Junior IT-konsultti asiakkaan projektissa

Sini Pajunen

Haaga-Helia ammattikorkeakoulu

Tradenomi

Opinnäytetyö

2023

Tiivistelmä

Tekijä(t)
Sini Pajunen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Junior IT-konsultti asiakkaan projektissa

Sivu- ja liitesivumäärä
48

Päiväkirjamuotoisessa opinnäytetyössä seurataan junior IT-konsultin työtä asiakkaan projektissa. Asiakkaana toimii suuri Suomalainen pörssiyritys ja työtehtävät ovat full stack -sovelluskehitystä. Opinnäytetyö alkaa johdannolla, jota seuraa lähtötilanteen kartoitus. Päiväkirjan merkintöjä on tehty kahdeksan työviikon ajalta syksyllä 2022. Jokaiselle seurantajakson viikolle on määritelty tavoitteet, joiden täyttymistä arvioidaan viikkoanalyyseissä.

Opinnäytetyö käsittelee kirjoittajan ammatillista kehitystä ennalta määriteltyjen oppimistavoitteiden puitteissa. Ensimmäinen oppimistavoite on sovelluskehittäjänä kehittyminen, joka pitää sisällään frontend-kehittäjän taidot, luettavan ja laadukkaan koodin tuottamisen ja kehitystiimin jäsenenä toimimisen. Toisena oppimistavoitteena on ketterän kehityksen viitekehyksen ymmärtäminen ja tiimin käyttämien työkalujen hyödyntäminen. Opinnäytetyössä käsitellään ketterään kehitykseen liittyen SAFe-viitekehystä ja Kanban-projektinhallintamenetelmää. Työvälineistä käsitellään erityisesti Git-versionhallintaa ja Jira-tiketinhallintaa.

Opinnäytetyön lopussa on pohdinta, jossa analysoidaan ja arvioidaan työlle asetettujen oppimistavoitteiden täyttymistä. Pohdinnan lopputuloksena todetaan kehittymistä tapahtuneen kaikilla tavoitteiksi asetetuilla osa-alueilla. Opinnäytetyö antaa käsityksen uran alkuvaiheessa olevan kehittäjän työtehtävistä ja vaatimustasosta, sekä konsulttina toimimisen haasteista.

Asiasanat

Ohjelmistokehitys, ammatillinen kehitys, ketterät menetelmät

Sisällys

Käsitteet

Kehitystiimi	Koostuu joukosta sovellustiimejä ja niitä tukevia rooleja, kuten testaajia.
Sovellustiimi	Sovelluskohtainen tiimi, koostuu kehittäjistä ja tuote-omistajasta.
QA, Quality Assurance Engineer	Sovelluksen testaaja
PO, Product Owner	Tuoteomistaja, on vastuussa sovelluksen kehityksestä.
DM, Development Manager	Kehityspäällikkö, on vastuussa, että sovellustiimi ymmärtää liiketoiminnan vaatimukset ja toiveet.
BO, Business Owner	Liiketoiminnan edustaja, antaa määritykset uusille kehityksille ja priorisoi ne. On vastuussa, että sovellustiimi tuottaa liiketoimintahyötyä.
SAFe, Scaled Agile Framework	Ketterän kehittämisen viitekehys
PI, Program Increment	Program Increment (PI) on SAFe:en kuuluva 8–12 viikon mittainen inkrementti eli ajanjakso, jonka aikana pyritään toimittamaan sovellukseen haluttuja toiminnallisuuksia.
PI-suunnittelutapahtuma	PI alkaa inkrementin suunnittelutapahtumalla, jossa määritellään tulevan PI:n tavoitteet.
Kanban	Ketterän projektinhallinnan menetelmä ja -työkalu, jonka periaatteena on visualisoida työtehtävät ja niiden eteneminen Kanban-taululle.
Kanban-taulu	Kanban-taulun avulla seurataan työ etenemistä ja läpimenoaikaa, sekä rajoitetaan keskeneräisen työn määrää.
Daily, The Daily Stand-Up	Päivittäinen sovellustiimin tapaaminen, jossa kehittäjät käyvät läpi työn alla olevia töitään ja mahdollisia esteitä niiden edistämiselle.

PR, Pull request	GitHub-versionhallintaan avattava muutospyyntö, jonka hyväksymisen jälkeen kehitetty toiminnallisuus yhdistetään sovelluksen koodiin.

