LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

OHJELMISTOKEHITTÄJÄN PÄIVÄKIRJAOPINNÄYTETYÖ

LAHDEN AMMATTIKORKEAKOULU
Insinööri (AMK)
Tieto- ja viestintätekniikka
Ohjelmistotekniikka
Syksy 2019
Pyry Heiskanen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Heiskanen, Pyry	Opinnäytetyö, AMK	Syksy 2019
	Sivumäärä	
	52	

Työn nimi

Ohjelmistokehittäjän päiväkirjaopinnäytetyö

Tutkinto

Insinööri (AMK), tieto- ja viestintätekniikka, ohjelmistotekniikka

Tiivistelmä

Päiväkirjaopinnäytetyö toteutettiin ohjelmistokehittäjän näkökulmasta startup-yrityksessä. Opinnäytetyö kuvaa ohjelmistokehittäjän päivittäisiä tehtäviä ja työskentelyä. Päiväkirjaopinnäytetyö koostuu kymmenestä seurantaviikosta, joiden aikaista työtehtäviä, ongelmia ja työssä tapahtunutta kehitystä on kuvattu päivätasolla. Päivittäisen seurannan lisäksi jokaisen viikon tapahtumia, kehitystä ja ongelmia pohditaan syvemmin viikoittaisissa viikkoanalyyseissa. Viikkoanalyyseissa käsitellään työtehtäviä ja esitellään sekä tutkitaan niissä käytettyjä teknologioita ja menetelmiä erilaisia lähteitä hyödyntäen.

Opinnäytetyö toteutettiin suomalaisen terveysteknologia-alan startup-yrityksessä. Seurantajakson aikainen työskentely tapahtui osana yrityksen teknologiatiimiä, joka on vastuussa yrityksen tuotteiden ja palvelujen ohjelmistokehityksellisitä tarpeista. Opinnäytetyön tekijän työnkuvaan ohjelmistokehittäjänä kuului enimmiltä osin erilaisten selainpohjaisten sovellusten kehittäminen, päivittäminen ja ylläpitäminen nykyaikaisia teknologioita hyödyntäen.

Opinnäytetyön seurantajakson ajalle asetettiin tavoitteita päivä- ja viikkotasolla. Tavoitteet koostuivat työtehtävien kokonaisuuksista ja niiden tarkoitus oli kehittää opinnäytetyön kirjoittajan ammatillisia tietoja ja taitoja.

Kokonaisuutena tämä päiväkirjaopinnäytetyö on dokumentti, joka kuvaa kirjoittajan seurantajakson aikaista työskentelyä ja sen myötä tapahtuvaa ammatillista kehitystä. Opinnäyte kuvaa myös taitoja ja tekniikoita, joita ohjelmistokehittäjä työssään tarvitsee ja käyttää.

Asiasanat

Ohjelmistokehitys, Angular, TypeScript, React, agile

Abstract

Author(s)	Type of publication	Published
Heiskanen, Pyry	Bachelor's thesis	Autumn 2019
	Number of pages	
	52	

Title of publication

Diary thesis of a software developer

Name of Degree

Bachelor of Engineering

Abstract

This thesis was written from the point of view of a software developer working in a startup company. The thesis consists of a description of the daily work of the software developer during a span of ten weeks. The day-to-day objectives, goals, challenges and professional development of the software developer were observed and documented through the ten weeks.

The thesis was written while working for a Finnish healthcare technology startup. The work was done as a part of the company's technology team. The technology team is responsible for the software development needs of all the products and services produced by the company. The software developer's daily work consists mainly of development and maintenance of browser-based applications using modern technologies.

The ten-week period included goals on a daily and weekly basis. The goals consist of collections of work-related objectives and the main purpose of these goals is to aid the professional development of the software developer.

The end-product of this thesis is a documentation of the professional development of a software developer. The resulting document also describes many of the technologies and skills required in the daily work of a software developer.

Keywords

software development, Angular, TypeScript, React, agile

KÄSITTEET

API = Application programming interface. Rajapinta, jota erilaiset ohjelmat käyttävät keskustellakseen keskenään.

Bugi = Ohjelmointivirhe.

JavaScript = Ohjelmointikieli, jota käytettään pääasiassa selainpohjaisten sovellusten kirjoittamiseen, mutta nykään myös muuhun ohjelmointiin.

NPM = Node Package Manager. Paketinhallintatyökalu erilaisten JavaScipt-moduulien helppoa hallintaa varten.

RxJS = JavaScript-kirjasto reaktiivista ohjelmointia varten.

URI = Uniform Resource Identifier. Merkkijono, joka sisältää tietyn tiedon sijainnin tai nimen.

SISÄLLYS

1 JOHDANTO

Tämä päiväkirjamuotoinen opinnäytetyö kuvaa työskentelyäni suomalaisessa startup-yrityksessä kymmenen seurantaviikon ajan. Opinnäytetyön tarkoituksena on seurata ja dokumentoida omaa päivittäistä työtäni ja henkilökohtaisen osaamiseni kehittymistä ohjelmistokehittäjän ammatillisissa tehtävissä. Seurantajakson aikainen dokumentointi koostuu päivittäisestä seurannasta ja työtehtävien kuvauksista sekä viikkoanalyyseista, joissa käsitellään viikon aikana kohtaamiani ongelmia ja tilanteita laajemmin, erilaisia kirjallisia lähteitä hyödyntäen. Seurantajakso on toteutettu syksyllä 2018.

Yritys, jossa tätä opinnäytetyötä tehdessäni työskentelen, on verrattain nuori suomalainen terveysteknologialan startup-yritys. Työskentelen yrityksessä ohjelmistokehittäjänä osana alle kymmenhenkistä teknologiatiimiä. Tehtäväni on pääasiallisesti erilaisten selainpohjaisten sovellusten suunnittelu, kehittäminen ja ylläpitäminen. Tämän lisäksi työtehtäviini kuuluu yhä enenevissä määrin backend-kehitykseen liittyviä tehtäviä. Sovelluksiin, joiden parissa työskentelen, kuuluu niin yrityksen sisäiseen käyttöön tarkoitettuja, kuin asiakkaidenkin päivittäiseen käyttöön tarkoitettuja sovelluksia ja työkaluja. Tässä opinnäytetyössä ei työnantajan toiveesta eritellä eri sovellusten nimiä tai yksityiskohtaisia käyttötarkoituksia sen tarkemmin, kuin mitä omien työtehtävieni kuvaaminen vaatii.

Tämän opinnäytetyön tarkoituksena on dokumentoida henkilökohtaista kehitystäni ohjelmistokehittäjänä työtehtäviä suorittaessani. Opinnäytetyössä seurataan ammatillista kehittymistäni osittain puhtaasti teknologiallisesti, mutta myös kehittymistäni osana asiantuntijoista koostuvaa tiimiä analysoidaan ja seurataan. Myös erilaisien ketterän kehityksen menetelmien hyödyntämistä työpaikalla seurataan koko seurantajakson ajan.

Päivittäinen seuranta koostuu suurelta osin päivittäisten työtehtävieni kuvaamisesta sekä kohtaamieni ongelmien ja niihin löytämieni ratkaisujen pintapuolisesta kuvaamisesta. Viikkoanalyyseissa sen sijaan pureudutaan ongelmiin sekä hyödyntämiini teknologioihin ja ratkaisuihin syvemmin erilaisia kirjallisia lähteitä hyväksi käyttäen.

Opinnäytetyön lähdemateriaali koostuu niin erilaisten seurantaviikon aikana hyödyntämieni teknologioiden ja ohjelmistokirjasten dokumentaatioista, kuin varsinaisista kirjallisuuden julkaisusta, artikkeleista ja verkkomateriaaleista.

2 LÄHTÖTILANTEEN KUVAUS

2.1 Oman työn analyysi

Työskentelen ohjelmistokehittäjänä osana startup-yrityksen teknologiatiimiä. Olen pääasiallisesti web- ja frontend-kehittäjä, joten vastuualueeni on web-sovelluksiemme kehittäminen ja ylläpitäminen. Pienessä yrityksessä roolini ei kuitenkaan ole aivan niin yksiselitteinen ja selkeästi jaoteltu, joten tarvittaessa teen ja autan myös muilla osa-alueilla.

Tiimin pääasiallisena web-kehittäjänä vastaan muutoksista, päivityksistä ja korjauksista yrityksen Angular- ja React-pohjaisiin web-sovelluksiin. Tehtävä vaatii tiivistä yhteistyötä backend-kehittäjien kanssa, sillä suurempia muutoksia täytyy suunnitella huolella käyttöliittymästä aina tietokantamuutoksiin asti. Saan vaikuttaa tehtyihin valintoihin ja ratkaisuihin sekä valita parhaaksi näkemäni keinot ja työkalut omien tehtävieni toteuttamiseksi.

Työnkuvani kannalta keskeisimmät teknologiat ovat Angular ja React, jotka ovat frameworkeja selainpohjaisten sovelusten tekemiseen. Nämä tuovat mukanaan myös tarpeen osata perinteisempiä web-kehityksen työkaluja, kuten CSS:ää, HTML:ää ja JavaScriptiä. Myös frameworkien kannalta merkittävien kirjastojen, kuten RxJS ja Redux, käyttö täytyy hallita. Tämän lisäksi tarvitsen työssäni Pythonin ja SQL-kielen osaamista sekä ymmärrystä palvelimiin liittyvistä teknologioista, esimerkiksi nginxistä, jotta pystyn tarvittaessa olemaan hyödyksi myös muilla osa-alueilla.

Työ on siis varsin itsenäistä ja itseohjautuvaa, mutta vaatii myös runsaasti tiimityöskentely- ja ryhmätyötaitoja. Teknologiatiimin koosta johtuen pääsen olemaan mukana myös backendin kehityksessä sekä osallistumaan testaukseen. Backend-puolen opetteleminen on ollut tärkeää ja mielenkiintoista, ja hoidan nykyään yksinkertaisempia tehtäviä myös backendiin, frontend työni ohella. Backend-kokemuksesta on hyötyä, sillä työskentelyni alkuvaiheessa backend-kehittäjien kiireisyys muodostui välillä pienimuotoiseksi hidasteeksi frontendin puolella.

Yrityksessä pyritään noudattamaan jonkinlaista ketterää ohjelmistokehitysmallia. Teemme noin kuukauden mittaisia sprinttejä, jotka suunnitellaan aina sprintin alussa yhteisessä sprinttipalaverissa. Pidämme myös joka maanantai yhteisen maanantaipalaverin, jossa käydään läpi edellisen viikon saavutetut tavoitteet ja määritellään jokaisen henkilökohtaiset tavoitteet alkavalle viikolle. Tiimin pienestä koosta johtuen päivittäisille kokouksille ei ole tarvetta, sillä kommunikointi on jatkuvasti sujuvaa ja runsasta tiimimme kesken. Pienestä tiimistä ja suuresta työmäärästä johtuen tilanteet kuitenkin muuttuvat usein nope-

asti, eikä sovituista sprinteistä aina pystytä pitämään kiinni uusien vaatimusten ja tarpeiden esiin tullessa. Tämä on asia, jota pyritään jatkuvasti kehittämään, kuten myös muita yrityksen sisäisiä prosesseja.

Työ startup-yrityksessä saattaa välillä olla stressaavaa ja kaoottista, sillä selkeitä prosesseja ja toimintatapoja ei aina ole juurikaan määritelty. Tarkoitus ei usein olekaan täydellisyyden tavoittelu, vaan pääosin julkaisukelpoisen sovelluksen valmiiksi saaminen, jota sitten kehitetään jatkossa sitä mukaa, kun esimerkiksi uusia tarpeita tai vikoja ilmenee. Koen startup-yrityksessä työskentelemisen oppimiseni kannalta kuitenkin hyvänä asiana, sillä koen työskentelyn vapaampana ja olosuhteiden pakottamana myös monipuolisempana, kuin mahdollisesti suuremmassa yrityksessä.

Kansainvälisestä teknologiatiimistä johtuen pääasiallisena työkielenä yrityksessä käytetään englantia. Erinomainen englannin kielen taito on siis edellytys sujuvan ja tehokkaan työskentelyn kannalta. Myös ohjelmistokehitysalan niin sanottu ammattisanasto on yleensä englanninkielistä, mikä myös osaltaan vaatii alalla työskenteleviltä hyvää englannin kielen osaamista.

Olen saanut koulutuksestani hyvän pohjan alalla menestymiseen, mutta koen toden teolla oppineeni alasta vasta työskenneltyäni ohjelmistokehittäjänä jonkin aikaa. Koulutus on tarjonnut tiettyjä perusperiaatteita ja -taitoja, jotka ovat olleet ensiarvoisen tärkeitä, mutta varsinaista kuvaa alalla työskentelystä ja sen vaatimuksista en vielä koulussa pystynyt hahmottamaan.

2.2 Sidosryhmät työpaikalla

Yrityksen pienestä koosta sekä matalasta organisaatiorakenteesta johtuen sidosryhmiä työpaikalla ei ole kovinkaan paljoa. Sidosryhmät voidaan jakaa karkeasti ulkoisiin ja sisäisiin sidosryhmiin.

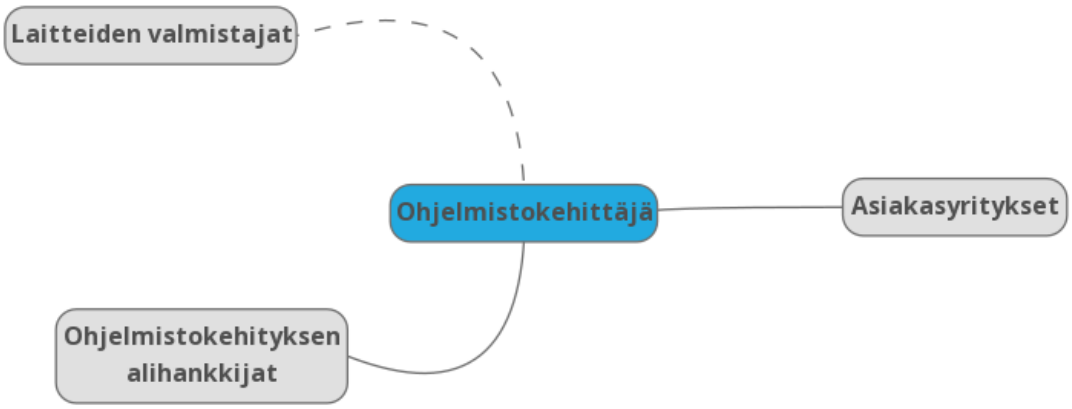
Ulkoisiin sidosryhmiin luetaan yrityksen ulkopuoliset tahot, jotka liittyvät yrityksen toimintaan ja joiden kanssa olen työni puolesta tekemisissä. Sisäisiin sidosryhmiin taas luetaan yrityksen sisäiset tahot, jotka ovat merkittäviä ohjelmistokehittäjän työn kannalta.

2.2.1 Ulkoiset sidosryhmät

Ulkoisiin sidosryhmiin (kuvio 1) voidaan lukea yrityksen tarpeisiin erilaisia laitteita valmistavat yritykset, joiden kanssa en työssäni ohjelmistokehittäjänä kuitenkaan ole juurikaan

tekemisissä. Oman työni kannalta keskeisempiä ulkoisia sidosryhmiä ovat tietysti asiakasyritykset, joilta tulevat toiveet ja palaute sovelluksiin liittyen vaikuttavat suoraan työhöni ja tekemiini ratkaisuihin ohjelmistokehittäjänä.

Muita merkittäviä ulkoisia sidosryhmiä ovat erilaiset sovelluskehityksessä apuna käytettävät alihankkijat. Näitä alihankkijoita hyödynnetään sovelluskehityksessä, sillä yrityksen sisäiset resurssit eivät aina riitä kaikkien tarvittavien ratkaisujen kehittämiseen vaadittavassa ajassa. Olen työssäni suoraan tekemisissä myös alihankkijoiden kanssa, etenkin jos tilatut sovellukset liittyvät jotenkin omiin vastuualueisiini yrityksessä.



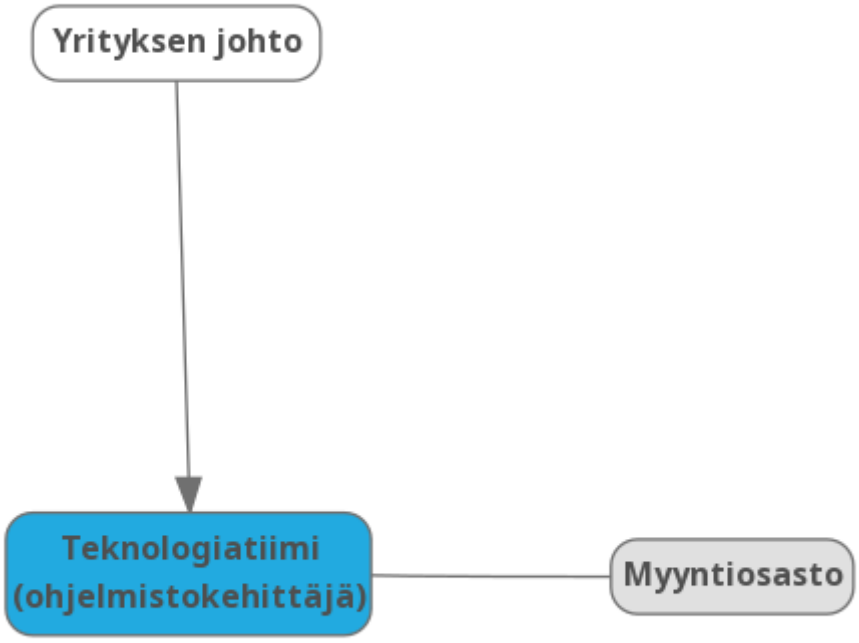
KUVIO 1. Ulkoiset sidosryhmät

2.2.2 Sisäiset sidosryhmät

Ohjelmistokehittäjänä työskentelen osana yrityksen teknologiatiimiä, jota käytännössä johtaa yrityksen teknologiajohtaja. Organisaatiorakenne on matala, ja yritys on henkilömäärältään pieni, joten sisäisiä sidosryhmiä (kuvio 2) ei ole kovinkaan montaa.

Uusien asiakkaiden hankinnasta ja tuotteen myymisestä vastaa yrityksen myyntiosasto, jonka kanssa olen ohjelmistokehittäjänä suoraan tekemisissä. Yleensä potentiaalisten uusien asiakkaiden toiveet ja tarpeet välittyvät myyntitiimin kautta teknologiatiimille, jolloin ohjelmistokehittäjän tehtävänä on arvioida kykyä toteuttaa toivetta sekä arvioida tälle aikataulua.

Toisena sisäisenä sidosryhmänä voidaan pitää yrityksen johtoa, joka on myös suoraan tekemisissä teknologiatiimin ja yksittäisten ohjelmistokehittäjien kanssa matalasta hierarkiasta johtuen. Yrityksen suuret linjat ja hallinnolliset päätökset välitetään ohjelmistokehittäjille yrityksen johdon toimesta.



KUVIO 2. Sisäiset sidosryhmät

2.3 Vuorovaikutustaidot työpaikalla

Työ ohjelmistokehittäjänä vaatii hyviä vuorovaikutustaitoja. Vaikka varsinainen koodin kirjoittaminen on itsenäistä, on sovellusten ja ratkaisujen suunnittelu ja suurempien kokonaisuuksien hahmottelu aina ryhmätyötä. Yksilöiden hyvät vuorovaikutustaidot ovat edellytys tehokkaasti toimivalle ryhmälle ja tästä syystä olennainen osa ohjelmistokehittäjän ammattitaitoa.

