien testien kirjoittaminen jatkokehityksessä tapahtuvien virheiden havaitsemiseksi on asia, johon pystyn toivottavasti paneutua tulevaisuudessa nykyistä enemmän. Angular tukee ja sisältää myös kirjastoja erilaisten testien toteuttamiseen, joihin olisi varmasti hyödyllistä perehtyä. Erityisesti siksi, koska samoja kirjastoja voi käyttää myös muissa kuin Angular pohjaisissa JavaScript projekteissa. Lisäksi tuotantoonviennin automaatio on aihe, josta kiinnostaisi oppia enemmän.

Myös opinnäytetyöprosessin aikana pinnalla olleissa aiheissa on vielä todella paljon lisää kiinnostavaa opittavaa. Erityisesti jo aiemmin mainitsemani saavutettavuus on sellainen aihepiiri, josta haluaisin oppia lisää. Työpaikkani tarjoaa paljon erilaisia mahdollisuuksia itsensä kehittämiseen ja olen ajatellut hyödyntää näitä mahdollisuuksia oman osaamiseni kehittämisessä. Tietoevry tarjoaa lukuisia erilaisia rooleja teknologian parissa ja myös esimerkiksi tehtävää tai projektia vaihtamalla voisi olla mahdollisuus keskittyä erilaisiin itseä kiinnostaviin asioihin. Lisäksi jatko-opinnot tarjoavat kiinnostavia mahdollisuuksia itseni ja osaamisen kehittämiseen.

## Lähteet

Abuaddous, H.Y., Jali, M.Z. and Basir, N. (2016) “Web Accessibility Challenges,” *IJACSA) International Journal of Advanced Computer Science and Applications*, 7(10). Available at: www.ijacsa.thesai.org (Accessed: April 16, 2022).

Al-Mutawa, H.A. *et al.* (2014) “On the shape of circular dependencies in java programs,” *Proceedings of the Australian Software Engineering Conference, ASWEC*, pp. 48–57. doi:10.1109/ASWEC.2014.15.

Aluehallintovirasto (no date a) *Keitä digipalvelulaki velvoittaa? - Saavutettavuusvaatimukset*. Available at: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/soveltamisala-kuulummeko-lain-piiriin/ (Accessed: April 15, 2022).

Aluehallintovirasto (no date b) *Kenelle saavutettavuus on tärkeää? - Saavutettavuusvaatimukset*. Available at: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa/ (Accessed: April 15, 2022).

Aluehallintovirasto (no date c) *Saavutettavuuden lait ja standardit - Saavutettavuusvaatimukset*. Available at: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/ (Accessed: April 15, 2022).

Aluehallintovirasto (no date d) *Tietoa WCAG-ohjeistuksesta - Saavutettavuusvaatimukset*. Available at: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/ (Accessed: April 15, 2022).

Aluehallintovirasto (no date e) *Yleistä saavutettavuudesta - Saavutettavuusvaatimukset*. Available at: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/ (Accessed: April 15, 2022).

*Angular Reactive Templates with ngIf and the Async Pipe* (2022). Available at: https://blog.angular-university.io/angular-reactive-templates/ (Accessed: March 13, 2022).

Bai, A., Mork, H.C. and Stray, V. (2017) “A Cost-Benefit Analysis of Accessibility Testing in Agile Software Development Results from a Multiple Case Study,” *International Journal on Advances in Software* [Preprint].

Billiet, B. (2018) *RxJS best practices in Angular*. Available at: https://blog.strongbrew.io/rxjs-best-practices-in-angular/ (Accessed: March 19, 2022).

Cardozo, E.S.F. *et al.* (2010) “SCRUM and Productivity in Software Projects: A Systematic Literature Review.” doi:10.14236/EWIC/EASE2010.16.

Chenkie, R. (2017) *Angular Authentication: Using Route Guards*, *Medium*. Available at: https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3 (Accessed: March 22, 2022).

Fessenden, T. (2021) *Design Systems 101*. Available at: https://www.nngroup.com/articles/design-systems-101/ (Accessed: May 20, 2022).

Figma (no date) *Figma: the collaborative interface design tool.* Available at: https://www.figma.com/ (Accessed: May 8, 2022).

Gallardo, E.G. (no date) *RxJS Best Practices*, *2020*. Available at: https://betterprogramming.pub/rxjs-best-practices-7f559d811514 (Accessed: March 19, 2022).

Google (no date a) *Angular - AbstractControl*. Available at: https://angular.io/api/forms/AbstractControl (Accessed: March 27, 2022).

Google (no date b) *Angular - @angular/router*. Available at: https://angular.io/api/router#structures (Accessed: March 22, 2022).