1 Johdanto

IT-alan osaajapulasta puhutaan paljon. Joidenkin arvioiden mukaan alalle voitaisiin palkata noin 20 000 uutta osaajaa vuosittain, mutta korkeakouluista valmistuu vuosittain alle 5000 (Ussa 22.9.2022). Lisäksi ongelmana ei ole vain uusien tekijöiden saaminen alalle, vaan yritykset haluavat nimenomaan kokeneita osaajia (Remes 12.10.2021). Jokainen senior on kuitenkin ollut joskus junior ja kokemusta kertyy töitä tekemällä. Tämä päiväkirjaopinnäytetyö kertoo junior konsultin työstä kehittäjänä, päivittäisestä ammatillisesta kehityksestä sekä kohdatuista haasteista ja toisaalta myös aiemman uran aikana kerrytetyn kokemuksen tuomista hyödyistä.

Opinnäytetyöprojekti sijoittuu ajankohtaan, jossa olen työskennellyt IT-alalla noin puolitoista vuotta. Olen siirtynyt työelämässä toiselle toimialalle ja opiskellut sitä varten IT-alan tutkintoa ammattikorkeakoulussa. Lisäksi olen suorittanut verkkokursseja itseopiskellen ja käynyt Academic Workin Academy-intensiiviopinnot, jonka kautta pääsin heti tekemään kehittäjän töitä.

1.1 Opinnäytetyön tavoitteet

Opinnäytetyö käsittelee junior full stack -kehittäjän laajaa tehtävää suuressa Suomalaisessa pörssiyrityksessä. Yrityksellä on liiketoimintaa kolmella eri toimialalla. Työskentelen tiimissä, joka keskittyy kehittämään myyntiä tukevia sovelluksia rakentamisen ja talotekniikan kaupalle. Kauppa on suunnattu B2B asiakkaille ja toimii useassa maassa.

Opinnäytetyön päiväkirjamerkintöjä on kerätty kahdeksan työviikon ajalta ja ne sijoittuvat ajanjaksoille 5.9.–9.9.2022 ja 19.9.–4.11.2022. Päiväkirjamerkintöjä seuraa viikoittainen analyysi pohjautuen viikon aikana opittuihin asioihin. Opinnäytetyön tarkoitus on seurata kehittymistä seuraavissa oppimistavoitteissa:

1. Sovelluskehittäjänä kehittyminen
    a. Frontend-kehittäjän taidot (React, TypeScript, Redux, CSS)
    b. Luettavan ja laadukkaan koodin tuottaminen
    c. Kehitystiimin jäsenenä toimiminen (koodikatselmoinnit, yhteiset päivystys- ja valvontatehtävät)
2. Ketterän kehityksen viitekehyksen ymmärtäminen ja tiimin käyttämien työkalujen hyödyntäminen
    a. Ketterä kehitys: SAFe, Kanban
    b. Työvälineet: Git-versionhallinta, Jira-tiketinhallinta

Oppimistavoitteet on valittu tunnistettujen kehittymistä vaativien taitojen ja omien mielenkiintojen perusteella. Edellä luetelluissa tavoitteissa kehittyminen edesauttaa seuraavan tason saavuttamisen uralla.

1.2 Alan kirjallisuus ja tietolähteet

Kehittäjän työssä suurin osa tietolähteistä löytyy netistä erilaisten dokumentaatioiden ja tutoriaalien muodossa. Myös kokeneempien kehittäjien vertaisvastaukset erilaisilla keskustelupalstoilla auttavat usein ongelmissa. Laadukkaan ja luettavan koodin kirjoittamista olen opiskellut muun muassa Robert C. Martinin kirjoittaman teoksen Clean Code avulla. Martin on myös yksi ketterän ohjelmistokehityksen julistuksen kirjoittajista. Päivittäin käyttämiäni tietolähteitä ovat:

- Eri ohjelmointikielten dokumentaatiot, koulutusmateriaalit ja ohjekirjat
- Blogit ja tutoriaalit, esimerkiksi Medium, Baeldung
- Kehittäjien kysymyspalstat, esimerkiksi Stack Overflow

2 Lähtötilanteen kuvaus

Toimin junior IT-konsulttina suuressa Suomalaisessa pörssiyrityksessä junior full stack -kehittäjä työtehtävässä. Toimeksianto on konsulttiyrityksen kautta, jossa olen työskennellyt noin puolitoista vuotta. Opinnäytetyöprojekti liittyy nykyiseen toimeksiantoon, jossa olen opinnäytetyöprojektin alkaessa työskennellyt noin seitsemän kuukautta.