Vuorovaikutusta tiimin sisällä tapahtuu viikoittaisissa palavereissa, joissa käydään läpi kulunutta viikkoa, sekä suunnitellaan tulevaa. Kommunikaatio on tärkeässä roolissa myös päivittäisessä työssä niin kasvotusten, kuin pikaviestintäsovelluksienkin välityksellä, kun vastaan tulleita tilanteita ja ongelmia pohditaan yhdessä koko kokonaisuuden kannalta

parhaiden ratkaisujen löytämiseksi. Myös toisten, mahdollisesti kokeneempien, ohjelmistokehittäjien asiantuntemuksen hyödyntäminen, apua ja neuvoa kysymällä, on tärkeä osa tiimin sisäistä vuorovaikutusta.

Hyviä vuorovaikutustaitoja vaaditaan myös laajemmin yrityksen sisällä. Yrityksen pienestä koosta johtuen tulee ohjelmistokehittäjän työssä eteen vuorovaikutustilanteita niin yrityksen johdon, kuin asiakkaidenkin kanssa. Näissä tilanteissa hyvä vuorovaikutustaidot korostuvat entisestään, sillä etenkin asiakkaiden kanssa suoraan tekemisissä ollessa on ohjelmistokehittäjän vastuulla osittain myös koko yrityksestä saatu kuva. Näitä vuorovaikutustilanteita käydään niin sähköpostin välityksellä kuin kasvotustenkin.

Hyvät vuorovaikutustaidot hallitsemalla voi asiantuntijana tuoda esiin omia näkemyksiään ja osaamistaan osana teknologiatiimiä sekä antaa hyvän kuvan yrityksestä ulospäin erityisesti asiakkaiden suuntaan. Oli viestintäkanava sitten pikaviestintäpalvelu, sähköposti tai ihan kasvotusten keskustelu, ovat hyvät vuorovaikutustaidot merkittävä osa kaikkea ihmisten välistä kanssakäymistä koskien sidosryhmästä riippumatta.

3 PÄIVÄKIRJA

3.1 Ensimmäinen viikko

Maanantai

Seurantajakson ensimmäinen maanantai oli myös uuden sprintin ensimmäinen päivä, joten päivä alkoi sprinttipalaverilla. Tyypillisesti sprinttipalaverissa käydään läpi tulevan sprintin tavoitteet ja kartoitetaan tehtävät, jotka jokaisen tulisi saada tehtyä sprintin loppuun mennessä. Omalta osaltani sprintin pituus on vain kaksi viikkoa, sillä olen sprintin viimeisen viikon lomalla. Projektiin, josta olen vastuussa, ei siis tule mitään isoa muutosta. Tästä syystä sprintti on osaltani aika kevyt ja kaikki tehtäväni ovat jokseenkin pieniä. Iso osa kaksiviikkoisesta sprintistäni kuluu todennäköisesti testatessa tässä sekä jo edellisessä sprintissä tekemiäni uusia ominaisuuksia, jotta web-sovelluksen uusi versio olisi julkaisukunnossa lomaani mennessä. Näin julkaisu onnistuu muulta tiimiltä sujuvasti lomani aikana. Asetinkin päivän tavoitteekseni luoda itselleni suuntaa antavan aikataulun tulevista tehtävistäni sekä toteuttaa muutaman pienemmän virheenkorjaustehtävän.

Bugit, jotka aioin korjata, olivat pieniä, mutta niiden syyn paikantaminen vei jonkin verran aikaa. Pienen testailun jälkeen ongelmien syyt kuitenkin löytyivät, ja ryhdyin korjaamaan niitä. Syyt olivat lähinnä pienestä huolimattomuudesta johtuvia sekä pienestä väärinymmärryksestä minun ja backend-kehittäjien välillä. Sain ongelmat korjattua, ja näin myös päiväni tavoite toteutui.

Tiistai

Seurantajakson toisena päivänä otin tavoitteekseni uuden ominaisuuden lisäämisen vastuullani olevaan Angular-sovellukseen. Aikaisemmin tiedot, jotka käyttäjä pystyy lataamaan talteen, muutettiin CSV-muotoon jo backendissä, ja sovelluksessa oli vain linkki, josta tiedoston voi ladata. Tämä aiheutti kuitenkin ongelman aikaleimojen kanssa, sillä backend ei tiedä käyttäjän aikavyöhykettä, ja ajat olivat siksi aina UTC-ajassa.

Keskustelin asiasta backend-kehittäjän kanssa, ja päädyimme ratkaisuun, jossa backend lähettää tiedot sovellukselle JSON-objektina. Tällöin voin sovelluksen päässä muuttaa ajat käyttäjän aikavyöhykkeen mukaisiksi ja tämän jälkeen luoda JSON-datasta CSV-tiedoston.

Löysin CSV-muunnosta varten sopivan Angular-kirjaston ja ryhdyin toteuttamaan muutosta. Sain päivän tavoitteeni toteutettua ja aikaleimojen kanssa työskentelyn myötä opin myös uusia asioita JavaScriptin päivämääriä käsittelevistä funktioista, ja osaan jatkossa käyttää niitä paremmin osana omaa työtäni.

Keskiviikko

Aloitin ensimmäisen seurantaviikon kolmannen päivän testaamalla tiistaina tekemiäni muutoksia vielä hieman. Todettuani kaiken toimivan kuten pitääkin siirryin seuraavaan tehtävääni.

Alihankkijalta tilatusta React-projektista oli aamulla toimitettu uusi versio, jossa edellisestä versiosta löytämämme lukuisat virheet olisi pitänyt olla korjattu. Tarkoituksena on, että alihankkijan saatua projekti alustavasti julkaisukelpoiseksi, siirtyy se minun vastuulleni jatkokehityksen osalta. Päätin siis käydä läpi projektin koodia perusteellisesti, jotta olisin mahdollisimman hyvin perillä sovelluksen toiminnasta ja rakenteesta sen siirtyessä vastuulleni. Samalla tietysti testasin sovelluksen toimintaa ja raportoin alihankkijan kanssa yhteydessä olevalle kollegalleni löytämistäni virheistä.

Tutkiessani sovelluksen koodia, rakennetta ja toimintaa löysin useita virheitä, joista raportoitiin alihankkijalle. Tein itselleni muistiinpanoja ja kommentoin koodin osia, joita haluaisin ehkä uudelleen kirjoittaa projektin siirtyessä minun vastuulleni. Sovelluksesta löytyi vielä niin paljon virheitä, ettei se kelvannut sellaisenaan julkaistavaksi. Totesin sovelluksen tarvitsevan myös melkoisen määrän uudelleenkirjoittamista, jotta kaikki ongelmat saadaan korjattua. Toistaiseksi suurimmat virheet kuitenkin raportoitiin alihankkijalle ja myöhemmin päätetään jatkaako alihankkija vielä projektin kanssa, vai siirtyykö se minulle korjattavaksi tai uudelleenkirjoitettavaksi.

Torstai

Aamulla huomasin kollegani raportoineen löytäneensä illalla virheen vastuullani olevasta Angular-sovelluksesta ja päätinkin aloittaa päiväni korjaamalla raportoidun bugin. Testasin raportissa mainittua kohtaa eri käyttäjillä ja arvoilla, mutta en millään saanut bugia toistettua. Päättelin raportista, että vika saattaisi sittenkin olla backendistä johtuva ja pyysin yhtä backend-kehittäjistä testaamaan sitä. Emme kuitenkaan kumpikaan saaneet toistettua bugia, joten paljoa ei ollut tehtävissä.

Tämän jälkeen päätin jatkaa eilen aloittamaan alihankkijalta saapuneen React-projektin koodin ja rakenteen läpikäymistä ja testaamista. Tein jälleen muistiinpanoja itselleni ja kommentoin koodia kohdista, jotka mielestäni vaikuttivat epäilyttäviltä, tai joiden huomasin olevan jotenkin viallisia. Raportoin jälleen löytämistäni ongelmista yrityksemme päässä projektista vastuusa olevalle kollegalleni.

En ole ollut Reactin kanssa kovin paljoa tekemisissä ja työkaluna esimerkiksi Angular onkin minulle paljon tutumpi. Hallitsen kuitenkin Reactin periaatteet ja koen sen parissa työskentelemisen mielenkiintoiseksi.

Perjantai

Aloitin perjantain jälleen React-projektin testaamisella ja koodin läpikäymisellä. Tässä vaiheessa minulle oli jo selvää, että projektin siirtyessä vastuulleni, tulee se vaatimaan melkoisen määrän uudelleenkirjoittamista. Sovellus ei ollut vielä lähelläkään julkaisukelpoista, joten päätin yrittää pyytää React-projektin ainoaksi tehtäväkseni seuraavaan sprinttiin, mikäli muilla vastuualueillani ei ilmene mitään kiirellisempää.

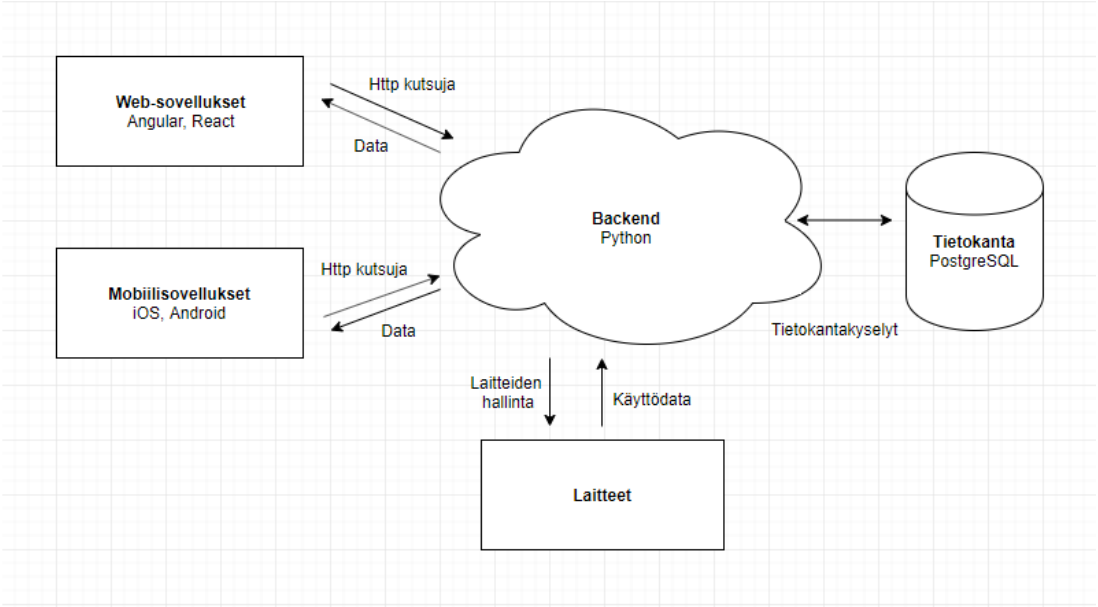
Iltapäivä oli varattu tiimimme kokeneimman kehittäjän pitämälle palvelinoppitunille. Kävimme kattavasti läpi Microsoft Azurea, jotta jokaisella olisi edes jonkinlainen käsitys asiasta. Yrityksen ja tiimin pienestä koosta johtuen on tärkeää, että kaikilla on edes jonkinlainen käsitys oikeastaan kaikesta mitä teemme, jotta tietyt asiat eivät olisi vain yhden henkilön toteutettavissa.

Viikkoanalyysi

Ensimmäinen seurantaviikko kului tyypillisten työtehtävieni parissa. Angular ja React ovat frameworkeja, joita käytän työssäni selainpohjaisten sovellusten kanssa lähes päivittäin. Viikko alkoi sprinttipalaverilla, joita pidetään aina noin kuukauden mittaisten sprinttien alussa.

Yrityksen teknologiatiimissä noudatetaan jonkinlaista muunnelmaa ketterästä Scrum-ohjelmistokehitysmenetelmästä. Tavallisesti tiimi työskentelee noin neljän viikon sprinteissä, jotka alkavat sprintin suunnittelupalaverilla ja päättyvät katselmukseen sprintin tavoitteiden toteutumisesta. Päivittäisten palaverien sijaan pidetään palaveri yleensä kerran viikossa, jolloin kerrataan viikon tapahtumat, ja tarkennetaan vielä tulevan viikon tehtäviä ja tavoitteita. Scrum-mallin hyötyjä ovat esimerkiksi sprinttien jälkeisten katselmusten tarjoama mahdollisuus jatkuvasti parantaa tiimin suorituksia, sekä vaatimusten priorisointi, joka mahdollistaa tärkeiden ominaisuuksien nopean ja tehokkaan tuottamisen (Vanderjack 2015).

Teknologiatiimi vastaa kaikista yrityksen ohjelmistoihin liittyvistä ratkaisuista ja toteutuksista, laitteiden kehityksen ja tuottamisen ollessa alihankkijoiden vastuulla. Kullakin tiimin jäsenellä on oma erikoisalueensa aina palvelimien ylläpidosta aina mobiilisovelluksien kehitykseen. Kaikilta tiimin jäseniltä kuitenkin edellytetään osaamista koko ohjelmistokokonaisuuden (kuvio 3) eri osa-alueilla.



KUVIO 3. Palvelun toimintaperiaate

Yksinkertaistettuna kokonaisuus koostuu Flaskilla toteutetusta backendistä ja PostgreSQL-tietokannasta, useista web- ja mobiilisovelluksista sekä erilaisista laitteista. Laitteet lähettävät käyttötietoja backendiin, joka tallettaa tietoja tietokantoihin sekä toimittaa http-kutsujen avulla näitä tietoja sovelluksille. Eri laitteisiin ja niiden kokonaisuuksiin liittyviä käyttäjätietoja hallitaan web- ja mobiilisovellusten avulla, jolloin backend tallentaa tiedot tietokantaan ja käyttää niitä esimerkiksi hälytysten ja ilmoitusten luomiseen ja lähettämiseen.

Omassa työssäni frontend-kehittäjänä pyrin mahdollisimman toimiviin ja selkeisiin ratkaisuihin, sillä laitteiden lisäksi juuri tämä osa-alue on se, joka näkyy asiakkaalle ulospäin ja vaikuttaa täten käyttäjän mielikuvaan jopa koko yrityksestä. Tiedostan kuitenkin myös muun kokonaisuuden merkityksen tuotteen toimivuuden kannalta ja pyrin kehittämään osaamistani myös muilla osa-alueilla.

3.2 Toinen viikko

Maanantai

Viikko alkoi jokaviikkoisella maanantaipalaverilla. Maanantaipalaverissa käydään läpi, mitä viime viikolla on tehty, mitä alkavalla viikolla aiotaan tehdä sekä mitä sprintille suunniteltuista tehtävistä on vielä tekemättä ja pystytäänkö kaikki näillä näkymin toteuttamaan aikataulussa.

Palaverin jälkeen tarkistin sprinttini tilanteen Jiran sprinttitaululta ja otin sieltä tavoitteekseni erään pienen uuden ominaisuuden lisäämisen Angular-projektiin. Ominaisuus vaati uuden API:n backendiin, joten keskustelin asiasta kollegani kanssa, joka backend muutoksen tekisi. Pääsimme asiasta yhteisymmärrykseen ja ryhdyin työstämään omaa osuuttani ominaisuudesta Angular sovelluksen puolella.

Kollegani sai tehtyä tarvitun backend muutoksen loppupäivästä ja uusi ominaisuus saatiin valmiiksi. Päiväni tavoite siis toteutui. Tehtävä sujui osaltaan rutiinilla, mutta backend-kehittäjien kanssa yhdessä keskustelu ja ominaisuuksien suunnittelu on aina oman kehitykseni kannalta antoisaa. Koen jatkuvasti oppivani uutta myös sillä osa-alueella.

Tiistai

Alihankkijalta oli viimein saapunut niin sanotusti viimeinen versio React-projektista. Aloitin siis päiväni testaamalla sitä, sekä tutkimalla, oliko viimeisimmät bugit saatu korjattua. Totesimme projektin olevan nyt sen verran toimiva, että alihankkija voi lopettaa sen työstämisen ja vastuu projektin viimeistelystä siirtyisi minulle. Painotin kuitenkin, että sovellus vaatii mielestäni vielä uudelleenkirjoittamista ollakseen varmasti toimiva ja helpommin ylläpidettävä. Koodi myös oli vaikealukuista, joten myös siltä osin olisi sitä vielä parannettava.

Olen huomannut toisen kirjoittamaa koodia läpi käydessäni, että se on välillä todella hankalaa ja vaikeaa ymmärtää. Mikäli koodi ei ole selkeää, on sen lukeminen todella haastavaa, varsinkin jos sitä ei epäselvissä kohdissa ole kommentoitu kunnolla. Tästä oppineena yritänkin jatkossa kiinnittää enemmän huomioita koodini luettavuuteen ja kommentointiin.

Keskiviikko

Aloitin keskiviikon tarkistamalla sprinttitaulusta sprinttini tilanteen ja totesin olevani tehtävieni osalta valmis. Jatkoin siis jälleen sovelluksen testausta ja hiomista. Dokumentoin myös lyhyesti tekemäni uudet muutokset, sekä kirjoitin ohjeita mahdollisten ongelmien paikantamiseen ja korjaamiseen, mikäli niitä vielä ensi viikolla löytyy. Nyt tavoitteeni oli kuitenkin varmistaa, että sovellus toimisi ja julkaisu sujuisi mutkitta.

Kehityspalvelimella kaikki näytti toimivan kuten pitääkin, joten päiväni päätteeksi pyysin kollegaani päivittämään sovelluksen uuden version testipalvelimelle. Testipalvelimen idea on siis käytännössä simuloida tuotantopalvelinta turvallisessa ympäristössä. Kun kaikki todetaan toimivaksi testipalvelimella, voidaan tuote vihdoin julkaista. Seuraavan päivän ohjelma tulisikin siis olemaan kahden eri sovellusversion testaaminen testipalvelimilla.

Torstai

Torstaiaamuna testipalvelin oli saatu valmiiksi ja pystyin aloittamaan julkaistavan sovelluksen testaamisen. Kävin läpi uudet ominaisuudet ja varmistin, että kaikki toimii kuten pitääkin. Automatisoitua testausta ei ole vielä saatu käyttövalmiiksi, joten suurin osa testaamisesta tapahtuu käsin. Uudet ominaisuudet näyttivät toimivan, joten siirryin testaamaan myös muuta kokonaisuutta varmistaakseni, ettei mitään ole hajonnut. Kaikki näytti toimivan muutamaa backendin bugia lukuun ottamatta, jotka kuitenkin osoittautuivat vain pieniksi ongelmiksi versionhallinnan kanssa.

Päivän tavoitteekseni otinkin käydä testipalvelimella web-sovelluksen niin tarkasti läpi kuin pystyn, sillä tiedostan kollegoideni kiireen, ja haluan kaiken omassa projektissani toimivan kunnolla ennen lomalle lähtöäni. Ilman kunnollista testausprosessia tämä on tietysti hieman hankalaa, eikä kaikkia virheitä tietystikään pysty löytämään. Pyrin kuitenkin olemaan mahdollisimman järjestelmällinen ja tarkka, sekä kehittymään myös testaajana koko ajan.

Perjantai

Poissa töistä.