Google (no date c) *Angular - AsyncPipe*. Available at: https://angular.io/api/common/AsyncPipe (Accessed: March 13, 2022).

Google (no date d) *Angular - FormControl*. Available at: https://angular.io/api/forms/FormControl (Accessed: March 27, 2022).

Google (no date e) *Angular - Glossary*. Available at: https://angular.io/guide/glossary (Accessed: March 13, 2022).

Google (no date f) *Angular - HttpClient*. Available at: https://angular.io/api/common/http/HttpClient (Accessed: March 13, 2022).

Google (no date g) *Angular - JsonPipe*. Available at: https://angular.io/api/common/JsonPipe (Accessed: March 13, 2022).

Google (no date h) *Angular - Observables compared to other techniques*. Available at: https://angular.io/guide/comparing-observables (Accessed: March 13, 2022).

Google (no date i) *Angular - The RxJS library*. Available at: https://angular.io/guide/rx-library (Accessed: March 13, 2022).

Google (no date j) *Angular - Transforming Data Using Pipes*. Available at: https://angular.io/guide/pipes (Accessed: March 13, 2022).

Google (no date k) *Angular - What is Angular?* Available at: https://angular.io/guide/what-is-angular (Accessed: March 12, 2022).

Google (no date l) *Angular Material UI component library*. Available at: https://material.angular.io/ (Accessed: April 16, 2022).

Google (no date m) *Material Design*. Available at: https://material.io/ (Accessed: May 8, 2022).

Google (no date n) *RxJS - Introduction*. Available at: https://rxjs.dev/guide/overview (Accessed: March 13, 2022).

Graziotin, D. and Fagerholm, F. (2019) “Happiness and the Productivity of Software Engineers,” *Rethinking Productivity in Software Engineering*, pp. 109–124. doi:10.1007/978-1-4842-4221-6_10.

Gu, Q. (2021) “Design System as a Service Providing in-house design systems from the perspective of design system creators.”

IBM (no date) *Carbon Design System*. Available at: https://carbondesignsystem.com/ (Accessed: May 8, 2022).

*JSON* (no date). Available at: https://www.json.org/json-en.html (Accessed: March 19, 2022).

Kaiser, G., Posniak, M. and Bent, S. (2020) *Reimagining Design Systems at Spotify | Spotify Design*. Available at: https://spotify.design/article/reimagining-design-systems-at-spotify (Accessed: May 8, 2022).

Koivisto, J. (2019) “The Building Blocks Of a UI Sandwich Examining the Efficiency and the Customization of Design Systems.” Available at: www.aalto.fi (Accessed: May 8, 2022).

Lokalise (no date) *Lokalise API 2.0 documentation*. Available at: https://app.lokalise.com/api2docs/curl/#resource-keys (Accessed: April 9, 2022).

Luukkainen, M. and Ilves, K. (2021) *Helsingin Yliopiston Ohjelmistotuotanto 2021*. Available at: https://ohjelmistotuotanto-hy.github.io/osa3/ (Accessed: April 11, 2022).

Sergi Mansilla (2018) *Reactive Programming with RxJS 5*. Available at: https://learning.oreilly.com/library/view/reactive-programming-with/9781680505528/?ar= (Accessed: March 13, 2022).

Microsoft (no date) *TypeScript: JavaScript With Syntax For Types.* Available at: https://www.typescriptlang.org/ (Accessed: May 19, 2022).

Mozilla (no date a) *Array.prototype.forEach() - MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach (Accessed: March 19, 2022).

Mozilla (no date b) *HTTP*. Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP (Accessed: May 19, 2022).

Mozilla (no date c) *Truthy - MDN Web Docs Glossary: Definitions of Web-related terms*. Available at: https://developer.mozilla.org/en-US/docs/Glossary/Truthy (Accessed: May 20, 2022).

Mozilla (no date d) *WAI-ARIA basics*. Available at: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics (Accessed: April 27, 2022).

Mundra, A., Misra, S. and Dhawale, C.A. (2013) “Practical scrum-scrum team: Way to produce successful and quality software,” *Proceedings of the 2013 13th International Conference on Computational Science and Its Applications, ICCSA 2013*, pp. 119–123. doi:10.1109/ICCSA.2013.25.

Radigan, D. (no date) *What are story points and how do you estimate them?* Available at: https://www.atlassian.com/agile/project-management/estimation (Accessed: May 20, 2022).

Red Hat (no date) *What is an API?* Available at: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces (Accessed: May 20, 2022).

RxJS Team (no date) *RxJS - withLatestFrom*. Available at: https://rxjs.dev/api/operators/withLatestFrom (Accessed: March 27, 2022).