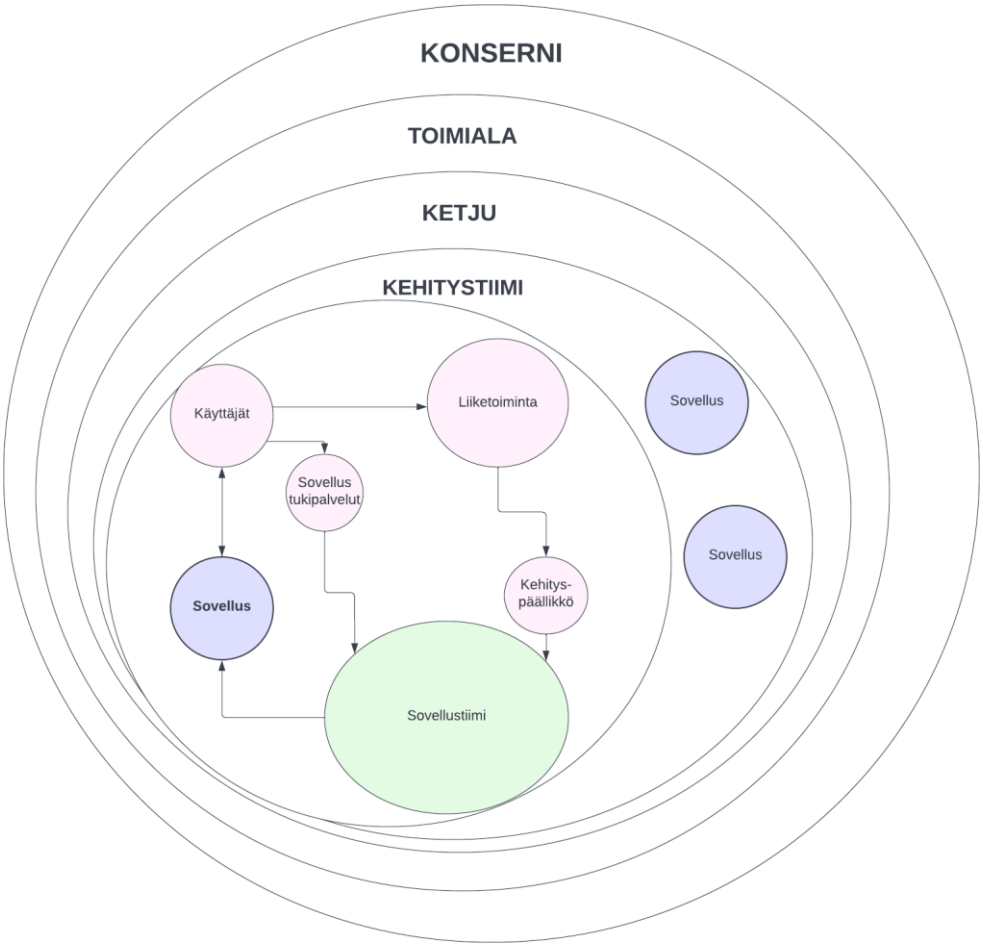
Aiemmin työskentelin kaupan alan tehtävissä ja vaihdoin uran suuntaa it-alalle muutama vuosi sitten. Ensimmäisessä IT-alan toimeksiannossa tein vuoden ajan backend-kehitystä. Toiveenani oli päästä kehittymään myös frontend-kehittäjänä, jonka nykyinen toimeksianto ja full stack -kehittäjän työtehtävät mahdollistavat. Yksi opinnäytetyöprojektille asettamistani oppimistavoitteista on frontend-kehittäjän taitojen syventäminen modernien teknologioiden, kuten Reactin ja TypeScriptin osalta. Kehittyminen auttaa myös ammatillisessa tavoitteessani hakeutua lähitulevaisuudessa frontend-kehittäjäksi.

2.1 Oman nykyisen työ analysointi

Yritys ja tiimin rakenne

Yrityksellä on liiketoimintaa päivittäistavarakaupan, autokaupan sekä rakentamisen ja talotekniikan kaupan toimialoilla. Rakentamisen ja talotekniikan kauppa jakautuu eri ketjuihin, joille yritys tuottaa myyntiä tukevia digitaalisia palveluita talon sisäisinä projekteina. Ketjun sisällä on yksi iso kehitystiimi, joka jakautuu pienempiin eri sovellusten kehittämiseen keskittyviin sovellustiimeihin. (Kuva 1, 6.)

Tiimin rakenne ja sidosryhmät on havainnollistettu seuraavassa kuvassa 1.