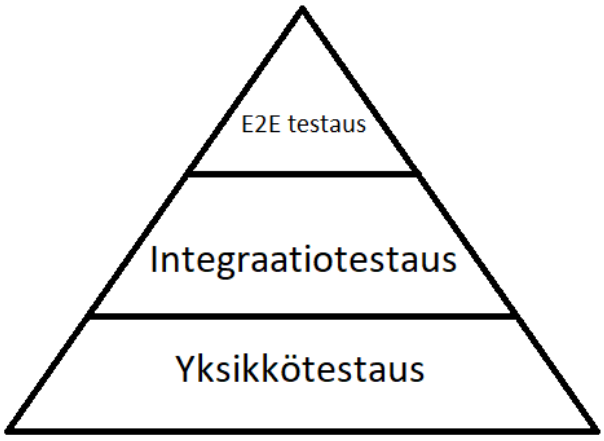
Viikkoanalyysi

Suuri osa toisesta seurantaviikosta kului tekemiäni muutoksia testatessa. Olen huomannut, että hieman kokemattomampana kehittäjänä stressaan välillä tekemäni työni laadusta. Tästä syystä koen, että testaus ja koodin vertaisarviointi on tärkeää, sillä silloin yksittäinen kehittäjä ei ole yksin vastuussa koodin toimivuuden varmistamisesta.

Mahdollisimman kattava testaus vaatii muun muassa niin matalan tason yksikkötestausta, kuin korkeamman tason End-to-End-testaustakin. End-to-End-testaus on tärkeä osa ohjelmistotuotteen laadun varmistamista ja kaiken huomioivien käyttötapausten luominen yrityksen sisäisesti voi olla haastavaa aika- ja resurssipulan vuoksi. Siksi testauksen tarkka suunnittelu niin, että rajattua määriä käyttötapauksia yhdistelemällä pystyttäisiin luomaan mahdollisimman kattavasti testausta, on tärkeää. (Liang, Malatpure, Shafiei, Vago & Zheng 2012.)

End-to-End-testaus on kuitenkin niin korkean tason testausta, että siinä löytyneiden virheiden todellista syytä saattaa olla hankala paikantaa. Tästä syystä matalan tason yksikkötestaus on tärkeää, ja helpottaa ongelmien syyn löytämistä ja virheiden paikantamista jo kehityksen aikaisemmassa vaiheessa. Tätä kuvaa myös niin sanottu testauspyramidi (kuvio 4). Yksikkötestauksella tarkoitetaan testausta, jossa testataan ohjelman pienimpiä yksiköitä. Käytännössä tämä tarkoittaa yksittäisiä metodeja tai funktioita. Yksikkötestauksen

tarkoitus on varmistaa, että jokainen yksikkö toimii kuten pitää (Software Testin Fundamentals 2019).

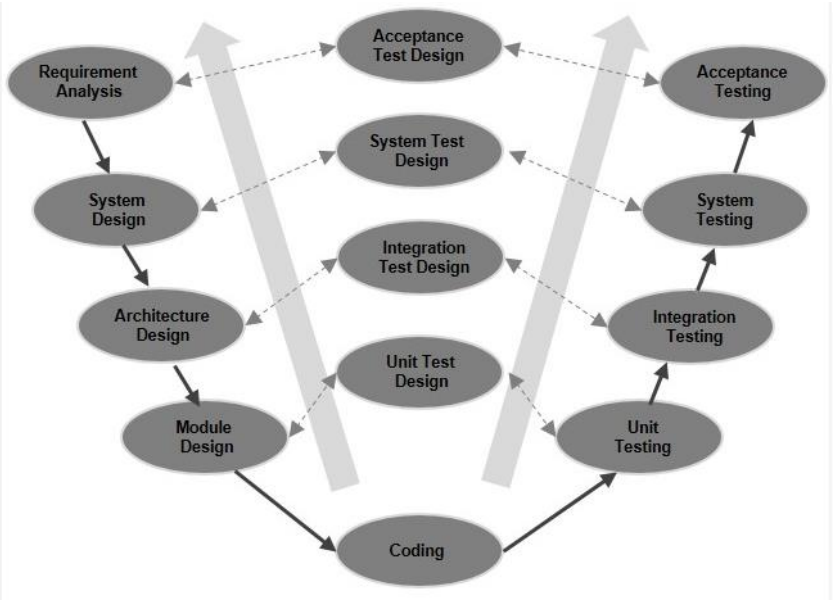


KUVIO 4. Testauspyramidi

Koska yksikkötestauksessa keskitytään testaamaan yksikköjen toimintaa riippumatta muista sovelluksen osista, täytyy siinä käyttää apuna niin sanottuja mockup objekteja. Dummyn tarkoitus on simuloida jonkin testattavan yksikön kanssa tekemisissä olevan ominaisuuden toimintaa, jotta yksikön toimintaa voidaan testata täysin riippumatta muusta sovelluksesta. Integraatiotestauksen tarkoitus taas on testata eri komponenttien toimintaa keskenään (kuvio 5).

KUVIO 5. Integraatiotestaus (Jyväskylän yliopisto 2019)

Osa testauksen suunnittelua on v-mallin käyttö. V-malli on kehitysprosessimalli, jossa toisiaan vastaavat kehitys- ja testausvaiheet suunnitellaan rinnakkain (kuvio 6). Tällöin esimerkiksi korkeimman tason toimintaa suunniteltaessa suunnitellaan myös esimerkiksi End-to-End testausta ja matalinta tasoa suunniteltaessa yksikkötestejä. Näin pystytään määrittelemään, mitä oikeasti halutaan testata ja mikä määrittelee onnistuneen testin. (Tutorialspoint 2019) Työssäni ei kuitenkaan noudateta kirjaimellisesti v-mallia, vaan testausta pyritään suorittamaan ketterien menetelmien mukaisesti mahdollisimman paljon jo kehitysvaiheessa. Todellisuudessa suuri osa testauksesta päädytään kuitenkin suorittaman vasta kehityksen loppuvaiheessa, jolloin prosessi mukailee v-mallia. Myös testauksen jonkin asteinen suunnittelu rinnakkain kehityksen eri tasojen kanssa mukailee v-mallia.



KUVIO 6. V-malli (Tutorials Point 2019)

Kattava testaus on välttämätöntä asiakkaalle näkyvien virheiden välttämiseksi, joilla voi pahimmillaan olla vakavia seurauksia esimerkiksi yrityksen maineen ja pahimmillaan jopa käyttäjän terveyden kannalta (Homes 2013). Ymmärrän, ettei pelkästään itse tekemäni testaus ole riittävää, sillä kehittäjä on usein sokea omia virheitään kohtaan. Tästä syystä on tärkeää, että jokin ulkopuolinen taho, joka ei ole ollut tekemisissä testattavan koodin kanssa suorittaa varsinaisen testauksen. Itse kehittäjä näet tietää, miten ominaisuuden pitäisi toimia ja pyrkii todistamaan sen toimivaksi, eikä täten huomioi kaikkia tapauksia. Ulkopuolisen testaajan tavoite taas on pystyä osoittamaan, että jokin ei toimi ja yrittää täten osoittaa kaikki mahdolliset viat, joita kehittäjä itse ei ole huomioinut. (Homes 2013) Siksi,

varsinaisen testaustiimin puuttuessa, tiimissä pyritään siihen, että kaikkia uusia ominaisuuksia testaa myös kehittäjä, joka ei ole ollut tekemisissä ominaisuuden toteutuksen kanssa. Myös automatisoituun testaukseen pyritään panostamaan.

3.3 Kolmas viikko

Maanantai

Kolmannen seurantaviikon maanantaina alkoi jälleen uusi sprintti, joten päivä alkoi pitkällä sprinttipalaverilla. Palaverissa käytiin jälleen läpi kaikkien tavoitteet seuraavien kolmen viikon ajalle. Omalta osaltani olin varautunut lähinnä React-projektin korjausten aloittamiseen, mutta ilmeni, ettei se tulekaan olemaan sprinttini ohjelmassa. Sen sijaan tavoitteenani oli tehdä muita asiakkaiden toiveissa ilmenneitä uusia ominaisuuksia Angular projektiini. Backendin kohdalla tilanne alkaa olla se, että tehtävää on paljon enemmän, kuin tekijöitä, joten tehtävälistalleni tuli myös muutamia uusia ominaisuuksia backendiin liittyen.

Backend-kehittämiseen tutustuminen ja sen oppiminen ja parempi hallitseminen on pitkään ollut toiveissani. Koen, että työurani kannalta olisi arvokasta hallita myös se puoli ohjelmistokehityksestä.

Loppu iltapäivä vierähti toisessa palaverissa, jossa suunnittelimme, miten toteutamme erään asiakkaan toiveista. Asiakas näet halusi, että sovelluksen käyttäjien olisi tarvittaessa mahdollista saada hälytyksiä myös muista käyttäjäorganisaatioista kuin omastaan. Tällä hetkellä sovellus ei tue tätä, ja kyseessä on verrattain iso muutos, joten se vaati huolellista suunnittelua.

Tiistai

Tiistain aloitin tutustumalla hieman backendiimme, sekä rupesin kokeilemaan pieniä muutoksia. Tavoitteenani tälle päivälle oli tehdä ominaisuus, jonka avulla sovelluksien kirjautumisen istuntopituutta pystyisi muuttamaan. Tähän asti se on ollut aina vakio, eikä sovellusta käyttävä taho ole itse pystynyt sitä muuttamaan.

Tehtävä vaati pienen tietokantamuutoksen, sekä kaksi uutta API-pistettä, joita voidaan käyttää istuntopituuden arvon muuttamiseen sovelluksen puolelta. Pienen opettelun jälkeen sain tehtäväni tehtyä ja päivän tavoitteeni täytettyä. Kokeneemmat kollegani tarjoutuivat onneksi auttamaan ongelmia kohdatessani.

Koin backend muutoksen tekemisen todella mielenkiintoiseksi, vaikka muutos sinänsä oli yksinkertainen. Huomaan nauttivani uuden oppimisesta ja uudet haasteet lisäävät myös

motivaatiota työtäni kohtaan. Olen myös huomannut, että jonkin aikaa alalla työskenneltyäni uusien asioiden oppiminen ja omaksuminen on helpottunut, kun tietyt perusteet ja periaatteet ovat paremmin hallussa.

Keskiviikko

Aloitin päivän testaamalla vielä tiistaina tekemiäni API-kutsuja Postmanilla. Kaikki näytti toimivan kuten pitää, ja päätin ottaa tavoitteeksi ominaisuuden toteuttamisen myös web-sovelluksen puolella.

Sovellukseen tarvittiin siis uusi asetus, jolla istuntopituutta pystyy muuttamaan. Eilen tekemäni API:t näyttivät toimivan hyvin, joten sain uuden ominaisuuden lisättyä sovellukseen rutiininomaisesti.

Tämä oli ensimmäinen ominaisuus, jonka tein kokonaan itse, myös backendin puolelta. Vaikka ominaisuus loppujen lopuksi oli pieni ja yksinkertainen, koin sen valmiiksi saamisen palkitsevaksi. Odotankin jo innolla pääseväni tekemään yhä enemmän ja isompia tehtäviä myös backendiin, sillä haluan pystyä olemaan yhä enemmän hyödyksi koko tiimille, sekä kehittyä itse ohjelmistokehittäjänä.

Torstai

Torstaina otin tavoitteekseni suunnitella backend-kehittäjän kanssa uutta isoa ominaisuutta, josta oli sovittu maanantain palaverissa. Tämä vaatisi suuria lisäyksiä niin web-sovelluksen, kuin backendinkin puolella.

Suunnittelimme ja keskustelimme ominaisuuden tarvitsemista uusista tietokantarakenteista, sekä muista backendiin liittyvistä ominaisuuksista, joita tulen tarvitsemaan uutta ominaisuutta varten.

Suunnitelma saatiin sovittua ja backend-kehittäjän ryhtyessä tekemään sovittuja tehtäviään, rupesin luomaan Angular sovellukseen uusia komponentteja uutta ominaisuutta varten. Päivän päätteeksi sain uusien komponenttien rakenteen valmiiksi, ja pystyisin huomenna aloittamaan niiden varsinaisen koodaamisen.

Perjantai

Perjantaina keskustelin ensimmäiseksi uudesta ominaisuudesta vastaavan backend-kehittäjän kanssa ja hän kertoi, että oli saanut backendin siihen tilanteeseen, että voin ruveta testaamaan muutoksia web-sovelluksen päässä. Niinpä pyysin häntä lähettämään minulle tiedot uusista kutsuista ja otin päiväni tavoitteeksi alustavan version kirjoittamisen uusista ominaisuuksista.

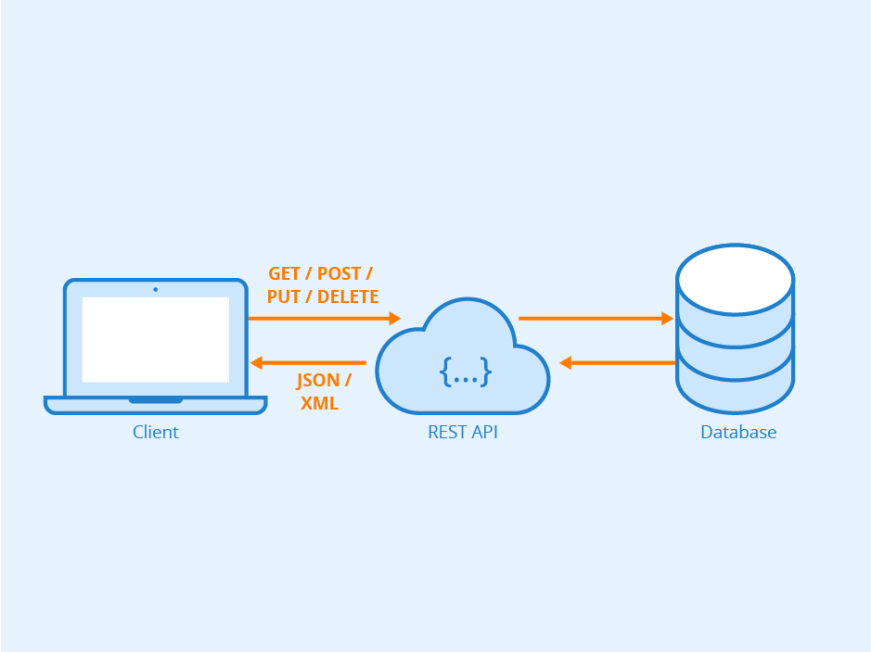
Aloitin kirjoittamalla tarvittavat ominaisuudet uuden tyyppisten tietojen luomista, muokkaamista, sekä listan hakemista varten. Olin saanut kollegaltani tiedot API-kutsuista, joten tämä sujui suoraviivaisesti. Testasin myös kovakoodatuilla arvoilla, että kaikki näyttäisi toimivan, kuten pitää. Perustoiminta tietojen hakemiseen ja muuttamiseen näytti toimivan, kuten pitääkin, joten siirryin tekemään tarvittavaa käyttöliittymää.

Käyttöliittymän ulkoasu ei sinänsä tarvinnut suurempaa suunnittelemista, sillä toteutin sen tietysti saman henkisenä, kun kaikki muutkin sovelluksen jo olemassa olevat ominaisuudet. Päivän päätteeksi sain vielä hieman opastusta backendin kanssa, sillä tulevalla viikolla kaikki backend-kehittäjät tulevat todennäköisesti olemaan poissa toimistolta melko paljon. Tarvittaessa pystyisin siis ehkä ratkaisemaan itsekseni mahdollisia yksinkertaisempia ongelmia.

Viikkoanalyysi

Kolmannella seurantaviikolla pääsin tekemään jonkin verran muutoksia backendiin. Minulle uutena asiana pääsin toteuttamaan uusia API:a. Nämä API:t toteutettiin REST arkkitehtuurimallin mukaisesti.

REST (Representational State Transfer) on arkkitehtuurimalli HTTP-protokollaan perustuvien ohjelmistorajapintojen toteuttamista varten (kuvio 7). Kuten muillakin arkkitehtuurimalleilla, on myös REST:illä määritellyt ehdot, jotka niin sanotusti RESTfuliksi luettavan rajapinnan tulee täyttää. (RESTfulapi.net 2019.)



KUVIO 7. REST API kaavio (Seobility 2019)

Tärkein vaatimus REST:ssä on asiakasohjelman ja palvelimen pitäminen erillään (RESTfulapi.net 2019). Tämä tarkoittaa, että kumpaakin, palvelinkoodia ja asiakaskoodia, voidaan muuttaa erillään toisistaan ilman, että se vaikuttaa kummankaan toimintaan. Käytännössä tämä tarkoittaa, että niin kauan, kun kaikki rajapintoihin yhteydessä olevat sovellukset tietävät mitä viestejä kuuluu lähettää tai vastaanottaa, voidaan niitä kehittää erillään täysin riippumatta tosistaan.

Toinen tärkeä vaatimus on tilattomuus, mikä tarkoittaa, että niin palvelin, kuin asiakaskin pystyvät vastaanottamaan ja ymmärtämään minkä tahansa viestin toisiltaan riippumatta aikaisemmista viesteistä tai niiden sisällöstä. Tämä tarkoittaa, että lähetettävän kutsun täytyy sisältää kaikki tarvittava tieto halutun toiminnon toteuttamiseksi ilman tarvetta erilaisille tilan tai kontekstin tarkistuksille. (RESTfulapi.net 2019.)

Osaltaan tilattomuuteen pakottaa resurssien (resource) käyttö. Resurssit ovat tunnisteita, jotka kuvaavat kutsun kohdetta, joka puolestaan voi olla esimerkiksi yksittäinen henkilö tai suurempi määritelty resurssien kokonaisuus. Esimerkiksi kuvion 8 ensimmäinen rivi kertoo, että kutsun kohteena on customers kokoelmaan kuuluva yksittäinen asiakas, joka määritellään customerId:n perusteella. Toinen rivi taas kertoo, että kutsu haluaa käsitellä tietyn asiakkaan tilejä ja kolmas kertoo, että halutaan käsitellä tietyn asiakkaan tiettyä tiliä. Näihin resursseihin päästään kutsuilla käsiksi endpointien kautta.

/customers/{customerId}
/customer/{customerId}/accounts
/customer/{customerId}/accounts/{accountId}

KUVIO 8. Resurssien määrittely

Endpointilla tarkoitetaan rajapinnan pisteitä, joiden kautta API kommunikoi muiden järjestelmien kanssa. Yleensä endpoint sisältää palvelimen tai palvelun URL osoitteen, sekä käsiteltävän resurssin tunnisteen. (RESTfulapi.net 2019.) Esimerkiksi kuvion 9 osoite osoittaa ”service” nimiseen palveluun ja resurssiin, joka on asiakas tunnisteella ”123”. Kuluneella viikolla testasin tekemiäni API:a lähettämällä Postman-työkalulla kutsuja niitä vastaaviin endpointeihin. Varmistin, että backend suoritti määritellyn toimenpiteen ja sain paluuarvona oikeanlaisen vastauksen.

https://www.esimerkki.com/service/customer/123

KUVIO 9. Endpoint

Pyrin noudattamaan näitä ohjeita parhaani mukaan uusia ominaisuuksia toteuttaessani. Etenkään nimeämiskäytäntöihin ei ole yrityksessä aikaisemmin juurikaan kiinnitetty huomiota. Tämä on kuitenkin asia, jota on suunniteltu tulevaisuudessa kehitettävän, joten yritän omassa työssäni tehdä hyvien nimeämiskäytäntöjen mukaisia rajapintoja heti alusta alkaen.