Cory Rylan (2017) *Angular Async Data Binding with ngIf and ngElse - Angular 13 | 12*. Available at: https://coryrylan.com/blog/angular-async-data-binding-with-ng-if-and-ng-else (Accessed: March 13, 2022).

Sadowski, C. *et al.* (2018) “Modern code review: A case study at google,” *Proceedings - International Conference on Software Engineering*, pp. 181–190. doi:10.1145/3183519.3183525/FORMAT/PDF.

Sadowski, C. and Zimmermann, T. (2019) “Rethinking Productivity in Software Engineering.” doi:10.1007/978-1-4842-4221-6.

Schwaber, K. and Sutherland, J. (2020) “The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game.”

Sean G. Wright (2016) *Do I need to unsubscribe from an observable?* Available at: https://www.seangwright.me/blog/development/unsubscribe-angular-2-http-observables/ (Accessed: March 27, 2022).

Spanner, C. (no date) *Scrum of scrums*. Available at: https://www.atlassian.com/agile/scrum/scrum-of-scrums (Accessed: May 20, 2022).

Spotify (no date) *Design Guidelines | Spotify for Developers*. Available at: https://developer.spotify.com/documentation/general/design-and-branding/ (Accessed: May 8, 2022).

Storey, M.A. *et al.* (2021) “Towards a Theory of Software Developer Job Satisfaction and Perceived Productivity,” *IEEE Transactions on Software Engineering*, 47(10), pp. 2125–2142. doi:10.1109/TSE.2019.2944354.

Taylor, W. (2019) *RxJS: Hot, Cold, Finite, Infinite, Unicast and Multicast Observables explained*. Available at: https://www.willtaylor.blog/rxjs-observables-hot-cold-explained/ (Accessed: March 27, 2022).

Tietoevry (2022a) “Tietoevry 2021 sustainability report.”

Tietoevry (2022b) *Tietoevry 2021 vuosikertomus*.

Troncone, B. (no date) *RxJS Primer - Learn RxJS*. Available at: https://www.learnrxjs.io/learn-rxjs/concepts/rxjs-primer (Accessed: March 19, 2022).

Valtiovarainministeriö (2019) *Digitalisaatiota edistävä uusi lainsäädäntö muokkaa viranomaisten toimintaa, mutta kehitettävää on edelleen*. Available at: https://valtioneuvosto.fi/-/10623/digitalisaatiota-edistava-uusi-lainsaadanto-muokkaa-viranomaisten-toimintaa-mutta-kehitettavaa-on-edelleen (Accessed: April 24, 2022).

Vardanyan, A. (2019) *Rx.js: Best Practices*. Available at: https://medium.com/angular-in-depth/rxjs-best-practices-6a3b095ffb04 (Accessed: March 19, 2022).

Vinje, T. (2021) *Putting trust to the test – TietoEVRY employees choose hybrid work | Tietoevry*. Available at: https://www.tietoevry.com/en/blog/2021/11/putting-trust-to-the-test--tietoevry-employees-choose-hybrid-work/ (Accessed: April 9, 2022).

Vizard, L. (2020) *Introduction & Getting Started With Design Systems*. Available at: https://xd.adobe.com/ideas/principles/design-systems/introduction-to-design-systems/ (Accessed: May 8, 2022).

W3C (no date a) *Easy Checks – A First Review of Web Accessibility*. Available at: https://www.w3.org/WAI/test-evaluate/preliminary/#title (Accessed: April 26, 2022).

W3C (no date b) *Evaluating Web Accessibility Overview*. Available at: https://www.w3.org/WAI/test-evaluate/ (Accessed: April 26, 2022).

W3C (no date c) *How to Meet WCAG (Quickref Reference)*. Available at: https://www.w3.org/WAI/WCAG21/quickref/ (Accessed: April 26, 2022).

W3C (no date d) *WAI-ARIA Overview*. Available at: https://www.w3.org/WAI/standards-guidelines/aria/ (Accessed: April 27, 2022).

W3C (no date e) *WCAG 2 Overview*. Available at: https://www.w3.org/WAI/standards-guidelines/wcag/#intro (Accessed: April 15, 2022).

Wang, H. and Zhou, H. (2012) “Basic design principles in software engineering,” *Proceedings - 4th International Conference on Computational and Information Sciences, ICCIS 2012*, pp. 1251–1254. doi:10.1109/ICCIS.2012.91.

Wunder (no date) *Semanttinen HTML*. Available at: https://wunder.io/fi/wunderpedia/saavutettavuus/saavutettava-kayttoliittyma-ui/semanttinen-html/ (Accessed: April 27, 2022).