Kuva 1. Tiimin rakenne ja sidosryhmät

Ketjun sisäisessä kehitystiimissä (kuva 1) työskentelee noin parikymmentä henkilöä ja siihen kuuluu:

- Sovelluskehittäjät
- Designerit
- Testaajat (QA)
- SAP-kehittäjät
- Tuoteomistajat (PO)
- Kehityspäälliköt (DM)
- Hallinnosta ja julkaisuista vastaavat

Sovelluskohtaiset tiimit koostuvat sovelluskehittäjistä ja niille on nimetty oma tuoteomistaja ja kehityspäällikkö. Designerit, testaajat ja SAP-kehittäjät tukevat kaikkia sovellustiimejä tarpeen mukaan. Kehitystiimin jäseninä on sekä yrityksen omia työntekijöitä, että alihankkijoilta ostettuja konsultteja.

Työtehtävät

Sovellustiimissä työskentelee kanssani kaksi full stack -kehittäjää ja tuoteomistaja. Sovellustiimin muut kehittäjät ovat kokeneempia kuin minä, joten he mentoroivat ja auttavat minua aina tarvittaessa. Sovellustiimi työskentelee pääosin yrityksen pääkampuksella, joka on helpottanut kommunikaatioyhteyden muodostamista ja avun kysymistä (Tolvanen 30.3.2021).

Sovellustiimin kehittäjät ovat full stack -kehittäjiä, joten jokaisella on valmiudet tehdä kaikkia kehitysjonossa olevia frontend- ja backend-tikettejä. Usein tiketit kuitenkin jaetaan yhteisymmärryksessä kunkin kehittäjän mielenkiintojen ja aiemmissa työtehtävissä kerrytetyn kokemuksen mukaan. Työtehtäviin sovellustiimissä sisältyvät ensisijaisesti sovelluksen jatkokehittäminen ja ylläpito sekä ilmenneiden virheiden korjaus. Sovelluskehitykseen liittyen osallistun tiimin palavereihin, joissa muun muassa määritellään uusien toiminnallisuuksien tekniset vaatimukset ja työmääräarviot. Isomman kehitystiimin puitteissa hoidan vuorollani määrättyjä päivystys- ja ylläpitotehtäviä.

Aiemmassa toimeksiannossa työskentelin backend-kehittäjänä. Full stack -kehittäjän rooli on laajempi verrattuna backend-kehittäjään (Eddie 27.2.2022). Full stack -kehittäjä kehittää sovelluksen kaikkia osa-alueita tietokannasta käyttöliittymään ja mahdollisesti pilvipalveluihin saakka (W3Schools s.a.a). Aiemmasta toimeksiannosta minulle ovat tuttuja backend-kehityksen lisäksi perustason versionhallinnan käyttö, kehitysympäristön pystytys, lokien luominen ja lukeminen, debuggaus eli virheenjäljitys ja etätyökäytännöt. Frontend-kehitys ei myöskään ole minulle täysin uutta, sillä olen suorittanut siihen liittyviä kursseja osana opintojani ja kehittänyt osaamistani tällä saralla myös itseopiskellen. Uusia frontend-teknologioita minulle ovat sovelluksessa käytettävät TypeScript ja Redux-tilanhallinta. Myös työskentely osana isoa kehitystiimiä, sekä SAFe:n toimintamallien hyödyntäminen ovat minulle uutta. Pilvipalveluista minulla on vain pintapuolen tietämys, mutta tiimissä tämä osaaminen ei ole välttämätöntä, sillä työtehtäväni eivät ulotu pilvipalvelurajapintaan.

Työmenetelmät

Tiimissä käytetään työtapana SAFe ketterän kehityksen viitekehystä. Kehittäjän näkökulmasta tämä tarkoittaa osallistumista PI-suunnittelutapahtumiin, ketterän kehityksen toimintatapojen hyödyntämistä ja mahdollisuutta julkaista uusi ohjelmistoversio heti kun jokin kokonaisuus valmistuu (Hietaniemi 17.5.2016). Työvuosi jaetaan 8–12 viikon mittaisiin PI:hin, joiden puitteissa pyritään kehittämään sovellukseen kulloinkin priorisoidut ominaisuudet (Scaled Agile 6.9.2022). Sovellustiimi käyttää työtapana Kanban-projektinhallintamenetelmää, jossa kaikki työtehtävät ovat nähtävillä tiimin yhteisellä Kanban-taululla (Koskinen 26.3.2021). Monissa sovellustiimin kehityskohteissa on riippuvuuksia toisiin tiimeihin, joten edistämme tikettejä siinä järjestyksessä, kun se on mahdollista.

2.2 Sidosryhmien esittely

Kuvassa 1 sivulla 6 on esitelty sovellustiimin työhön liittyvät sidosryhmät, jotka ovat kaikki konsernin työntekijöitä ja näin ollen sisäisiä sidosryhmiä (WS Finance Oy 19.7.2019).