3.4 Neljäs viikko

Maanantai

Maanantai alkoi jälleen jokaviikkoisella maanantaipalaverilla. Kerroin saaneeni viime viikolla valmiiksi ensimmäisen backend muutokseni, sekä toteuttaneeni kyseinen ominaisuuden myös frontendin puolella. Kerroin myös päässeeni hyvään vauhtiin uuden ison ominaisuuden implementoinnin kanssa, ja totesin jatkavani tämän tehtävän parissa tälläkin viikolla. Saatuani ominaisuuden valmiiksi, siirtyisin seuraaviin sprintin mukaisiin tehtäviini.

Palaverin jälkeen jatkoinkin siis tämän uuden ominaisuuden parissa. Otin päiväni tavoitteeksi ominaisuuden käyttöliittymän hiomisen. Perustoiminta oli viime viikon jäljiltä jokseenkin kunnossa, joten päätin lisätä esimerkiksi varmistusdialogeja erinäisiin toimintoihin. Lisäsin myös vaatimusten mukaisesti erilaisia varoituksia eri toimintoihin käyttöliittymän selkeyttämiseksi ja käyttäjän auttamiseksi.

Tiistai

Tiistaina tarkoituksenani oli jatkaa vielä uuden ominaisuuden hiomista. Aamulla kuitenkin selvisi, että eri sovelluksiemme käyttäjämanuaalit oli saatava valmiiksi perjantaihin mennessä. Suurin osa teknologiatiimistä oli tekemässä asennuksia ulkomaisen asiakkaan luona, ja vähäisestä miehityksestä johtuen jouduin auttamaan manuaaleista vastaavaa markkinointihenkilöä manuaalien tekemisessä. Ajattelin tämän samalla toimivan myös kertauksena itselleni sovelluksiemme kokonaisuudesta ja sen toiminnasta.

Käytännössä loin pienen esimerkkiympäristön, jota käytin eri sovelluksilla ja otin kuvakaappauksia eri ominaisuuksista. Kirjoitin kuvien yhteyteen niissä näkyvistä toiminnoista ja lähetin ne markkinointipäällikölle hiottavaksi ja muutettavaksi varsinaisen käyttöoppaan muotoon.

Päivä toimi muistutuksena pienessä yrityksessä työskentelyn haasteista. Välillä voi tosiaan joutua tekemään jotain aivan muuta, kuin oli aamulla töihin tullessa suunnitellut. Kaiken kaikkiaan kuitenkin opin hahmottamaan systeemimme kokonaisuutta paremmin sekä pohtimaan asioita hieman eri näkökulmasta kuin normaalisti sovelluskehittäjänä.

Keskiviikko

Keskiviikkona pääsin jatkamaan taas omia töitäni. Backend-kehittäjä, joka oli vastuussa aiemmin aloittamastamme uudesta ominaisuudesta, oli saanut sen backendin osalta valmiiksi, joten otin päiväni tavoitteeksi sen saamisen valmiiksi myös sovelluksen puolella.

Suunnittelemamme API:t olivat muuttuneet hieman suunnitellusta, joten tein pieniä muutoksia koodiini, jotta sain kaiken toimimaan sujuvasti. Löysin yhdestä API:sta vielä pienen bugin, jonka kollegani korjasi nopeasti. Päivän loppuun mennessä ominaisuus näytti toimivan hyvin ja olin saanut päivän tavoitteeni täytettyä. Tehtäväksi jäi kuitenkin vielä ominaisuuden toiminnan hiominen tehokkaammaksi ja suoraviivaisemmaksi.

Torstai

Poissa töistä.

Perjantai

Aloitin perjantain hiomalla vielä uuden ominaisuuden ulkoasua, sekä parantelemalla hieman sen toimintaa. Vähensin turhia HTTP-kutsuja tallentamalla osia tiedoista paikallisiin objekteihin. Näin sain ominaisuuteen liittyvän listan tiedot päivittymään dynaamisesti hakematta tietoja joka muutoksen jälkeen uudestaan backendistä.

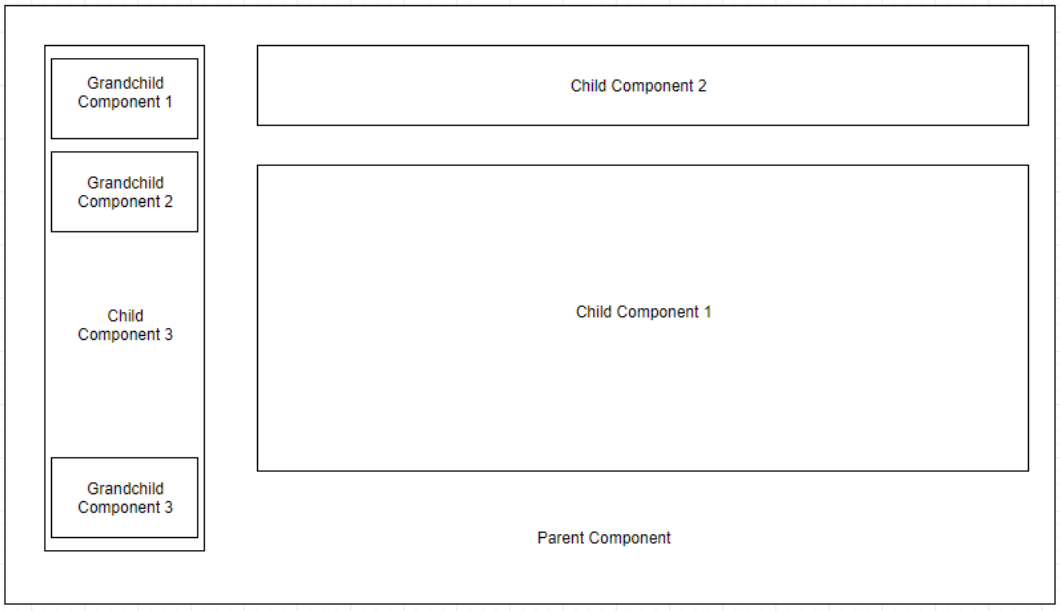
Saatuani uuden ominaisuuden valmiiksi testausta varten, siirryin seuraavaan tehtävääni. Angular sovellukseen tarvittiin uusi ominaisuus, joka vaatisi minulta jälleen myös muutoksia backendiin. Jouduin tekemään pieniä muutoksia tietokantakyselyyn, mutta tämä onnistui helposti matkimalla hieman erästä toista tietokantakyselyä, joka haki akkutietoja eri tarkoitusta varten. Testasin Postmanilla, että API toimii, joten maanantaina pääsisin tekemään ominaisuuden valmiiksi frontendiin.

Viikkoanalyysi

Suurin osa viikosta kului uuden ominaisuuden implementoinnissa Angular sovellukseen. Oman haasteensa viikkoon toi keskellä viikkoa tullut yllättävä vastuu käyttäjämanuaaleihin liittyen, mutta muuten viikko sujui ilman suurempia ongelmia. Koin tämän kuitenkin hieman häiritsevänä, sillä olin päässyt hyvään vauhtiin uuden ominaisuuden kanssa ja tällainen keskeytys keskellä viikkoa häiritsi keskittymistäni alkuperäiseen tehtävääni.

Työskentelen päivittäisessä työssäni runsaasti Angularin kanssa, joka on TypeScriptillä toteutettu framework selainpohjaisten sovellusten tekemiseen. Perusrakenne Angular-sovelluksessa koostuu komponenteista (component). Komponentit ovat ohjelman osia, niin sanottuja sivuja tai paloja, jotka hallitsevat, mitä käyttäjä näkee ja pystyy sovelluksessa tekemään. Komponentti koostuu TypeScriptillä koodatusta komponenttiluokasta, sekä HTML:ää ja Angularin omaa syntaksia sisältävästä templatesta. (Angular 2019.)

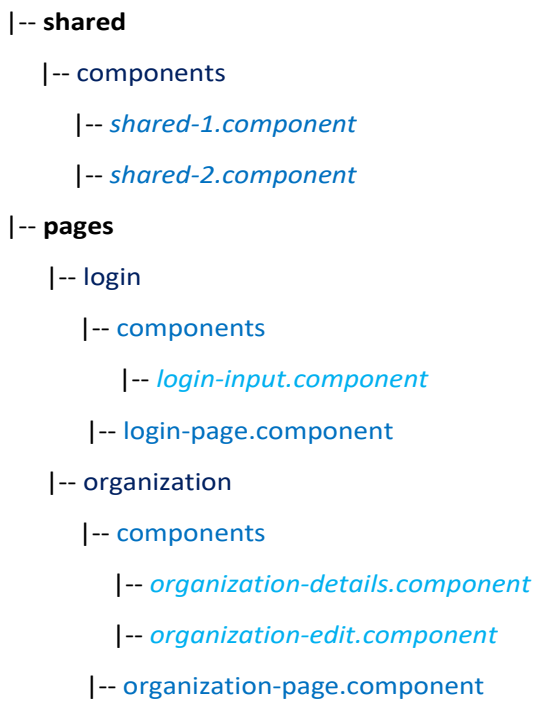
Jokainen Angular sovellus sisältää vähintään yhden komponentin. Tämä komponentti toimii pohjana, jonka päälle sovellus rakennetaan. Komponentti voi sisältää useita lapsikomponentteja, jotka puolestaan voivat sisältää omia lapsikomponenttejaan (kuvio 10). Näin muodostuu hierarkkinen komponenttirakenne, jossa dataa voidaan välittää ylemmän tason komponenteilta alemman tason komponenteille. Tämä, yhdessä Angularin data binding ominaisuuden kanssa mahdollistaa yleiskäyttöisten komponenttien tekemisen, joita voidaan käyttää ympäri sovellusta, eikä saman tyylisiä osia tarvitse koodata jokaiseen näkymään erikseen.



KUVIO 10. Komponenteista muodostuva käyttöliittymä

Osa hyvää komponenttirakennetta on lähdekoodin selkeä puurakenne, joka mahdollistaa eri komponenttien helpon käsittelyn ja paikantamisen (Angular 2019). Angularin tyyliohje ei määrittele hakemistorakennetta tämän enempää, joten on kehittäjien ja tiimien vastuulla järjestellä komponentit järkevästi. Itse olen työssäni usein päätynyt käyttämään kuvion 11 mukaista mallia, jossa ympäri sovellusta käytettävät yhteiset komponentit ovat omassa

hakemistossaan. Tämän lisäksi sovellus on jaettu osiin varsinaisten selattavien sivujen mukaan ja jokaisen sivun omat komponentit ovat sivun hakemistossa.



KUVIO 11. Komponenttien puumalli

Olennainen ominaisuus, joka mahdollistaa komponentin TypeScript koodin ja sen HTML-templaten välisen kommunikaation, on data binding. Data bindingin avulla ominaisuuksia voidaan välittää dynaamisesti suoraan komponenttiluokasta templateen, sekä tapahtumia templatesta komponenttiluokkaan (kuvio 12). Yksinkertaisimmillaan data binding tarkoittaa, että ominaisuus välitetään komponenttiluokasta templateen vain kirjoittamalla ominaisuuden nimi HTML-koodin joukkoon Angularin syntaksin mukaisesti kaksien aaltosulkujen sisään. Data binding on Angularissa dynaamista, mikä tarkoittaa, että ominaisuuden arvon muuttuessa, näkyy arvon muutos myös templatessa ilman, että sitä täytyy erikseen hakea uudestaan. Tämä mahdollistaa dynaamisten ja rikkaiden näkymien suoraviivaisen toteuttamisen.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1>{{title}}</h1>
    <h2>{{greetings}}</h2>
    `
})
export class AppComponent {
  title = 'Esimerkki otsikko';
  greetings = 'Hello world!';
}
```

KUVIO 12. Komponentti ja template, sekä data binding

Angular on teknologia, jota olen työssäni käyttänyt eniten ja jonka koen hallitsevani hyvin. Komponenteista koostuvasta rakenteesta johtuen Angular-sovellukset skaalautuvat hyvin ja niiden koodi on helppo pitää selkeänä, vaikka sovelluksen koko kasvaisi. Hyvin suunnitellut ja toteutetut komponentit ovat uudelleen käytettäviä, mikä poistaa tarpeen koodin toistolle. Yksittäisistä komponenteista hierarkkisesti rakennettu sovellus on myös helposti ylläpidettävä, sillä yksittäisiä komponentteja voi muokata ilman, että se suoranaisesti vaikuttaa muiden komponenttien toimintaan.

Ymmärrän, että tässäkin teknologissa on vielä paljon ominaisuuksia, joihin en vielä ole perehtynyt, tai joita en ole vielä tullut käyttäneeksi. Tämän lisäksi Angularia päivitetään kovaa vauhtia, joka myös osaltaan vaatii aiheen jatkuvaa opiskelua ja seuraamista. Pyrinkin työni ohessa seuraamaan aiheeseen liittyviä julkaisua ja blogikirjoituksia pysyäkseni ajan tasalla sen kehityksestä.

3.5 Viides viikko

Maanantai

Maanantai alkoi jälleen joka viikon avaavalla maanantaipalaverilla. Palaveri kesti tällä kertaa pitkään, sillä koko teknologiatiimi oli pitkästä aikaa paikalla, ja ulkomailla asennuksia tehdessä oli noussut esiin asioita, jotka tuli käydä läpi. Omalta osaltani kerroin suurimman osan viimeviikosta kuluneen käyttäjämanuaalien parissa, mutta sprinttini olevan siitä huolimatta hyvässä vaiheessa. Tulisin tänään saamaan laitteiden akkutason ja tilan esittämisen valmiiksi Angular sovelluksessa. Tiistaina ehtisin vielä käydä läpi tehtäviäni, ja keskiviikkona aloittaisimme jälleen testaamisen julkaisua varten.

Ryhdyin työstämään akkutason ja laitteen tilan näyttämistä ja asetin päiväni tavoitteeksi tämän ominaisuuden valmiiksi saamisen. Uutta rajapintaa tätä ominaisuutta varten ei tarvittu, sillä lisäsin uudet tiedot jo käytössä olevan rajapinnan toteutukseen, jolloin lisääminen oli suoraviivaista. Lisäsin vaatimusten mukaisesti laitteiden listaan näkymän, joka näyttää laitteen akkutason prosentteina, sekä tätä vastaavan ikonin. Lisäsin samaan kohtaan myös joko BLE tai GSM ikonin (kuvio 13), riippuen laitteen tilasta. Sain päiväni tavoitteen toteutettua rutiininomaisesti. Uusien pienten ominaisuuksien lisääminen Angular sovellukseen alkaa olla jo tuttua ja koen Angular osaamiseni olevan muutenkin jo hyvällä tasolla.

KUVIO 13. BLE ja GSM ikonit

Tiistai

Aloitin päivän tarkistamalla sprinttini tehtävät ja totesin, että olin saanut kaiken testausta vaille valmiiksi. Testausta päätettiin kuitenkin lykätä seuraavalle viikolle toisten kehittäjien kiireiden vuoksi, joten päätin ryhtyä työstämään pienempiä virhekorjaustehtäviä, jotka olivat jääneet roikkumaan kaiken muun kiireen keskellä. Lisäksi olisin tietysti saatavilla muiden avuksi, mikäli tarve vaatisi.

Iso osa ticketeistä oli lähinnä kosmeettisia virheitä, mutta myös käyttäjän kannalta todella ärsyttäviä. Esimerkiksi laitelista ei päivittynyt automaattisesti, kun uusi laite lisättiin, eikä uusi laite täten näkynyt siinä heti. Sainkin hyvän määrän pieniä bugeja korjattua.

Tällaiset pienemmät, käyttäjän ja toiminnan kannalta vähemmän merkittävät bugit jäävät usein roikkumaan pitkiksikin ajoiksi, tärkeämpien tehtävien mennessä näiden edelle. Siksi koin hyväksi, että olin saanut sprinttini ajoissa valmiiksi ja pystyin keskittymään myös niihin. Mikäli näitä pieniä bugeja kertyy paljon, ilman, että niitä ehditään milloinkaan korjaamaan, saattavat ne myöhemmin aiheuttaa odottamattomia uusia virheitä, kun kaikki ei toimikaan odotetulla tavalla.

Keskiviikko

Olin varautunut aloittamaan testaamisen keskiviikkona, mutta koska se päätettiin siirtää seuraavalle viikolle jatkoin jo eilen aloittamani pikkuvikojen korjaamista. Päiväni tavoitteeksi asettuikin siis pienempien virheiden korjaaminen. Edelleen olisin myös apuna muille tarvittaessa.

Loppupäivästä kävi ilmi, että eräs mobiilisovellusten uusi ominaisuus tarvitsisi toimiakseen järkevästi vielä uuden asetuksen vastuullani olevaan Angular-sovellukseen. Keskustelimme mobiili- ja backend-kehittäjien kanssa vaatimuksista ja päätimme ryhtyä toteuttamaan tätä seuraavana päivänä.

Sain päivän aikana hyvän määrän pikkuvikoja korjattua ja koin päiväni tavoitteen toteutuneeksi. Koen kokeneempien kollegojeni kanssa suunnittelun aina opettavaiseksi kokemukseksi, ja tunsin tänäänkin kehittyneeni alan ammattilaisena.

Torstai

Päivä alkoi pienellä palaverilla mobiili- ja backend-kehittäjien kanssa, jossa käsittelimme vielä eilen keskusteltua muutosta. Suurimmat muutokset koskivat tietysti backendiä ja mobiilisoveluksia ja lopulta Angular-sovelluksen asetuksiin tarvittiin vain yksi uusi asetus tämän uuden ominaisuuden intervalliajan asettamista varten.

API:t tätä ominaisuutta varten eivät olleet kovin monimutkaisia, joten ryhdyin tekemään niitä kokeneemman backend-kehittäjän keskittyessä ominaisuuksiin, joita tarvittiin mobiilisovelluksia varten. Saatuani API:t kirjoitettua testasin Postmanilla, että ne toimivat tarkoitetulla tavalla. Tehtyäni tämän siirryin toteuttamaan asetusta sovellukseen. Kaikki näytti toimivan kuten pitää, joten ilmoitin asetuksen olevan valmis testausta varten, kunhan mobiilikehittäjät saavat ominaisuutensa valmiiksi.

Perjantai

Perjantai alkoi taas pienellä yllätyksellä. Sain tiedon, että viime viikolla tehdyt käyttäjämanuaalit tulisi saada nettiin asiakkaille ladattaviksi ja tätä varten tarvittaisiin tietysti jonkinlainen verkkosivu. Web-kehittäjänä luonnollinen valinta tehtävään olin minä. Päiväni tavoitteeksi asettuikin siis tämän sivun tekeminen.

Manuaaleista vastaava markkinointihenkilö oli tehnyt esimerkkikuvan siitä, miltä manuaalien lataussivun tulisi näyttää. Sain häneltä tarvittavat kuvat sivua varten ja asian kiireellisyydestä johtuen päätin, että nopein tapa toteuttaa se on yksinkertainen HTML ja CSS sivu ilman mitään suurempia temppuja. Sivun ainoa tarkoitus tulisi näet olemaan, että asiakkaat voivat ladata manuaalit sieltä itselleen ikoneja klikkaamalla.

Päiväni tehtävä oli yksinkertainen ja suoraviivainen, mutta toimi tietynlaisena kertauksena perusteista. Samalla se myös muistutti omasta kehityksestäni, sillä vaikka tehtävä oli yksinkertainen, olisi se todennäköisesti vaatinut hieman enemmän pohtimista ja aikaa vielä vaikkapa vuosi sitten. Nyt tiesin kuitenkin heti, miten aion tehtävän toteuttaa ja teinkin sen sujuvasti.

Viikkoanalyysi

Kuluneen viikon aikana toteutin muun muassa verkkosivun käyttäjämanuaalien lataamista varten. Lataussivun ulkoasu oli tarkkaan määritelty (kuvio 14), eikä siltä vaadittu muuta toiminnallisuutta, kuin linkit, joita klikkaamalla käyttäjä voi ladata pdf-muotoiset laitteiden käyttömanuaalit itselleen luettavaksi. Tästä syystä päädyin toteuttamaan sivun vain yhdellä HTML-tiedostolla, joka sisälsi tarvittavan CSS-koodin ulkoasuvaatimusten saavuttamiseksi (kuvio 15). Koin tämän nopeimmaksi, mutta toimivaksi tavaksi saavuttaa vaadittu toiminnallisuus.

KUVIO 14. Verkkosivun ulkoasu

Verkkosivu on yksinkertaisimmillaan vain HTML dokumentti, jonka selain lukee ja näyttää käyttäjälle verkkosivuna. HTML on merkintäkieli, jota käytetään erityisesti verkkosivujen koodaamiseen. (W3Schools 2019.) Se koostuu sisäkkäin ja peräkkäin kirjoitetuista kulmasuluin merkityistä tunnisteista (kuvio 15). Nämä tunnisteet toimivat ohjeina selaimelle

verkkosivun elementtien ja ominaisuuksien näyttämiseksi, sillä rakenteellinen merkitsemistapa määrittää elementtien tarkoituksen ja järjestyksen. Sijoittamalla elementtejä esimerkiksi sisäkkäin, jolloin yksi elementti toimii useamman elementin niin sanottuna wrapperina, voidaan vaikuttaa siihen, miten elementit verkkosivulla asettuvat, tai miltä ne näyttävät. Elementeillä on erilaisia käyttötarkoituksia, esimerkiksi *<h3>Otsikko</h3>* tarkoittaa, että sana ”Otsikko” on kolmannen asteen otsikko. Elementtien merkinnät eivät kuitenkaan suoraan määritä sitä, miltä elementin tulisi verkkosivulla näyttää, mutta yleensä selaimet noudattavat saman suuntaisia oletustapoja saman tyylisten elementtien näyttämiseen (W3Schools 2019).

Pelkällä HTML koodilla voidaan siis jokseenkin määritellä verkkosivun rakenne ja eri elementtien tarkoitus. Verkkosivun, ja sen elementtien, tarkempaan muotoiluun käytetään kuitenkin CSS-koodia. CSS on tyyliohjekieli HTML- ja XML-dokumenttien esitystavan määrittelemiseksi (MDN Web Docs 2019). Se on yksi avoimen verkon ydinkielistä ja on standardoitu, jotta eri verkkoselaimet käsittelisivät tyylit yhdenmukaisesti (MDN Web Docs 2019). CSS-koodia voidaan käyttää esimerkiksi sivun asettelun, värien, fonttien ja jopa animaatioiden muokkaamiseen ja luomiseen. CSS-koodin syntaksi koostuu tunnistimesta, joka määrittelee mitä elementtiä ehtojen halutaan koskevan, sekä ominaisuudesta ja ominaisuudelle annettavasta arvosta (kuvio 15).