Sidosryhmiä ovat:

- Sovellustiimin jäsenet
- Muut sovellustiimit
- Kehityspäällikkö
- Liiketoiminnan edustajat
- Sovelluksen tukipalvelu
- Sovelluksen käyttäjät

Sovellustiimin jäsenet sopivat keskenään työlistalla olevien kehitystöiden jakamisesta. Mikäli sovellustiimin sisältä ei löydy tarvittavaa osaamista, kysytään apua muilta sovellustiimeiltä. Liiketoiminnan edustajat ovat vastuussa, että sovellustiimin työ tuottaa liiketoiminta-arvoa ja vastaa määrityksiä (Scaled Agile 10.2.2021a). Liiketoiminnan edustajat kommunikoivat kehityspäällikön kanssa, joka puolestaan pitää huolen, että sovellustiimi ymmärtää kehityskohteiden sisällön. Sovelluksen tukipalvelut kouluttavat ja auttavat käyttäjiä mahdollisissa virhetilanteissa. Mikäli tukipalvelut eivät saa ongelmaa ratkaistua, he ovat yhteydessä sovellustiimiin. Sovelluksen käyttäjät ovat konsernin sisäisiä, rakentamisen ja talotekniikan kaupan työntekijöitä.

2.3 Työpaikan vuorovaikutustilanteet

Kehitystiimissä työskentelee henkilöitä useasta eri maasta ja myös Suomen sisällä useasta eri kaupungista, siten tiimin yhteinen työkieli on englanti. Yrityksen toimitilat sijaitsevat Helsingissä ja kampuksella on mahdollista halutessaan työskennellä. Tiimin hajaantuneisuuden vuoksi suurin osa kommunikaatiosta ja palavereista tapahtuu kuitenkin verkon välityksellä. Palaverit pidetään yleensä Microsoft Teams -sovelluksen kautta ja muunlaiseen pikaviestintään käytetään Slack-sovellusta. Tikettejä hallinnoidaan Atlassian Jira -ohjelmiston kautta ja ohjeita ja dokumentaatiota kirjoitetaan saman yrityksen Confluence-alustalle.

Vuorovaikutustaitoni ovat hioutuneet aiemman urani aikana kaupan alalla ja erilaisissa asiakaspalvelutehtävissä, joten viestiminen erilaisissa yhteyksissä on luontevaa. Toisinaan todella teknisten asioiden ilmaisu on haastavaa, mutta opiskelen sitä lisää jatkuvasti esimerkiksi lukemalla eri ohjelmointikielien dokumentaatioita.

SAFe

Virallisista SAFe:en kuuluvista tapahtumista tiimissä järjestetään PI-suunnittelutapahtuma neljä tai viisi kertaa vuodessa. PI-suunnittelutapahtumaan osallistuvat kaikki kehitystiimin jäsenet sekä myös liiketoiminnan edustajat. PI-suunnittelutapahtumassa suunnitellaan nimensä mukaisesti seuraavavan inkrementin kehitystöitä. Lisäksi tiimissä on käytössä Mid PI-review, joka järjestetään inkrementin puolivälissä samalla kokoonpanolla kuin suunnittelutapahtuma. Mid PI-review:ssa käydään lyhyesti läpi kaikkien sovellustiimien kehityksen tilanne.

Kehitystiimi

Kehitystiimi kokoontuu päivittäin dailyn muodossa, jossa käydään läpi kaikkia koskevia asioita sekä sovellusten julkaisuun liittyviä seikkoja. Kehitystiimi viestii myös Slack-sovellukseen perustetuilla kanavilla. Kaikkia koskevia asioita ovat esimerkiksi pilvipalveluiden asetukset ja turvallisuus, sekä äkillisesti ilmenneet tuotanto-ongelmat. Oma kanavansa löytyy myös vapaammalle viestinnälle ja avatuille pull requesteille.

PR

PR eli pull request on versionhallintaan avattava tila, jonka avulla toinen kehittäjä voi katselmoida tehtyä koodia ja jättää tarvittaessa kommentteja. Kehittäjä liittää pull requestin yhteyteen muutoksen pohjana toimineen tiketin sekä testausohjeet. Kun pull request on hyväksytty, voidaan tehty muutos yhdistää sovelluksen muuhun koodiin.

Sovellustiimi

Sovellustiimi kokoontuu myös päivittäin oman dailyn muodossa. Sovellustiimin dailya vetää tuoteomistaja ja dailyn perustana on sovellustiimin Jira-taulu. Dailyssä käydään SAFen mallin mukaisesti läpi (Scaled Agile 10.2.2021b):

- Mitä tein eilen?
- Mitä teen tänään?
- Onko työlle jotain esteitä tai riippuvuuksia?

Dailyssa sovitaan myös seuraavan sovellusversion julkaisuun ja testaamiseen liittyvistä seikoista. Muita sovellustiimin yhteisiä tapahtumia ovat tikettien määrittely ja tulevien kehitystöiden suunnittelu.

3 Seurantajakson raportointi viikkoanalyyseineen

Seurantajakson raportoinnissa on kerrottu päiväkohtaisesti työtehtävistä ja jokaista raportointiviikkoa seuraa viikkoanalyysi. Seuraavaksi esitetyt seurantaviikot, päivien kulku ja niiden tapahtumat on kuvattu lukijalle vain siitä näkökulmasta, että päivän tapahtumat kuvaavat asetettuja oppimistavoitteita.

3.1 Seurantaviikko 1

Tämän viikon tavoitteena oli osallistua PI-suunnittelutapahtumaan ja ymmärtää tapahtuman kulku sekä tavoitteet. Lisäksi tavoitteena oli, että uudet kehitystyöt alkavat välittömästi.

Maanantai 5.9.2022

Tänään oli PI-suunnittelutapahtuman ensimmäinen päivä. Kehitystiimin johtajan puheenvuoro sisälsi tiimissä tapahtuneita muutoksia ja työskentelytapoja. Myös työhyvinvointiin ja taukojen pitämiseen kiinnitettiin huomiota. Sen jälkeen yksi sovellustiimien kehityspäälliköistä kävi läpi edellisen PI:n tavoitteita ja miten ne oli saavutettu. Sitten saivat puheenvuoron liiketoiminnan edustajat eri maista. Liiketoiminnan edustajat kertoivat talouden lukuja, kehittymistä ja digitaalisten palveluiden osuutta liikevaihdosta. Puheenvuorojen jälkeen käytiin vielä läpi eri maiden liiketoimintojen asettamia tavoitteita tulevalle PI:lle. Asetetut tavoitteet koskivat tiimiä kokonaisuutena sisältäen myös liiketoiminnan edustajat. Haluttiin painottaa, että tavoitteet eivät olisi vain kehitystiimin harteilla. Liiketoiminnan asettama tavoite saattoi olla esimerkiksi: “Haluamme palvella asiakasta paremmin tarjoamalla hänelle enemmän toimitusvaihtoehtoja”. Tavoitteiden avulla oli tarkoitus tarjota lisätietoa ja motivaatiota kehitystöiden taustalle.

Yhteisten puheenvuorojen jälkeen jakaannuimme sovellustiimeissä breakout-kokouksiin, jotka ovat oleellinen osa PI-suunnittelutapahtumaa. Kokouksissa käytiin tulevia kehitystöitä läpi erilaisilla kokoonpanoilla. Sovellustiimin (kehittäjät ja tuoteomistaja) lisäksi mukana oli kehityspäällikkö, liiketoiminnan edustaja, designer, SAP-asiantuntija sekä muita mahdollisia tarpeellisia asiantuntijoita. Kehitystyöt pyrittiin määrittelemään mahdollisimman tarkasti, jotta kaikille oli selvää ainakin karkealla tasolla mihin pyritään ja mikä on työmääräarvio. Teknisiä yksityiskohtia ei vielä määritelty, mutta tarkoituksena oli selvittää mitä kaikkea tulisi tehdä, jotta tavoite ja MVP saavutetaan.

Ensimmäiselle PI-suunnittelutapahtumanpäivälle oli aikataulutettu kaksi breakout-kokousta, joista meidän sovellustiimimme käytti molemmat kokoukset. Päivän päätteeksi kokoonnuimme vielä koko tiimin kanssa yhteen ja kävimme läpi esiin nousseita kysymyksiä ja tunnistettuja riskejä.

Tiistai 6.9.2022

Tänään oli PI-suunnittelutapahtuman jälkimmäinen päivä. Päivä alkoi lyhyellä yhteisellä kokoontumisella, jonka jälkeen jakaannuttiin jälleen breakout-kokouksiin. Breakout-kokouksia oli aikataulutettu tälle päivälle peräti neljä kappaletta, jokainen reilun tunnin kestoltaan. Omalla sovellustiimilläni ei ollut breakout-kokouksia tälle päivälle, mutta seurasin muiden tiimien kokouksia. Kokouksissa sai hyvän käsityksen muiden sovellustiimien tulevista kehitystöistä ja mahdollisuuden esittää ehdotuksia ongelmien ratkaisuksi. Breakout-kokouksissa käsiteltyjä kehityskohteita oli toki valmisteltu jo ennalta, ennen PI-suunnittelutapahtumaa. Myös designerit olivat ehtineet tehdä ensimmäiset luonnoksensa, joita sitten käytiin läpi kokouksissa liiketoiminnan edustajien ja kehittäjien kanssa. Oli todella mielenkiintoista kuulla perusteluita designerilta, miksi jokin tapa oli hänestä käyttäjäystävällisempi kuin jokin toinen. Jos yhdessä keskustellen löydettiin jokin parempi tapa esittää asia, designer teki muutoksia suunnitelmaansa.