```
<html>
<head>
<title>Esimerkki sivu</title>
<style>
    .main-container {
        background-color: white;
    }
    h1 {
        color: blue;
    }
    .content {
        background-color: black;
    }
    p {
        color: white;
    }
</style>
</head>
<body>
<div class="main-container">
    <div class="title">
        <h1>Otsikko</h1>
    </div>
    <div class="content">
        <p>Ensimmäinen kappale</p>
        <p>Toinen kappale</p>
    </div>
</div>
</body>
</html>
```

KUVIO 15. Esimerkki HTML-dokumentti, tyylit määritelty CSS:llä

Nämä teknologiat ovat perustyökaluja, joita työssäni käytän myös Angular- ja React-sovellusten kanssa. Kuluneella viikolla vaadittu sivu ei kuitenkaan ollut varsinainen sovellus, kuten esimerkiksi Angularilla toteuttamani ratkaisut, joten manuaalisivun toteuttamiseksi päädyttiin käyttämään pelkästään näitä yksinkertaisimpia teknologioita. Suurempien kokonaisuuksien tai enemmän toimintoja vaativien sivujen toteuttamiseen pelkästään nämä tekniikat eivät kuitenkaan enää riitä.

3.6 Kuudes viikko

Maanantai

Maanantai alkoi tavan mukaan maanantaipalaverilla, jossa käytiin läpi viime viikkoiset saavutukset. Kerroin omalta osaltani saaneeni valmiiksi kaikki tehtäväni tätä sprinttiä varten ja uuden sovellusversion olevan valmiina testattavaksi. Alkavan viikon onkin tarkoitus olla testausviikko julkaisua varten, joten aikatauluni oli pitänyt hyvin. Myös muiden tehtävät olivat enimmäkseen hyvällä mallilla, joten pystyisimme aloittamaan testauksen tänään.

Palaverin jälkeen varmistin, että kehityspalvelimella on viimeisimmät sovellukseen tekemäni muutokset, jotta kaikki voivat käyttää sitä testaamiseen. Sovelluksesta löytyikin heti pieni bugi, jonka korjasin nopeasti. Päivän aikana ilmeni vielä muutamia pieniä ongelmia, jotka sain korjattua. Testasin samalla myös mobiilisovellusta, jonka julkaisulla oli hieman suurempi kiire, sillä toinen mobiilikehittäjistä jäisi lomalle keskiviikkona.

Tiistai

Tiistaina testaaminen jatkui. En kokenut vielä olevani yhtä järjestelmällinen ja tarkka testaaja kuin kokeneemmat kollegani, joten osana päivän tavoitettani yritin opetella järjestelmällisempää työskentelytapaa testatessani.

Päivän aikana web-sovelluksessani ilmeni pari pientä bugia, jotka sain korjattua helposti. Olen huomannut, että suurin syy virheilleni on yleensä ehkä pieni huolimattomuus kiireen keskellä, mutta nämä bugit johtuivat pienestä epäselvyydestä minun ja backendistä vastaavan kollegani välillä. Testasin tekemäni korjaukset ja ne näyttivät toimivan kuten piti.

Testatessani omia muutoksiani, löysin ongelman eräässä mobiilisovelluksen uuden ominaisuuden toiminnassa. Kokeilin samaa vielä muutaman kerran varmistaakseni, että saan bugin toistettua ja näin ollen myös selitettyä, miten se ilmenee. Tutkittuani lisää, päättelin, että ongelma johtuu todennäköisesti jonkinlaisesta virheestä backendissä, ja raportoin bugista muutoksesta vastaavalle kollegalleni. Kävimme vielä yhdessä läpi, miten kyseisen

bugin saa toistettua ja kollegani ilmoittikin heti aavistavansa, mistä se saattaisi johtua. Päivä oli kuitenkin niin pitkällä, että korjaaminen jäisi huomiseksi.

Keskiviikko
Poissa töistä.

Torstai

Torstai alkoi pienellä yhteistyöllä toisen kehittäjän kanssa. Hänen oli pienistä ongelmista johtuen muutettava paria API:a hieman, joka edellytti pieniä toimenpiteitä myös minulta web-sovelluksen puolella. Tarkistin varmuuden vuoksi, ettei web-sovellus käytä näitä API-kutsuja missään muualla kuin ilmeisimmissä paikoissa ja muutin niiden kutsut uusia vaatimuksia vastaavaksi. Testasimme vielä backend kehittäjän kanssa, että ne toimivat kuten ennenkin, eikä mikään ollut hajonnut ja siirsin muutokset kehityspalvelimelle testattavaksi.

Suuri osa tehtävistäni vaatii yhteistyötä eri osa-alueista vastaavien kollegoideni kanssa ja olenkin huomannut, että osaamiseni kehittyessä myös yhteistyö muiden kanssa on helpompaa. Olen koko ajan paremmin perillä järjestelmämme eri alueista ja puolista, joka helpottaa yhteistyötä, sillä ymmärrän ja hahmotan paremmin vaatimukset myös kollegoideni vastuualueiden kannalta. Tämä taas on auttanut minua tekemään parempia ja toimivampia ratkaisuja myös omissa tehtävissäni.

Perjantai

Perjantaina kollegani ilmoitti, että kaikki on valmista testaamisen aloittamista varten testipalvelimella. Niinpä minun piti vielä varmistaa, että kaikki toimii ja kääntää uusin versio sovelluksesta valmiiksi buildiksi. Testatessa oli kuitenkin ilmennyt vielä yksi asia, joka tulisi korjata. Muutama lista sovelluksessa ei ollut aakkosjärjestyksessä, joka osaltaan hankaloitti käyttäjäkokemusta jonkin verran. Niinpä päiväni tavoitteeksi muodostui järjestää listat aakkosjärjestykseen ja saada oma osuuteni testattavaksi staging palvelimelle.

Tehtävä oli varsin yksinkertainen ja sainkin sen nopeasti tehtyä. Testasin nopeasti, että listat järjestyvät oikein tuotantopalvelimella ja siirsin valmiin paketin testipalvelimelle. Testipalvelimella huomasin kuitenkin heti pienen huolimattomuudesta johtuvan ongelman. Eräs lista sovelluksessa oli tarkoitus järjestää sukunimien mukaiseen aakkosjärjestykseen. En kuitenkaan huolimattomuuttani tullut ajatelleeksi, että mikäli sukunimeä ei ole lainkaan asetettu, hajoaa koko lista. Lisäsin koodiin tarkistuksen tyhjien sukunimien varalle, käänsin koodin uudelleen ja vein uuden version stagingille.

Vaikka kyseessä ei ollut iso ongelma ja se oli helppo korjata, oli se kuitenkin hyvä muistutus huolellisuuden tärkeydestä. Onneksi staging palvelimella oli käyttäjä, jonka sukunimi

oli jäänyt tyhjäksi, joten ongelma ilmeni vielä testausvaiheessa. Jatkossa pyrin olemaan huolellisempi ja ottamaan myös tällaiset harvinaisemmat tapaukset, jotka saattavat hajottaa koko sovelluksen, paremmin huomioon. Olen huomannut, että mitä enemmän työtä tekee, sitä paremmin tulee huomioineeksi myös tällaiset tapaukset, ja olenkin kehittynyt myös tässä paljon viimeisen vuoden aikana.

Viikkoanalyysi

Kuudes viikko sisälsi uusien ominaisuuksien testaamista ja vikojen paikantamista staging palvelimella. Tämä on tärkeä osa uusien ominaisuuksien kehitystä, sillä staging palvelimella uusia ominaisuuksia päästään testaamaan ensimmäistä kertaa tuotantoympäristön kaltaisessa ympäristössä. Kehityspalvelimilla tehdään jatkuvasti ohjelmistokehitystä ja niillä käytettävät ympäristöt ja ohjelmistot muuttuvat jatkuvasti. Staging palvelin sen sijaan pidetään käytännössä identtisenä tuotantopalvelimen kanssa, jolloin voidaan mahdollisimman hyvin varmistaa uusien ominaisuuksien toimivuus myös tuotantoympäristössä. Tärkeä osa eri tätä prosessia on versionhallinta, joka mahdollistaa eri sovellusversioiden kehittämisen erillään toisistaan, sekä eri ympäristöihin tarkoitettujen sovellusten versioiden hallinnan.

Yrityksessä, jossa työskentelen, käytetään versionhallintaan Gitiä. Git on avoimen lähdekoodin hajautettu versionhallintajärjestelmä (Git 2019). Perusperiaate Gitissä on, että yhdestä projektista, esimerkiksi sovelluksen lähdekoodista, on Gitissä useita haaroja (brancheja). Tämä mahdollistaa usean eri kehittäjän työskentelemisen saman projektin parissa ilman, että kehittäjien kirjoittama koodi olisi hallitsemattomasti päällekkäistä toisten kehittäjien kirjoittaman koodin kanssa. Tiimin kesken tämä on toteutettu niin, että mikäli saman projektin parissa työskentelee useampi kehittäjä, on heistä jokaisella oma kehityshaara. Kun kaikki ovat saaneet tahoillaan ominaisuutensa valmiiksi, yhdistetään (merge) koodi yhteiseen kehityshaaraan, jolloin Git ilmoittaa mahdollisista koodikonflikteista ja mahdollistaa niiden ratkaisemisen. Kehittäjät tekevät muutoksia koodiinsa paikallisesti ja tämän jälkeen tallentavat (commit), muutokset paikalliseen Git hakemistoonsa. Tämän jälkeen muutokset voi työntää (push) esimerkiksi palvelimella sijaitsevaan Git hakemistoon, jolloin tehdyt muutokset päätyvät koko tiimin kehittäjien nähtäville. Git tallentaa myös muutosten historian, mikä ongelmien ilmetessä mahdollistaa takaisin päin palaamisen.

Usean kehittäjän yhteistyön lisäksi Gitin haarat mahdollistavat myös projektin eri versioiden tallentamisen ja hallinnan. Useimmissa projekteissa, joiden parissa työskentelen, on Git hakemistossa omat haarat kehitys-, staging- ja tuotantoympäristöihin tarkoitetuille projektiversioille. Tämän vuoksi eri versiot pysyvät selkeästi erillään ja muutoksia voidaan tehdä oikeaan paikkaan ilman pelkoa esimerkiksi tuotantoversion hajoamisesta. Uuden

ominaisuuden matka kehityksestä tuotantoon tapahtuu yleensä niin, että ensin teen muutoksen omaan kehityshaaraani, jonka alkuvaiheessa tulisi olla ajan tasalla yleisen kehityshaaran kanssa. Tämän jälkeen yhdistän muutoksen yleiseen kehityshaaraan, jolloin uusi sovellusversio saadaan käyttöön yleiselle kehityspalvelimelle. Testiviikolle siirryttäessä yleinen kehityshaara yhdistetään edelleen staging haaraan, jolloin uusi versio saadaan käyttöön staging palvelimella. Kun sovellusversio on testattu stagingillä ja julkaistaan, yhdistetään se tuotantohaaraan, jolloin uusi versio muutoksineen saadaan näkyviin tuotantopalvelimilla. Staging ja tuotantoympäristöjen ollessa käytännössä identtiset vältytään ikäviltä yllätyksiltä tuotantovaiheeseen siirryttäessä.

3.7 Seitsemäs viikko

Maanantai

Maanantaiaamupäivä kului uuden sprintin aloittavassa sprinttipalaverissa. Testaaminen päätettin suorittaa loppuun pienemällä joukolla ja julkaista uudet sovellukset myöhemmin tällä viikolla, muiden aloittaessa jo sprinttinsä uusien asioiden parissa. Itse tulisin olemaan mukana alkuviikon julkaisua varten testaamisessa, jonka jälkeen ryhtyisin tekemään sprinttini tehtäviä. Varsinaiset tehtäväni sprintissä olivat varsin vähäisiä, joten tarkoituksenani olisi olla saatavilla kaikkia sprintin aikan ilmeneviä niin sanotusti ylimääräisiä tehtäviä varten. Samalla voisin tehdä vähemmän tärkeitä ominaisuuksia, jotka eivät ole olleet kiireisiä ja ovat täten jääneet toteuttamatta. Tälle päivälle ja alkuviikolle tavoitteeni oli kuitenkin testata mahdollisimman kattavasti testipalvelimella.

Iltapäivällä testipalvelimella testatessani löysin backend bugin, jota luulin vain pieneksi ongelmaksi, mutta joka osoittautuikin hieman suuremmaksi uudesta muutoksesta johtuvaksi ongelmaksi, jota ei ollut huomattu tai huomioitu aikaisemmissa testeissä. Raportoin ongelmasta kollegalleni ja hän ryhtyi välittömästi korjaamaan sitä. Korjaaminen kuitenkin venyisi todennäköisesti huomiseen.

Tiistai

Tiistaina kollegani kertoi saaneensa eilen löytämäni bugin korjattua eilisiltana. Testasimme sitä hieman, jolloin ilmeni vielä, että korjaus oli aiheuttanut pienen ongelman iOS sovellukseen. Tämä oli kuitenkin helposti korjattavissa ja kokeneempi kollegani tekikin sen nopeasti.

Latasin uusimman testiversion mobiilisovelluksesta ja varmistin erilaisten hälytysten ja muiden toimivan edelleen kuten pitääkin. Kaikki näytti hyvältä, ja raportoin mobiilikehittäjälle, etten ainakaan minä löytänyt mitään ongelmia.

Olen oppinut, että erilaiset ongelmat ja bugit ovat osa työtä, eikä niiltä voi välttyä, vaikka kuinka yrittäisi. Aina tulee esiin tapauksia, joita ei tule huomioineeksi ja etenkin kovassa kiireessä on välillä hankala miettiä kaikkia uusia muutoksia monelta eri kantilta. Etenkin alussa koin aina hieman epäonnistuneeni, mikäli tekemistäni sovelluksita löytyi ongelmia, mutta nyt olen oppinut niiden kuuluvan asiaan. Samalla olen tietysti myös oppinut paremmin välttämään ilmeisimpien virheiden tekemistä uusia ominaisuuksia tehdessä. Lisäksi pyrin aina tuottamaan mahdollisimman selkeää ja hyvin suunniteltua koodia, jottei muualla ilmene odottamattomia ongelmia uutta ominaisuutta tehdessä.

Keskiviikko

Aloitin keskiviikon testaamalla vielä hieman stagingilla, sillä uudet sovellukset olisi tarkoitus julkaista tänään. Otinkin päivän tavoitteekseni siis varmistaa, että kaikki on kunnossa ja julkaisuvalmiina, jotta julkaisu saadaan hoidettua ajallaan, jonka jälkeen siirtyisin seuraavan sprintin tehtäviin. Kaikki näytti hyvältä, ja ilmoitin sovellukseni olevan valmis julkaistavaksi.

Myöhemmin, kollegani hoidettua julkaisut tuotantopalvelimelle, löytyi tuotantopalvelimella web-sovelluksesta vielä pieni bugi. Ongelma oli onneksi yksinkertainen ja sain sen nopeasti korjattua, eikä julkaisu viivästynyt.

Iltapäivällä aloittelin uuden sprinttini tehtäviä pitämällä palaverin kollegani kanssa. Backendimme rajapintoja aletaan yhdenmukaistamaan, mikä vaatisi useita muutoksia myös web-sovelluksiimme, jotka niitä käyttävät. Keskustelin kollegani kanssa siitä, millaista kaavaa API:t tulevat tulevaisuudessa noudattamaan, jolloin niiden käyttö ja rakenne tulisi olemaan johdonmukaisempaa ja selkeämpää.

Olen työskennellessäni päässyt olemaan mukana jo monessa julkaisussa, mutta edelleen julkaisupäivät ovat hieman jännittäviä ja stressaavia. Kokemukseni karttuessa ja itseluottamukseni kasvaessa, olen kuitenkin pystynyt olemaan melko varma, että pystyn julkaisupäivään mennessä tuottamaan toimivia ominaisuuksia. Julkaisuvaiheessa löytynyt bugi jäi kuitenkin hieman ärsyttämään, joten yritän tulevaisuudessa olla vielä tarkempi ja huolellisempi uusia ominaisuuksia kirjoittaessani.

Torstai

Torstaina sain kollegaltani listan muutamista API-muutoksista, ja ryhdyin tekemään muutoksia sovellukseen niiden mukaisesti. Päivän tavoitteeni olikin siis API-kutsujen muuttaminen uusien muutosten kanssa yhteen sopiviksi, sekä sivussa hoitaa omia pienempä tehtäviäni.

API muutosten teko oli vaivatonta, joten päätin samalla toteuttaa myös muita muutoksia sovellukseen. Esimerkiksi sovelluksen sisäiset päivämäärät tulisi saada näytettyä käyttäjän maan mukaisessa formaatissa. Sovellus käyttää käyttäjän sijaintitietoja jo erinäisiin tarkoituksiin, joten päivämäärien muuttaminen moment.js -kirjaston avulla onnistui helposti.

Perjantai

Perjantaina jatkoin jälleen pienten muutosten parissa, jotta sovellus pysyy ajan tasalla backendin API-muutosten mukaisesti. Päivän tavoitteeni oli siis kutakuinkin sama kuin torstaina. Web sovelluksen muutokset API-muutosten osalta ovat pieniä, mutta kun muutokset koskevat lähes joka API:a, jota sovellus käyttää, on pieniä muutoksia tehtävä runsaasti.