Päivän päätteeksi keräännyimme jälleen yhteen PI-suunnittelutapahtuman päätöstä varten. Ensin kävimme läpi keskustellen sovellustiimien tunnistamat riskit ja miten niitä hallittaisiin. Riski joko hyväksyttiin sellaisenaan, riskiä pyrittiin lieventämään jollain keinolla, tai riskin poistamiseksi nimettiin vastuuhenkilö. Seuraavaksi käytiin läpi kehitystyöt tiimeittäin prioriteettijärjestyksessä, jotta kaikki saivat näkyvyyttä muiden kehitysjonossa oleviin töihin. Sitten käytiin vielä uudelleen läpi liiketoiminnan asettamat tavoitteet tälle PI:lle ja keskusteltiin ovatko tavoitteet saavutettavissa. Lopuksi tästä pidettiin vielä luottamusäänestys ja keskusteltiin sen lopputuloksesta.

Keskiviikko 7.9.2022

Tänään oli ensimmäinen varsinainen työpäivä tällä PI:llä. Aamupäivällä olivat normaalit daily-rituaalit, ensin koko tiimin kattava daily ja sitten oman sovellustiimimme daily. Meidän tiimimme pääasiallinen tavoite tälle PI:lle oli laajentaa kehittämämme sovelluksen käyttöä uuteen maahan. Tähän liittyviä tikettejä ei oltu määritelty vielä valmiiksi, ja usea vanha tiketti edelliseltä PI:ltä oli jumissa ulkopuolisista syistä johtuen. Varsinaista PI työtä ei ollut siis vielä tarjolla.

Omien tikettien puuttuessa tein koodikatselmointeja muille kehittäjille, joka onkin loistava tapa tutustua kokeneempien kehittäjien koodiin ja käytäntöihin. Minun osaltani katselmointi on lähinnä sitä, että pyrin ymmärtämään mitä koodissa tapahtuu ja totean että toiminnallisuus toimii testausohjeiden mukaisesti. Toki jos löytäisin jotain virheitä esimerkiksi käännöksissä tai kirjoitusvirheitä koodissa, huomauttaisin niistä tekijälle. Testasin ja mergesin myös muutaman projektin riippuvuuden versionumeron, joita Githubin dependabot suositteli. Tämä ei ole varsinaista näkyvää työtä, mutta saattaa ehkäistä tietoturva-aukkoja ja pitää ohjelmiston käyttämien riippuvuuksien versiot ajan tasalla.

Torstai 8.9.2022

Aamu alkoi jälleen koodikatselmoinnilla. Minulla on tapana tehdä ne kaikessa rauhassa, jotta oikeasti voin käydä koodin huolella läpi. Ensi alkuun katselmoimani koodi oli minulle todella vaikeasti ymmärrettävää, mutta googlattuani muutaman asian pääsin jo hyvin kärryille. Tälläkään kertaa en löytänyt koodiin mitään parannusehdotuksia ja hyväksyin pull requestin.

Daily-rutiinien jälkeen vuorossa oli tikettien määrittelyä. Kuten mainittu meidän tiimimme suurin teema tällä PI:llä on sovelluksen julkaisu uuteen maahan. Tätä varten oli tehty jo aiemmin paljon tausta- ja tutkimustyötä, mutta nyt oli aika määritellä ensimmäiset varsinaiset tiketit. Saimme jaoteltua ensimmäisen teeman luontevasti tiimimme kolmen kehittäjän kesken, sekä määriteltyä tikettien sisällön, hyväksymiskriteerit, MVP:n ja työmääräarviot. Minulle näistä tiketeistä nimitettiin frontend-muutoksen tekeminen. Tämä sopi minulle paremmin kuin hyvin, sillä juuri frontend-kehittämisessä halusinkin kehittyä. Lähdin heti työstämään tikettiä (jatkossa tiketti yksi) ja sainkin sen jo ihan hyvään malliin päivän loppuun mennessä. Tiketin yksi perimmäinen tarkoitus oli lisätä käännöstiedostot uudelle kielelle ja mahdollisuus valita liiketoiminta-alue sekä haluttu käyttökieli. Tiketin keskeisenä tekniikkana ja minulle hieman uutena asiana toimi Reactin Redux-tilanhallinta, jota päädyin myös opiskelemaan päivän aikana.