Iltapäivällä ryhdyin valmistelemaan sovellusta uuden Angular version päivitystä varten. Tällä hetkellä sovellus käyttää Angular versio 5:ä ja uusi versio 7 julkaistiin jo jokin aika sitten. Sovellus oli siis kaksi kokonaista Angular versiota jäljessä. Onneksi Angular tarjoaa hyvän ohjeen sovellusten päivittämiseen Angular versiosta toiseen. Kävin listalta läpi asiat, jotka sen mukaan tuli muuttaa ennen sovelluksen päivittämistä, jotta voisin maanantaina aloittaa päivittämisen. Vaikka Angularin ohjeen mukaan päivittäminen tulisi olemaan helppoa ja suoraviivaista, arvelin kokemukseni pohjalta, että päivittäessä kuitenkin ilmenee jonkin verran ongelmia ja varauduin viettämään sen parissa ainakin osan seuraavasta viikosta.

Viikkoanalyysi

Seitsemäs seurantaviikko sisälsi paljon yrityksen rajapintojen yhdenmukaistamisprojektiin liittyviä tehtäviä. En ole aikaisemmin tehnyt varsinaista API-suunnittelua, eikä esimerkiksi rajapintojen nimeämiseen ole tähän mennessä kiinnitetty tiimin sisäisesti juurikaan huomiota. Nyt järjestelmän koko ajan kasvaessa on tämä kuitenkin muodostunut ongelmaksi, sillä eri API:t ja endpointit eivät noudata minkäänlaista loogista rakennetta. Tästä syystä niiden kanssa työskentely ja käyttäminen on työlästä. Ne eivät myöskään aina vastaa REST arkkitehtuurissa käytettyjä nimeämiskäytäntöjä ja käyttötapoja.

REST arkkitehtuurimallin mukaan resurssien nimeämisen tulisi olla johdonmukaista. Tämä helpottaa rajapintojen kanssa työskentelyä huomattavasti. Nimeämisessä tulee käyttää kauttaviivaa resurssien hierarkkisen järjestyksen osoittamiseen. (RESTfulapi.net 2019.) Tällöin eri kutsujen toiminta ja kohde on helposti ymmärrettävä. Esimerkiksi kuviossa 16 toinen rivi käsittelee käyttäjiä kokonaisuutena ja *kolmas rivi* yksittäistä käyttäjää.

http://api.esimerkki.com/user-management
http://api.esimerkki.com/user-management/users
http://api.esimerkki.com/user-management/users /{id}
http://api.esimerkki.com/user-management/users /{id}/accounts
http://api.esimerkki.com/user-management/users /{id}/accounts/{id}
KUVIO 16. Resurssien nimeäminen

REST:n mukaan nimeämisessä ei tulisi käyttää verbejä, vaan haluttava toiminta tulisi kertoa HTTP-metodin avulla (RESTfulapi.net 2019). Esimerkiksi, mikäli tietyn käyttäjän kaikki tilit halutaan hakea, on esimerkiksi kutsun ”getUserAccounts” käyttäminen tämän mukaan väärin. REST:n hyvien käytäntöjen mukaisesti tämä tehdään lähettämällä GET kutsu kuvion 16 neljänne rivin osoitteeseen. HTTP metodien käyttötarkoituksia ei REST:ssä ole varsinaisesti määritelty, mutta niiden käytön tulisi olla yhdenmukaista koko sovelluksessa. Yleisesti hyväksi todetun käytännön mukaan GET kutsua kuitenkin käytetään resurssin lukemiseen tai hakemiseen. POST kutsua sen sijaan käytetään resurssin lisäämiseen, PUT kutsua resurssin päivittämiseen ja DELETE kutsua resurssin poistamiseen. REST arkkitehtuuri määrittelee myös, että resurssien nimissä tulisi käyttää pelkästään pieniä kirjaimia ja esimerkiksi luettavuuden helpottamiseksi tulisi käyttää vain väliviivaa, ei alaviivaa tai camelCasea. (RESTfulapi.net 2019.)

Nyt kun rajapintoja päivitetään, on ne päätetty tiimin kesken toteuttaa niin, että ne noudattaisivat etenkin nimeämiskäytännöiltään REST arkkitehtuurin määritelmiä. Tulen kiinnittämään tähän huomiota myös omassa työssäni, sillä koen, että erityisesti johdonmukainen ja hierarkkinen nimeäminen selkiyttää API kokoelmaa ja sen kanssa työskentelyä huomattavasti.

3.8 Kahdeksas viikko

Maanantai

Maanantai aamupäivä kului jälleen maanantaipalaverissa. Selitin viimeviikolla olleeni alkuviikon mukana testaamisessa julkaisua varten, ja sen jälkeen aloitelleeni web-sovelluksen yhteensovittamista kollegani tekemien rajapintamuutosten kanssa. Viikon tavoitteekseni kerroin sovelluksen päivittämisen Angular 5:stä Angular 7:än, sekä mahdollisesti jatkamisen API-muutosten parissa. En osannut oikeastaan antaa arviota, siitä kuinka kauan päivittäminen kestää, sillä parhaimmillaan se saattaisi tapahtua helposti. Todennäköisesti se vaatisi kuitenkin ainakin muutaman päivän työtä, sillä päivitettävä sovellus on laaja.

Etukäteen olin varautunut siihen, että jo Angular 6 yhteydessä julkaistun uuden RxJS kirjaston toimimaan saaminen sovelluksen kanssa vaatisi eniten työtä, sillä se oli niin sanottu breaking change, joka vaatisi oikeastaan jokaisen kohdan korjaamista, jossa RxJS kirjastoa käytetään. Seurasin Angularin virallista päivitysopasta, jossa onneksi selitettiin hyvin askel askeleelta, mitä sovellusta päivitettäessä täytyy tehdä ja ottaa huomioon. Muutin sovelluksen HTTP-kutsut Angularin uuden HTTP-kirjaston mukaisiksi, joka sovelluksen laajuudesta johtuen vei paljon aikaa.

HTTP-kutsujen päivittäminen oli yksitoikkoista, mutta samalla opin ymmärtämään niiden toimintaa paremmin. Yritin aluksi tehdä muutokset, kuten ohjeessa neuvottiin, mutta tämä aiheutti vain suuren määrän virheilmoituksia sovellusta ajaessa. Niinpä jouduin selvittämään ja tutkimaan enemmän, mitä juuri tämän sovelluksen kohdalla oikeasti tulisi muuttaa. Kyseessä oli lopulta pieni asia, ja virheilmoitukset johtuivat lähinnä siitä, että yritin liian sokeasti seurata geneeristä ohjetta. Koin silti oppineeni jonkin verran uutta.

Tiistai

Tiistaina jatkoin Angular-sovelluksen päivittämistä. Päivän tavoitteekseni otinkin, että saisin sovelluksen siihen kuntoon, että se kääntyisi ilman virheitä Angular päivityksen jälkeen. Tämän jälkeen voisin ryhtyä selvittämään uuden RxJS-kirjaston toimintaa ja sen vaatimia muutoksia.

Eilen tekemäni HTTP-kutsujen muutokset näyttivät nyt toimivan ja pääsin päivittämään varsinaisen Angular core-paketin. Varsinainen päivitysvaihe sujui helposti, sillä se vaati vain viimeisimpien Angular ja ng-cli pakettien lataamista npm:llä. Ladattuani vaaditut paketit, yritin kääntää sovelluksen koodin ja sain odotetusti liudan virheilmoituksia liittyen RxJS-kirjastoon. Latasin vielä rxjs-compat nimisen paketin, jonka avulla vanha RxJS syntaksi toimisi uuden kirjaston kanssa, ja sain koodin käännettyä. Rxjs-compat pakettia kehotettiin kuitenkin käytettävän vain siirtymävaiheessa uuteen kirjastoon ja tulevissa päivityksissä sitä lakattaisiin tukemasta. Katsoin siis parhaaksi hoitaa muutokset uuteen muotoon heti.

Angularin päivitysohje kehotti lataamaan pienen ohjelman, jonka avulla RxJS:ää käyttävät kohdat koodissa saisi muutettua helposti kerralla, vain yhdellä komennolla. Todellisuudessa ohjelma löysi vain pienen osan vaadittavista muutoksista ja jouduin tekemään suurimman osan muutoksista manuaalisesti. Tutkin asiaa eri keskustelufoorumeilta ja huomasin muidenkin kokeneen saman kohtalon yrittäessään toimia päivitysohjeen mukaan. Niinpä ryhdyin päivittämään koodia manuaalisesti.

Koin tavallaan jopa hyväksi asiaksi, ettei automaattinen päivitys toiminut ja jouduin tekemään suurimman osan muutoksista käsin. Näin jouduin lukemaan läpi RxJS dokumentaatiota ja oppimaan asioita, joihin en todennäköisesti olisi paneutunut, mikäli automaattinen päivitys olisi toiminut. Ymmärsin paremmin, miten RxJS kirjastoa on tarkoitus käyttää ja millä tavalla voin hyödyntää sitä, joka osaltaan vaikuttaa suoraan työskentelyyni.

Keskiviikko

Aloitin keskiviikon asettamalla tavoitteekseni, että saisin päivityksen RxJS-muutosten osalta valmiiksi, jotta saisin koodin kääntymään ilman rxjs-compat pakettia. Jos aikaa jäisi, voisin vielä kokeilla, että kaikki toimii, kuten ennen päivitystä. Näin voisin varmistaa, ettei päivityksen myötä hajonnut ainakaan mitään kriittistä, ja laittaa päivitetyn sovelluksen kehityspalvelimelle muidenkin käytettäväksi.

Eilen oppimani perusteella olin hyvin perillä siitä, millaisia muutoksia uusi RxJS kirjasto vaatii sovelluksen koodissa. Poistin rxjs-compat-paketin, jotta sain jälleen päivitetyn kirjaston aiheuttamat virheet näkyviin ja ryhdyin käymään niitä läpi. Jouduin vielä katsomaan apua RxJS dokumentaatiosta, mutta päästyäni asiasta kärryille, sujui päivittäminen suoraviivaisesti. Aikaa tähän kului kuitenkin lähes koko päivä, sillä sovelluksen laajuudesta johtuen jouduin tekemään paljon pieniä muutoksia. Tehtyäni muutokset sovelluksen koodiin, antoi kääntäjä vielä virheilmoituksia muutamasta sovelluksen käyttämästä kirjastosta, jotka käyttivät RxJS-pakettia vielä vanhaan tapaan. Tutkin, onko kirjastoista uusia, RxJS 6:ta tukevia versioita, ja päivitin ne uusimpiin versioihin. Ainoastaan yhdestä kirjastosta en löytänyt RxJS 6:ta tukevaa versiota, eikä kirjastoa muutenkaan ollut päivitetty enää hetkeen. Löysin helposti kuitenkin vastaavaan uuden kirjaston, ja sain sen toimimaan sovelluksessa vanhan kirjaston tapaan pienillä muutoksilla.

Torstai

Torstaina jatkoin edelleen Angular päivitystä. Olin eilen saanut RxJS muutokset tehtyä, joten sain nyt koodin käännettyä ilman rxjs-compat-pakettia ja kaikki näytti toimivan kuten ennenkin.

Etukäteen Angular 7:n tuomia uudistuksia tutkiessani olin innostunut, uudesta Angular CDK ominaisuudesta, virtuaalisesta vierityksestä. Käytännössä virtuaalinen vieritys tarkoittaa sitä, että rullattavasta listasta renderöidaan vain tietyllä hetkellä ruudulla näkyvät elementit. Tavallisesti kaikki listan elementit renderöidaan kerralla, mikä saattaa aiheuttaa suorituskykyongelmia. mikäli lista on suuri. Tämä onkin ollut pieni ongelma sovelluksessa, sillä mikäli eräät listat ovat suuria ovat ne saattaneet aiheuttaa pientä nykimistä listaa ladatessa. Ongelma ei ole ainakaan nykyisen kokoisilla listoilla ollut niin suuri, että sitä olisi

koettu tarpeelliseksi ratkaista esimerkiksi jakamalla listan elementit useammalle sivulle, mutta koin virtuaalisen rullauksen sen verran helposti implementoitavaksi ratkaisuksi, että päätin toteuttaa sen päivityksen yhteydessä, nyt kun se kerran tuotiin uutena ominaisuutena päivityksen mukana.

Käytännössä virtuaalisen vierityksen implementointi oli helppoa, sillä se vaati vain pienen muutoksen listojen syntaksiin. Tässäkin tapauksessa muutos tuli tehdä kuitenkin useampaan paikkaan, ja muutokset vaativat myös hieman muutoksia ympäröiviin elementteihin, jotta sain uudet listat näyttämään samalta kuin ennen, hajottamatta muuta käyttöliittymää.

Perjantai

Aloitin perjantain käymällä läpi sovellusta ja varmistellen, että päivityksen jälkeen kaikki näyttäisi toimivan kuten pitää. Tähän asti olin käyttänyt päivitettyä sovellusta vain omalla paikallisella koneellani, joten päätin siirtää sen nyt myös kehityspalvelimellemme. Ilmoitin muille kehityspalvelimella olevan nyt Angular 7:än päivitetty sovellus ja raportoimaan minulle, mikäli he havaitsisivat ongelmia sovelluksen toiminnassa.

Backendin rajapintamuutoksia tekemäni kollegani oli saanut kuluneen viikon aikana paljon aikaan, joten asetin päiväni tavoitteeksi jatkavani sovelluksen muuttamisesta niiden mukaiseksi. Sain kollegaltani Postman kokoelman muutetuista API:sta, josta tietysti näin myös miten niitä käytetään. Rupesin käymään muutoksia läpi API kerrallaan ja törmäsin lähes heti muutokseen, joka vaatisi hieman enemmän pohtimista.

AIkaisemmin eräiden listojen filtteröinti sovelluksessa on toteutettu niin, että backendiin on lähetetty kutsu, joka on palauttanut vaaditun listan. Uusien muutosten mukaisesti tämä ei kuitenkaan ollut mahdollista kovin helposti, joten päätimme kollegani kanssa, että suoritan filtteröinnin kokonaisesta listasta sovelluksessa itsessään. Tämä on itseasiassa muutos, jonka olen halunnut tehdä jo pidempään, sillä uuden kutsun lähettäminen vain listojen suodattamiseksi on tuntunut hieman turhalta. Nyt sain hyvän syyn muuttaa sen ja mielenkiintoisen ongelman ratkaistavakseni.

Loppupäivän tavoitteeksi siis muodostuikin filtteröinnin toteuttaminen ainakin yhteen listoista, sillä puumallisesta datasta johtuen filtteröinti täytyi tehdä rekursiivisesti. Tämä vaati jonkin verran pohtimista, ja sainkin pari kertaa koko sovelluksen hajotettua, kun filtteröintifunktio ei ymmärtänyt lopettaa listan läpi käymistä uudelleen ja uudelleen, vaikka osumia ei enää löytynyt. Sain listan filtteröinnin lopulta toimimaan. Olisin voinut kysyä heti apua kollegaltani, mutta halusin ratkaista mielenkiintoisen ongelman itse ja pystyin siihen vieläpä kohtuullisessa ajassa. Oli hienoa ratkaista pulma itse, ja jälleen huomata kehittyneeni ohjelmistokehittäjänä.

Viikkoanalyysi

Mennyt viikko kului lähinnä vanhan sovelluksen päivittämisessä uuteen Angular versioon. Päivitys vei hieman enemmän aikaa, kuin olin alun perin ajatellut. Olin kuitenkin osannut varautua tähän, ja ilmoitin jo maanantain palaverissa päivitykseen kuluvan mahdollisesti jopa koko viikko. Olen huomannut, että erilaiset päivitykset sovellusten ja kirjaston päivitykset aiheuttavat useinkin odotettua suurempia ongelmia. Etenkin erilaiset yhteensopivuusongelmat eri kirjastojen versioiden välillä, ovat usein muodostuneet ongelmallisiksi päivitysten yhteydessä.

Suuri osa päivityksen ongelmista liittyi RxJS kirjaston uuteen versioon, jota uudempi Angular versio oli siirtynyt käyttämään. RxJS on JavaScript-kirjasto reaktiivista ohjelmointia varten. Se tarjoaa useita työkaluja asynkronisten ja tapahtumakeskeisten sovellusten luomiseen. Käytännössä tämä tarkoittaa sitä, että RxJS kirjaston avulla erilaisia tapahtumia voidaan tarkkailla ympäri sovellusta ja reagoida niihin toivotulla tavalla riippumatta muista tapahtumista tai siitä, milloin tapahtuma laukaistaan.

Keskeinen tyyppi, jonka kirjasto tarjoaa, on nimeltään Observable. Periaatteessa Observablet tarjoavat tuen erilaisten viestien välittämiseen sovelluksen sisällä, niin kutsutuilta julkaisijoilta tilaajille. RxJS kirjasto tarjoaa Observable-tyypin käsittelyyn ja tarkkailuun useita eri tyyppejä ja operaatioita, jotka mahdollistavat asynkronisten tapahtumien moninaisen käsittelyn. Perusperiaate on, että Observablet ovat niin sanottuja datavirtauksia, joita voidaan tilata eri puolilta sovellusta. (RxJS 2019.) Tilaajat taas reagoivat Observablen muutoksiin asynkronisesti ja suorittavat niille määriteltyjä toimintoja muutosten mukaisesti (kuvio 17).

```
import { Observable } from 'rxjs';

const exampleObservable = new Observable(subscriber => {
  subscriber.next('Esimerkki');
});

exampleObservable.subscribe(x => {
  console.log(x);
});

// Output: Esimerkki
```

KUVIO 17. Esimerkki Observablesta ja tilaajasta.

Keskeinen osa RxJS kirjaston työkaluvalikoimaa ovat myös erilaiset operaattorit, jotka mahdollistavat tilatun arvon käsittelyn halutulla tavalla ennen kuin se varsinaisesti saapuu

tilaajan käsiteltäväksi. Näitä operaattoreita voidaan putkittaa yhteen useita, jolloin tilatusta tapahtumasta tilaajalle saapuva arvo kulkee ensin jokaisen operaattorin läpi. Operaattoreista jokainen käsittelee arvoa sille määritellyllä tavalla. Lopulta operaattoreiden läpi kulkenut tilaus saapuu operaattoreiden mukaisesti muunneltuna tilaajalle (kuvio 18).

```
import { fromEvent } from 'rxjs';
import { throttleTime, scan } from 'rxjs/operators';

fromEvent(document, 'click')
  .pipe(
    throttleTime(1000), // Estää uusien tapahtumien pääsyn tilaajalle määriteltyä aikaa useammin
    scan(count => count + 1, 0) // Lisää edelliseen arvoon yhden, alkaen nollasta
  )
  .subscribe(count => console.log(`Clicked ${count} times`));
```

KUVIO 18. Tapahtuma putkitetaan operaattoreiden läpi.

RxJS-kirjasto on tärkeä osa Angularia, ja se käyttää kirjastoa sisäisesti. RxJS kirjaston tarjoamia tyyppejä ja operaattoreita käytetään laajasti myös Angular sovelluksissa tietojen käsittelyyn ja välittämiseen sovelluksen sisäisesti. (Angular 2019.)

RxJS kirjaston uuden version aiheuttamat ongelmat pakottivat minut perehtymään tarkemmin kirjaston toimintaan ja reaktiivisen ohjelmoinnin periaatteisiin ylipäätään. Koen viikon olleen opettavainen, sekä ymmärtäväni RxJS-kirjastoa, ja reaktiivista ohjelmointia ylipäätään, entistä paremmin. Vaikka olen aikaisemminkin käyttänyt RxJS kirjastoa Angularin kanssa työskennellessä, en välttämättä ole aina täysin ymmärtänyt sen toimintaa, vaan olen saattanut mennä vain valmiiden esimerkkien mukaan asiaa enempää miettimättä. Viikon aikana oppimani myötä pystyn jatkossa hyödyntämään kirjaston tarjoamia ominaisuuksia tehokkaammin ja järkevämmin. Koen myös, että parempi ymmärrys reaktiivisesta ohjelmoinnista on tärkeää ohjelmistokehittäjän urani kannalta.

3.9 Yhdeksäs viikko

Maanantai

Jälleen kerran maanantai alkoi jokaviikkoisella maanantaipalaverilla, jossa käytiin läpi viimeviikolla tehdyt asiat, sekä alkavan viikon tavoitteet. Omalta osaltani kerroin toteuttaneeni Angular päivityksen onnistuneesti, sekä jatkaneeni API-muutosten parissa. Tällä viikolla jatkaisin API:en kanssa, ja siihen tulisi todennäköisesti kulumaan koko viikko. Uusia

ja muutettuja rajapintoja oli näet runsaasti, ja kuten huomasin viimeviikolla, osa niistä saattaa vaatia hieman suurempiakin muutoksia.

Asetin päiväni tavoitteeksi filtteröinnin toteuttamisen myös muihin listoihin, viime viikolla toteuttamani lisäksi. Muut listat eivät tulisi olemaan yhtä haastavia, eivätkä vaatisi rekursiivista filtteröintiä, kuten viime viikolla tekemäni lista. Varmistin vielä kollegaltani, että olemme kyseisten API:en toteutuksesta samaa mieltä ja ryhdyin suorittamaan omia tehtäviäni.

Filtteröinnin toteuttaminen loppuihin listoihin sujui mutkattomasti. Muutokset vaativat hieman päivityksiä sovelluksen logiikkaan, mutta kutakuinkin samanlainen ratkaisu päti jokaiseen listaan. Viime viikkoiseen tehtävään verrattuna nämä tuntuivat yksinkertaisilta, mutta koin saaneeni paljon aikaan ja selkeyttäneeni sovelluksen toimintaa, kun listat filtteröidään nyt sovelluksen päässä, eikä backendiin tarvitse tehdä uutta kutsua joka kerta, kun valintaa muutetaan.

Tiistai

Tiistai alkoi kiireisesti, kun tuotantopalvelimella olevasta web-sovelluksesta oli löytynyt pieni bugi. Bugi oli vähän käytetyssä ominaisuudessa, joten se oli jäänyt testatessa huomaamatta. Päiväni tavoitteeksi tulikin siis bugin korjaaminen, ja mahdollisesti API-muutosten parissa jatkaminen.

Bugin havainnut kollegani näytti minulle, mistä bugi löytyy ja miten sen saa toistettua. Pystyin jo tässä vaiheessa sanomaan suhteellisen varmasti, mistä bugi johtuu, joten ryhdyin heti töihin. Ongelma oli, että eräs aikaleima lähetettiin väärässä formaatissa. Jokin aika sitten tehdyssä päivityksessä, jossa aikaleimojen formaatti yhdenmukaistettiin koko sovelluksen laajuisesti, oli tämä vähän käytetty ominaisuus unohtunut päivittää, eikä se sen takia toiminut. Sain bugin korjattua nopeasti ja varmistin vielä, että se toimii. Kaikki näytti hyvältä, joten korjaus voitiin siirtää myös tuotantopalvelimelle.

Olen oppinut, että näin pienessä yrityksessä ja kovalla temmolla uusia ominaisuuksia tehdessä, bugeja löytyy välillä vielä tuotantovaiheessa. Huomaan, että kokemukseni karttuessa, myös tällaiset kiireellisemmätkin asiat hoituvat niin sanotusti rutiinilla, eivätkä aiheuta niin paljon stressiä kuin alussa. Olen myös työskennellyt kyseisen sovelluksen parissa niin kauan, että pystyin heti bugin nähtyäni sanomaan, mikä sen aiheuttaa.

Keskiviikko

Keskiviikkona jatkoin jälleen API-muutosten parissa. Otin päivän tavoitteekseni siis niiden tekemisen, sekä samalla koodia läpi käydessäni, sen siistimisen ja niin sanotun kuolleen

koodin poistamisen. Tähän tarjoutui pienempiä API-muutoksia tehdessä oiva tilaisuus, sillä muutoksia piti tehdä useaan paikkaan, joten pystyin samalla käymään sujuvasti läpi paljon koodia.

Osa API-muutoksista vaatikin hieman suurempia toimenpiteitä, sillä jotkin API:t eivät enää toimineet aivan samalla tavalla kuin ennen. Jouduin keskustelemaan näiden osalta backendin rajapintamuutoksia tekevän kollegani kanssa ja päädyttyämme kumpaakin miellyttävään lopputulokseen ryhdyin tekemään sovellusta yhteensopivaksi uusien rajapintojen kanssa. Loppujen lopuksi muutokset selkiyttivät ja suoraviivaistivat tiettyjä toimintoja, joten pystyin karsimaan paljon turhaa koodia näistä paíkoista.

Kokemukseni koko ajan karttuessa huomaan pystyväni hahmottamaan paremmin sovelluksia ja niiden toimintaa kokonaisuutena ja tekemään parempia ja selkeämpiä ratkaisuja siltä pohjalta. Olen myös oppinut ymmärtämään backendin toimintaa yhdessä sovelluksien kanssa paremmin, joka tietysti osaltaan myös helpottaa työtäni ja auttaa tekemään parempia ja toimivampia ratkaisuja.

Torstai

Torstaina jatkoin edelleen API-muutosten parissa. Olin saanut kutakuinkin kaikki kollegani tähän mennessä tekemät backend muutokset kurottua kiinni, joten päätimme pitää pienen palaverin seuraavien osioiden vaatimista muutoksista ja API:en käyttötarkoituksista. Pääsin jälleen vaikuttamaan ja antamaan mielipiteeni, miten tiettyjen asioiden tulisi omasta näkökulmastani frontend kehittäjänä toimia ja usein päädyimmekin ratkaisuihin, jotka tekivät molempien työstä helpompaa ja koodista selkeämpää ja suoraviivaisempaa. Asetin päiväni tavoitteeksi sopimiemme muutosten toteuttamisen yhteistyössä kollegani kanssa, joka osaltaan tulisi kehittämään myös yhteistyötaitojani.

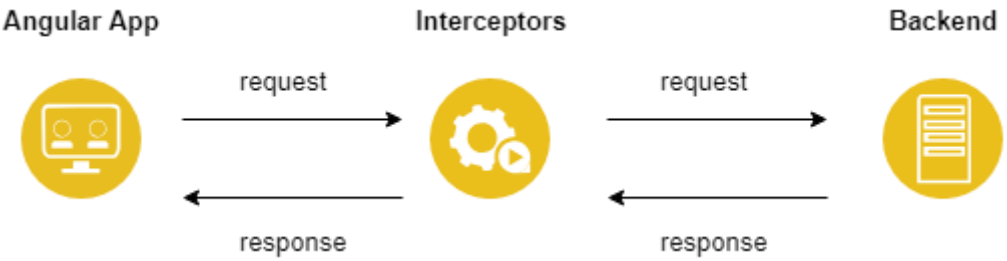
Palaverin jälkeen ryhdyin toteuttamaan sovittuja muutoksia sovelluksen päässä tiiviisti backendin muutoksista vastaavan kollegani kanssa. Välillä vastaan tuli suunniteltua hankalampia ongelmia, joita jouduimme taas miettimään yhdessä ja viilaaman APIen haluttua toimintaa vielä hieman.

Päivä kului oikeastaan kokonaan kollegani kanssa yhteistyötä tehden, mikä oli mukavaa vaihtelua tavalliseen. Keskustelimme joka muutoksesta yhdessä ja mietimme niitä monelta eri näkökulmalta, joka toimi minulle myös erinomaisena oppimiskokemuksena, kun pääsin näkemään ja kuulemaan, miten minua kokeneempi kehittäjä näkee nämä asiat. Edelleen myös käsitykseni backendistämme ja sen toiminnasta kehittyy koko ajan ja koenkin oppineeni päivän aikana runsaasti.

Viikkoanalyysi

Suurin osa viikosta kului, muutamaa pienempää tehtävää ja bugikorjausta lukuun ottamatta, web-sovellusten saattamisessa yhteensopivaksi backendiin tehtyjen rajapintojen yhtenäistämiseen liittyvien muutosten kanssa. Toisaalta tämä oli itseään toistavaa työtä, joka ei tuntunut kovin kehittävältä. Vastaan tuli myös pieniä haasteita, kun pieneltä tuntunut muutos vaikkapa API:n vastaustiedoissa aiheuttikin odotettua suuremman ongelman itse sovelluksen päässä. Nämä olivat hyviä oppimiskokemuksia, joissa pääsin hyödyntämään ongelamanratkaisutaitojani, sekä keskustelemaan muutoksista kokeneempien backend-kehittäjien kanssa.

Eräs backendin muutosten takia esiin noussut ongelma liittyi Angular-sovelluksen virheenkäsittelyyn, minkä takia päädyin oppimaan uutta Angularin http interceptoreista. Http interceptorin perusidea on, että kaikki moduulista tehdyt http-kutsut kulkevat sen läpi. Tämä tarkoittaa, että sekä lähtevä kutsu, että sen paluuarvo kulkevat interceptorin kautta (kuvio 19).



KUVIO 19. Interceptorin toiminta (Angular in Depth 2019)

Interceptorit pystyvät muokkaamaan kutsuja, esimerkiksi lisäämään tunnistustietoja (kuvio 20) tai muita header parametreja lähteviin kutsuihin (Angular 2019). Myös esimerkiksi koko lähtevän kutsun tietosisältöä voidaan tarvittaessa käsitellä halutulla tavalla interceptorissa ennen sen lähettämistä backendiin.

```
export class AuthHttpInterceptor implements HttpInterceptor {
  intercept(
    req: HttpRequest<any>,
    next: HttpHandler,
  ): Observable<HttpEvent<any>> {
    const token = this.fetchToken();
    const newRequest = req.clone({
      headers: req.headers.set('Authorization', 'Bearer ' + token),
    });
    return next.handle(newRequest);
  }
```

KUVIO 20. Interceptor

Kaikki interceptorin sisältävästä moduulista lähtevät sovelluksen kutsut kulkevat sen kautta. Tämä mahdollistaa käytännössä koko sovelluksen http-kutsujen keskitetyn käsittelyn, ja tästä syystä esimerkiksi juurikin tunnisteen lisääminen jokaiseen kutsuun voidaan hoitaa yhdessä paikassa. Koska myös paluuarvot kulkevat interceptoreiden kautta, voidaan interceptoreita käyttää myös paluuarvon muokkaamiseen tai esimerkiksi keskitettyyn virheenkäsittelyyn.

Tähän asti olen hoitanut virheenkäsittelyn erikseen jokaiselle kutsulle, ilman yhtenäistä virheenkäsittelymenetelmää kaikille kutsuille. Tämä juontaa juurensa siihen, että alussa, kun tarvittavia kutsuja ei vielä ollut niin paljon, oli virheenkäsittely helppo hoitaa yksittäin, eikä kunnollisen yhtenäisen virheenkäsittelijän tekemiseen ollut tarvetta. Nyt kun kutsujen määrä ja sovelluksen koko on kasvanut roimasti, on virheenkäsittelyn keskittäminen yhteen paikkaan järkevää.

3.10 Kymmenes viikko

Maanantai

Jälleen kerran viikko alkoi maanantaipalavareilla. Kerroin viimeviikolla pääasiassa työskennelleeni API-muutosten parissa, sekä myös jatkavani niiden parissa yhteistyössä kollegani kanssa. Tällä viikolla olisi tulossa myös mobiilisovelluksen uuden version julkaisu, joten tulisin auttamaan myös sen testauksessa tarvittaessa ja mahdollisuuksien mukaan.

Ennen kuin ehdin aloittaa suunnitelmani mukaisia tehtäviäni, sain tiedon, että sovelluksen vähemmän käytetyssä ominaisuudessa, jota korjailin jo viime viikolla, oli vieläkin ongelmia. Päiväni tavoitteeksi tulikin siis näiden ongelmien korjaaminen. Ongelmien korjaamisessa ei sinänsä ollut kovin paljoa työtä, mutta silti ne vaativat jonkin verran pohtimista. Nämä ongelmat johtuivat oikeastaan siitä, että vaikka ominaisuus on ollut sovelluksessa

jo jonkin aikaa, on se ollut vähäisellä käytöllä. Kaikkia käyttötapauksia ja ominaisuuden selkeyttä käyttäjän kannalta ei ehkä ollut ehditty hioa loppuun asti. Sain kuitenkin yksinkertaisilla muutoksilla parannettua ominaisuuden käytettävyyttä huomattavasti. Päätin kuitenkin tuoda ensi sprinttipalaverissa esille, että mikäli ominaisuus nyt tulee olemaan laajemmassa käytössä, täytyy sitä ehkä miettiä vielä uudestaan koko järjestelmän kannalta.

Tiistai

Tiistaina aloitin vielä testaamalla eilen tekemiäni korjauksia ja varmistin kollegaltani, että voin julkaista korjauksen myös vaaditulla tuotantopalvelimella. Tämän jälkeen testasin nopeasti vielä kyseisellä palvelimella, että korjaukset toimivat kuten pitääkin.

Kun olin saanut kyseisen korjauksen valmiiksi, ryhdyin työstämään jälleen API-muutoksia. Kollegani oli viime viikkoisen keskustelumme jälkeen saanut uusia muutoksia valmiiksi, joten minun tuli saada myös web-sovellus yhteensopivaksi niiden kanssa. En kuitenkaan päässyt työssä kovin pitkälle, sillä apuani vaadittiin mobiilisovelluksen testauksessa.

Testasin sovelluksen uutta ominaisuutta kahdella eri ominaisuudella ja löysin lähes heti ongelman, josta raportoin kollegalleni. Löytö oli hyvä, sillä myös hän oli havainnut saman ongelman, muttei ollut yrityksistään huolimatta saanut sitä toistettua. Kävimme yhdessä läpi, miten bugi ilmeni ja saimme sen myös toistettua, joten pystyimme ryhtyä paikantamaan, mistä se johtuu. Ongelman syyksi paljastui lopulta uuden ominaisuuden kannalta viallinen SQL kyselylause backendissä, jonka backend kehittäjä lupasi korjata huomiseksi.

Alkuviikko on jälleen ollut osoitus pienessä startup-yrityksessä työskentelyn hektisyydestä. Tilanteet vaihtuvat nopeasti ja apuani on tarvittu monessa paikassa. Tekemistä on paljon ja monella saralla, joka on välillä hieman stressaavaa. Pidän tätä kuitenkin arvokkaana kokemuksena tulevaisuuteni kannalta alalla, sillä ympäristö on opettanut runsaasti paineensietokykyä ja olen päässyt oppimaan usealla eri alueella.

Keskiviikko

Keskiviikkona aloitin päiväni testaamalla eilistä ominaisuutta, sillä backend kehittäjä kertoi saaneensa ongelman korjattua. Ongelma näyttikin nyt olevan korjattu ja ominaisuus toimi, kuten pitääkin.

Tämän jälkeen ryhdyin taas työstämään eilen kesken jääneitä API-muutoksia web-sovellukseen. Samalla tarjoutui myös hyvä tilaisuus kuolleen koodin karsimiseen, sekä joidenkin rakenteiden selkeyttämiseen. Jälleen jouduin kuitenkin kohdentamaan huomioni muualle, sillä ulkomaalaisen asiakkaan luona asennuksia tekevät kollegani ilmoittivat löytäneensä sovelluksesta ongelman, joka häiritsi asennuksia suuresti. En saanut ongelmaa

toistettua millään muulla palvelimella, joten ongelman syyn selvittäminen osoittautui hieman hankalaksi. Kävin viallisen ominaisuuden toimintaa läpi rivi riviltä ja sain lopulta paikannettua ongelman syyn, joka loppujen lopuksi olikin yksinkertainen, eikä edes vaatinut minulta enempää toimenpiteitä. Kerroin ongelman löytäneille kollegoilleni sen syyn, jotta he pystyisivät korjaamaan vian omassa päässään. Pian he raportoivat kaiken toimivan nyt oikein. Merkitsin kuitenkin muistiin, että asia täytyy ottaa esille vielä myöhemmin, sillä sama ongelma voi toistua myös tulevaisuudessa ja on helppo varmistaa, ettei niin käy.

Päivän aikana sattunut tilanne oli hyvä oppimiskokemus. Kun kollegani ilmoittivat ongelmasta, stressaannuin hieman, sillä se vaikutti vakavalta, enkä osannut suoraan sanoa ollenkaan, mistä se voisi johtua. Kollegani olivat myös tekemässä asennuksia, ja halusin ratkaista ongelman mahdollisimman nopeasti, jotta he voivat jatkaa työtään. Ongelman ratkaiseminen kohotti luottamustani omiin kykyihini ammattilaisena, sekä toimi arvokkaana muistutuksena siitä, että ominaisuuksia tehdessä täytyy varautua myös kaikkein epätodennäköisimpiin tilanteisiin ja varmistaa, että myös ne on huomioitu, eivätkä toteutuessaan aiheuta ongelmia.

Torstai

Aloitin torstain edelleen API-muutoksien parissa, joten päiväni tavoitteeksi otin niiden tekemisen, sekä samalla mahdollisen kuolleen ja turhan koodin poistamisen. Samalla huomasin, että saisin yksinkertaistettua erään ominaisuuden toimintaa, mikäli kollegani lisäisi siihen liittyvän API:n palauttamaan listaan yhden parametrin lisää. Nyt kun API muutoksia muutenkin tehtiin, näin tämän loistavaksi tilaisuudeksi pyytää kyseistä muutosta.

Ilmoitin kollegalleni haluamastani muutoksesta ja hän totesi sen olevan helposti toteutettavissa. Niinpä ryhdyin toteuttamaan ominaisuuden muutosta yksinkertaisempaan suuntaan. Sain karsittua kyseisen toiminnon koodin määrän alle puoleen entisestä, ja samalla myös suoraviivaistettua ja selkeytettyä sitä. Tämä osaltaan vähentää ennalta arvaamattomien bugien syntyä, sekä helpottaa paikantamista, mikäli bugeja esiintyy.

Kokemukseni karttuessa huomaan, etteivät jotkin aikaisemmin tekemäni ratkaisut ole parhaita mahdollisia ja olen pystynyt nyt jälkeenpäin läpi käydessä muokkaamaan niitä paremmiksi. API-muutosten tekeminen onkin osoittautunut loistavaksi tilaisuudeksi myös tähän, sillä muutokset itsessään eivät ole kovin suuria, mutta vaativat, että käyn läpi suuren määrän koodia. Huomaan, että kokemukseni karttuessa pystyn tekemään koko ajan yhä toimivampia ja tehokkaampia ratkaisuja eri ongelmiin ja aikaisemmin tekemäni ratkaisut ovat välillä hyvinkin monimutkaisia ja jälkeenpäin hankalasti luettavia.

Perjantai

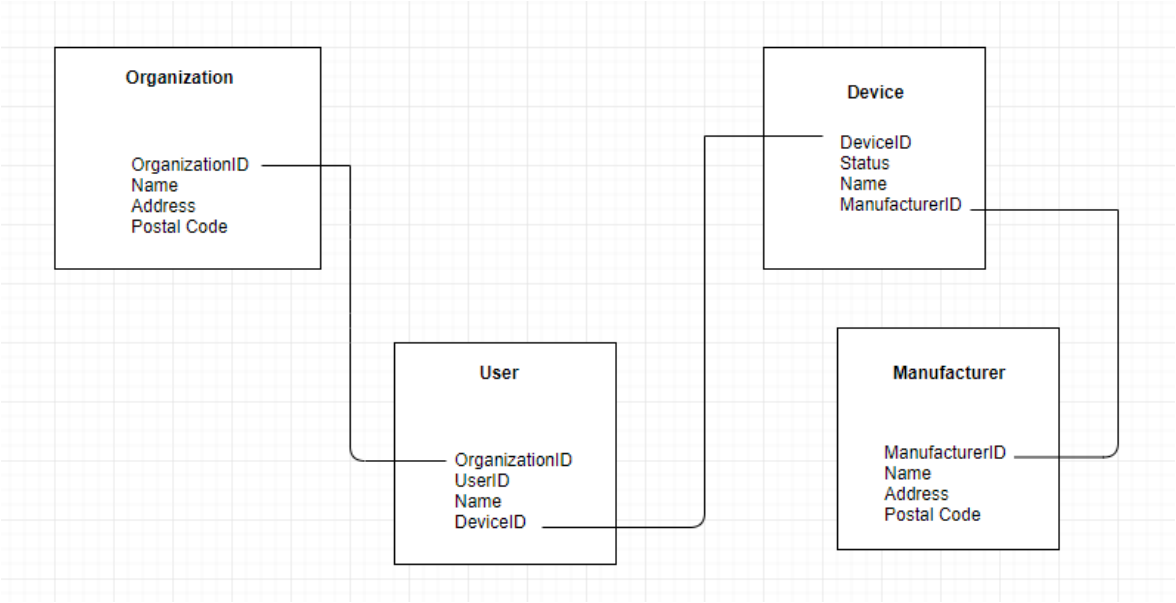
Kymmenennen seurantaviikon viimeinen päivä oli myös meneillä olevan sprintin viimeinen päivä. Olin saanut tehtäväni Angular 7 päivityksen, sekä siihen liittyvien isompien tehtävien osalta hoidettua ja API-tehtävätkin olivat hyvällä mallilla. Omalla sprinttitaulullani oli kuitenkin vielä yksi pieni tehtävä, jonka saisin varmasti vielä tänään tehtyä. Otinkin päiväni tavoitteeksi siis tämän toteuttamisen.

Tarvittavalle ominaisuudelle ei ollut vielä muuta backend tukea, kuin sitä varten tehty esimerkki API, joka ei käytännössä tehnyt mitään muuta kuin tarkisti, että lähetetty arvo on oikeassa muodossa. Pystyin siis lisäämään ominaisuuden jo sovellukseen siltä pohjalta, miten sen tulee toimia, mutta sen todellinen testaaminen ja toteuttamisen pitäisi hoitaa myöhemmin.

Viikkoanalyysi

Kymmenes viikko kului suurelta osin API-muutoksiin liittyvien tehtävien parissa. Useimpien API:en käyttötarkoitus on lopulta lisätä, päivittää, poistaa tai lukea jotain yrityksen tietokannasta, joten olen näiden tehtävien myötä ollut tekemisissä myös tietokantoihin liittyvien teknologioiden kanssa.

Tietokantaratkaisuna yrityksessä käytetään PostgreSQL tietokannan hallintajärjestelmää. PostegreSQL on vapaan lähdekoodin relaatiomalliin perustuva tietokantajärjestelmä. Tämä tarkoittaa, että tietokantoihin tallennettava data esitetään listoina, jotka on relaatioiden mukaan jaettu tauluiksi (kuvio 21). Tauluissa jokaista tallennettua kohdetta kuvaa rivi, mikä tarkoittaa, että esimerkiksi yhden käyttäjän tiedot vastaavat yhtä riviä tietokannan taulussa. Jokainen taulun rivi on puolestaan jaettu sarakkeisiin kohteen attribuuttien mukaisesti. Sarakkeita on taulun jokaisella rivillä sama määrä ja ne vastaavat samoja attribuutteja, mikä tarkoittaa, että kaikki tauluun tallennetut kohteet ovat attribuuttiensa tietotyypeiltä identtisiä. Taulut taas kootaan varsinaisiin tietokantoihin ja tietokantojen kokoelma, jota yksittäinen PostgreSQL palvelininstanssi hallitsee, koostuu ryhmästä tällaisia tietokantoja. PostgreSQL tukee suurta osaa SQL standardista. (PostgreSQL 2019.)



KUVIO 21. Relaatiotietokannan tauluja

PostgreSQL käyttää asiakas/palvelin mallia, mikä tarkoittaa, että tietokantojen hallintaa hoitava sessio koostuu kahdesta prosessista; palvelinprosessista, joka hoitaa varsinaisen tietokantojen käsittelyn ja hallinnan, sekä asiakassovelluksesta, joka tarvittaessa pyytää palvelinprosessia esimerkiksi muuttamaan tai lukemaan tietoja tietokannoista (PostgreSQL 2019). Esimerkiksi ratkaisut, joita työssäni olen kohdannut, ovat koostuneet palvelimella olevasta PostgreSQL palvelinsessiosta, sekä Pythonilla koodatusta backendistä, joka toimii tarvittaessa myös asiakassovelluksena. Käytännössä siis esimerkiksi tekemäni web-sovellus tekee API-kutsun Python backendiin rajapinnan kautta, joka puolestaan tekee pyynnön PostgreSQL palvelinsessiolle. Palvelinsessio tekee varsinaisen tietokantakyselyn ja palauttaa vastauksen Python backendille, joka puolestaan palauttaa sen JSON-muotoisena paluuarvona web-sovelluksesta tehdylle kutsulle. Näin tietokantoja voidaan lukea ja muokata aina web-sovelluksesta asti, mikä mahdollistaa esimerkiksi käyttäjien hallinnan selkeästä käyttöliittymästä.

Muihin avoimen lähdekoodin SQL-tietokantoihin verrattuna PostgreSQL:n etuna on etenkin se, että se ei ole pelkästään tavallinen relaatiokanta, vaan se käyttää olio-relaatiota. Olio-relaation perustavanlaatuinen ero on se, että se tukee käyttäjän määrittelemiä olioita ja ominaisuuksia aina datatyypeistä funktioihin ja indekseihin. Tämä tekee PostgreSQL:sta hyvin joustavan ja vakaan. (PostgreSQL 2019.) Olio-relaatio mahdollistaa myös monimutkaisten datarakenteiden, esimerkiksi taulukkojen ja moniuloitteisten tauluk-

kojen, tallentamisen (kuvio 22). Muissa yleisesti käytetyissä avoimen lähdekoodin tietokantaratkaisuissa täytyy taulukkojen tallentamiseen keksiä vaihtoehtoisia menetelmiä. (PostgreSQL 2019.)

```
CREATE TABLE sandwich_menu (
                name varchar(50),
                sandwich text[],
                side text[] [], --moniuloitteinen lista
        );

INSER INTO sandwich_menu VALUES (
                'the menu',
                '{'beef', 'cheese', 'ham'},
                '{
                   {'fries', 'soda', 'bread sticks'},
                   {'carrots', 'milk', 'cherry tomatoes'}
                 }'
        );
```

KUVIO 22. Taulukkojen tallentaminen PostgreSQL tietokantaan

Tietokantojen kanssa työskentely on mielenkiintoista vaihtelua, jota olen rajapintamuutoksiin liittyviä tehtäviäni suorittaessani päässyt tekemään. Koen rajapintojen yhtenäistämisprojektin olleen oman kehitykseni kannalta hyväksi, sillä olen sen myötä päässyt työskentelemään käytännössä koko yrityksen ohjelmistokokonaisuuden parissa ja oppinut ymmärtämään sen eri osien toimintaa paremmin. Sen myötä koen kehittyneeni ohjelmistokehittäjänä kokonaisvaltaisemmin, kuin pelkästään frontendin parissa työskennellessäni.

4 POHDINTA

Tämän päiväkirjamallisen opinnäytetyön tavoitteena on ollut oman työskentelyn seuraaminen ja sitä myötä myös henkilökohtaisen oppimisen ja kehittymisen dokumentoiminen. Myös asioiden ja teknologioiden tarkempi tarkastelu viikoittaisissa analyyseissa lähteitä hyväksi käyttäen on merkittävä osa opinnäytetyön toteutusta.

Seurantajakson aikainen kehittymiseni ammattilaisena ohjelmistokehittäjän työssä on ollut huomattavaa. Vaikka olen työskennellyt yrityksessä samoissa tehtävissä jonkin aikaa jo ennen seurantajakson toteuttamista, olen alalla kuitenkin vielä kovin nuori, jolloin huomattavaa kehitystä tapahtuu koko ajan. Koen myös oppivani uutta lähes joka päivä. Osaltaan päiväkirjamallisen opinnäytetyön kirjoittaminen on myös pakottanut kiinnittämään huomiota henkilökohtaiseen kehitykseen ja oppimiseen.

Seurantajakson aikaiset työtehtäväni koostuivat suurelta osin Angular- ja React-pohjaisten sovellusten kehittämis- ja korjaustöistä, mutta seurantaviikkojen aikana eteen tuli myös yllättävämpiä työtehtäviä aina käyttäjämanuaalien tekemisestä erilaisiin tietokanta- ja backend-tehtäviin. Varsinaiset työtehtävät olivat kuitenkin pitkälti samaa, mitä olen tottunut yrityksessä tekemään jo ennen opinnäytetyön seurantajakson aloittamista. Päivittäiset työtehtävät ovat suurelta osin koostuneet puhtaasti ohjelmistokoodin kirjoittamisesta ja kohtaamieni ongelmien ratkaisemisesta. Myös vuorovaikutustilanteet kollegoideni kanssa erilaisia ominaisuuksia ja ratkaisuja suunniteltaessa ja toteutettaessa ovat olleet merkittävässä osassa päivittäistä työskentelyäni seurantajakson aikana. Olen pyrkinyt sisällyttämään työpäiviini myös henkilökohtaisen osaamiseni kartuttamista alan blogeja seuraamalla, erilaisia ohjelmistoalaan liittyviä julkaisuja lukemalla, sekä käyttämieni teknologioiden dokumentaatioita läpi käymällä ja opiskelemalla.

Seurantajakson aikaiset työtehtäväni on dokumentoitu päivätasolla. Kehittymistäni, uusien asioiden oppimista, sekä kohtaamieni ongelmieni ratkaisuja on syvemmin tarkasteltu ja analysoitu viikkokohtaisissa analyyseissa. Nämä viikkoanalyysit ovat seurantajakson aikana käsitelleet asioita aina yksittäisistä JavaScript kirjastoista ja teknologiallisista oppimistilanteista aina kollegoiden välisiin ryhmätyöskentelytilanteisiin ja vuorovaikutustaitojeni kehityksen pohtimiseen. Olen viikkoanalyyseja kirjoittaessani pyrkinyt löytämään viikon aikaisista työtehtävistäni jonkinlaisen yhdistävän teeman, vaikka viikot välillä ovatkin sisältäneet sisällöltään rikkonaisesti työtehtäviä. Opinnäytetyön viikkoanalyysejä on lähdetty rakentamaan näiden viikoittaisten teemojen pohjalta erilaisia lähteitä hyödyksi käyttäen. Näin jokaiselta viikolta on löytynyt uutta opittavaa, vaikka työtehtävät viikon aikana eivät sinänsä olisi juurikaan poikenneet mistään aikaisemmin kohtaamastani.

Juuri nämä viikoittaiset analyysit ovat tarjonneet mahdollisuuden perehtyä ja palata vielä tarkemmin viikon tapahtumiin sekä analysoida tapahtumia ja tehtäviä yksityiskohtaisemmin. Osaltaan tämä on johtanut uusien asioiden oivaltamiseen ja seurantajakson aikana oppimieni taitojen parempaan sisäistämiseen.

Kun vertaan seurantajakson lähtötilannetta nykyiseen, huomaan olevani erilaisissa työhön liittyvissä vuorovaikutustilanteissa itsevarmempi ja pystyväni tuomaan oman näkemykseni ja asiantuntemukseni paremmin esille. Näin pystyn paremmin toimimaan osana tiimiä ja koen, että juuri tämän kaltaisesta kehityksestä on suuresti hyötyä niin omalle henkilökohtaiselle työuralleni, kuin työnantajallenikin. Koen oman kehitykseni vaikuttaneen suoraan myös koko tiimin tehokkuuteen, sillä pystyn yhä enenevissä määrin tarjoamaan omia näkökulmiani ja osaamistani asioiden ratkaisemiseksi. En enää vain kysy apua muilta, vaan pystyn myös tarjoamaan omaa apuani muiden sitä tarvitessa.

Huomaan seurantajakson aikaisen kehitykseni myös omia päivittäisiä työtehtäviäni suorittaessa. Olen koko ajan valmiimpi ratkaisemaan työssä kohtaamiani ongelmia, sekä kokemukseni karttuessa huomaan, että todella hankalan tuntuisia ongelmia tulee vastaan jatkuvasti vähemmän, kuin aikaisemmin. Pystyn päivittäisessä työssäni peilaamaan kohtaamiani tilanteita aikaisempiin kokemuksiini ja selviytymään vastaan tulevista tilanteista näin huomattavasti tehokkaammin. Myös seurantajakson aikana kertynyt kokemus koko ohjelmistorakenteesta kokonaisuutena auttaa minua tekemään yhä parempia ratkaisuja päivittäisessä työssäni, jota silti edelleen teen lähinnä selainpohjaisten sovellusten parissa, frontend-kehittäjänä.

Opinnäytetyötä kirjoittaessani, olen oppinut paremmin käyttämään hyödyksi erilaisia lähteitä kohtaamieni ongelmien ratkaisuksi, sekä henkilökohtaisen osaamiseni syventämiseksi. Juuri yllä mainittu pyrkimys seurata ohjelmistoalan julkaisuja oman henkilökohtaisen osaamiseni kehittämiseksi onkin alkanut juuri seurantajakson aikana viikkoanalyysejä varten lähteitä etsiessäni. Olen huomannut tämän olleen myös merkittävä tekijä oman asiantuntijuuteni kehittämiseksi ja tärkeä tapa alati kehittyvän alan mukana pysymiseksi.

Kaiken kaikkiaan koen päiväkirjaopinnäytetyön kirjoittamisen hyödyttäneen kehittymistäni aloittelevana ohjelmistokehittäjänä jonkin verran. Pidänkin päiväkirjamallista opinnäytetyönä hyvänä keinona seurata omaa työskentelyä uran alkuvaiheessa, jolloin päiväkirjamallisen opinnäytetyön kirjoittaja pystyy itse oppimaan ja kehittämään itseään omaa työskentelyään seuraamalla. Tämä auttaa huomaamaan erilaisia virheitä, tapoja tai pullonkauloja omassa työssä ja oppimaan niistä.

LÄHTEET

Angular 2019. Angular Docs. Google [viitattu 28.3.2019]. Saatavissa: https://angular.io/docs

Git 2019. Git Documentation. Git [viitattu 3.11.2019] Saatavissa: https://git-scm.com/doc

Homes, B. 2013. Fundamentals of Software Testing. Wiley 2013 [viitattu 28.3.2019]. Saatavissa: https://ebookcentral-proquest-com.aineistot.lamk.fi/lib/lamk-ebooks/detail.action?docID=1120766

Jyväskylän yliopisto 2019. Testauksen tasot. [viitattu 3.11.2019]. Saatavissa: http://smarteducation.jyu.fi/projektit/systech/Periaatteet/suunnittelun-periaatteet/testaus/testauksen-tasot

Karén, M. 2019. 10 Ways to use Interceptors in Angular. Angular in Depth [viitattu 7.11.2019]. Saatavissa: https://blog.angularindepth.com/top-10-ways-to-use-interceptors-in-angular-db450f8a62d6

MDN Web Docs 2019. CSS: Cascading Style Sheets [viitattu 2.11.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/CSS

Nayrolles, M., Hamou-Lhadj, A., Tahar, S., Larsson, A. 2015. JCHARMING: A bug reproduction approach using crash traces and directed model checking. ProQuest LLC [viitattu 4.4.2019]. Saatavissa: https://search-proquest-com.aineistot.lamk.fi/docview/1709723559/

Liang, C., Malatpure, A., Shafiei M., Vago M., Zheng T. 2012. Customer Scenario Focused End to End Testing. IEEE. [viitattu 5.4.2019]. Saatavissa: https://ieeexplore-ieee-org.aineistot.lamk.fi/document/6405421/authors#authors

PostgreSQL 2019. PostgreSQL Documentation. The PostgreSQL Global Development Group [viitattu 8.11.2019]. Saatavissa: https://www.postgresql.org/docs/

RESTfulAPI.net 2019. How to design a REST API. RESTfulAPI.net [viitattu 8.4.2019]. Saatavissa: https://restfulapi.net/

RxJS 2019. RxJS Docs. [viitattu 8.4.2019]. Saatavissa: https://rxjs-dev.firebaseapp.com/guide/

Seobility 2019. REST API. [viitattu 5.11.2019]. Saatavissa: https://www.seobility.net/en/wiki/REST_API

Software Testing Fundamentals 2019. Software Testing Fundamentals. [viitattu 1.11.2019]. Saatavissa: http://softwaretestingfundamentals.com/

Tutorials Point 2019. SDLC – V-Model. [Viitattu 21.11.2019]. Saatavissa: https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm

Vanderjack, B. 2015. The Agile Edge: Managing Projects Effectively Using Agile Scrum. Business Expert Press [viitattu 28.3.2019]. Saatavissa: https://ebookcentral-proquest-com.aineistot.lamk.fi/lib/lamk-ebooks/detail.action?docID=2145193

W3Schools 2019. HTML tutorial. [viitattu 1.11.2019]. Saatavissa: https://www.w3schools.com/html/default.asp