Perjantai 9.9.2022

Sain valmiiksi ensimmäisen version tiketillä yksi määritetyistä kehitystöistä. Tämä tuntui hyvältä, sillä tiketille oli annettu työmääräarvioksi viisi työpäivää ja oli mennyt vasta alle päivä. Dailyjen jälkeen pidin palaverin kokeneemman kollegan kanssa, jolloin tulikin palautus maan pinnalle. Koodi kyllä periaatteessa toimi, mutta toteutukseni oli aivan liian suoraviivainen, eivätkä datarakenteet olleet kohdillaan. Tämä aiheuttikin sitten loppupäiväksi ihmettelyä, kokeilua ja uuden opiskelua. Opiskelin Redux-tilanhallintaan liittyen, miten joitain tilan osia voitiin persistoida ja miten ne taas saataisiin tarpeen tullen resetoitua.

Viikkoanalyysi 1

Opinnäytetyön ensimmäinen viikko alkoi intensiivisellä mutta antoisalla PI-suunnittelutapahtumalla. Viikon tavoitteena oli osallistua PI-suunnittelutapahtumaan ja saada lisätietoa loppuvuoden kehitystöistä. Asetettu tavoite täyttyi, sillä osallistuin kaikkiin oman sovellustiimini breakout-kokouksiin sekä sain lisätietoa myös muiden sovellustiimien kehitystöistä. Liiketoiminnan edustajien kehitystöille antama taustatieto kasvatti motivaatiota ja ymmärrystä kehitystöiden tekemiselle. Myös asetettu tavoite kehitystöiden aloittamiselle täyttyi, sillä minulle nimettiin ensimmäinen tiketti sekä sain sitä hyvin edistettyä tämän viikon aikana.

PI-suunnittelutapahtuma on osa SAFea, ketterän kehityksen skaalautuvaa viitekehystä (Harle 3.5.2018). PI-suunnittelutapahtuma kestää kaksi työpäivää ja sitä suositellaan pidettäväksi kasvokkain. Kaikkien sidosryhmien saaminen samaan fyysiseen tilaan edistää yhteisiin tavoitteisiin sitoutumista ja tiimien välistä yhteistyötä. (Scaled Agile 10.2.2021c.) Koronavirusepidemian sekä sen vuoksi että tiimissä on jäseniä useista eri maista, PI-suunnittelutapahtuma järjestettiin niin sanotun hybridimallin mukaisesti. Kaikki halukkaat kokoontuivat konttorille tapahtumaa varten ja loput osallistuivat Teams-yhteyden välityksellä. Tämä onnistui yllättävän hyvin, vaikkakin oli havaittavissa ongelmia esimerkiksi äänentoiston kanssa, kun linjoilla olevat yrittivät kuulla mitä isossa projektihuoneessa keskusteltiin kasvokkain.

PI-suunnittelutapahtumassa on ennalta määrätty ohjelma ja aikataulu. Tyypillisesti ensin ääneen pääsevät liiketoiminnan edustajat, jotka kertovat esimerkiksi liiketoiminnan tilasta ja kehitettyjen ohjelmistojen tuomasta hyödystä asiakkaille. Tämän jälkeen esitellään tulevat kehityskohteet ja saatetaan keskustella ketterää kehitystä tukevista toimintamalleista. Tiimien breakout-kokouksissa tarkastellaan kehitysjonossa olevia kohteita, tunnistetaan niihin liittyviä riskejä ja riippuvuuksia muiden tiimien töihin, sekä aikataulutetaan ne alustavasti tulevan PI:n varrelle. Ensimmäisen päivän päätteeksi suunnitelmia tarkastellaan ja liiketoiminnan edustajat sekä johto pääsevät antamaan palautetta sekä ratkomaan mahdollisia ongelmia. Toisena suunnittelutapahtuman päivänä jatketaan tiimien breakout-kokousten parissa. Lopuksi tarkastellaan kaikkien tiimien suunnitelmia ja havainnoidaan löydetyt riskit. Riskit voidaan joko ratkaista heti, niille voidaan kehittää ratkaisusuunnitelma, ne voidaan hyväksyä sellaisenaan tai niille voidaan määrätä vastuuhenkilö, joka ajaa ongelman ratkaisua myöhemmässä vaiheessa. PI-suunnittelutapahtuman päätteeksi järjestetään luottamusäänestys, ovatko kaikki tiimin jäsenet luottavaisia PI:lle asetettujen tavoitteiden suhteen. (Scaled Agile 10.2.2021c.) Meidän PI-suunnittelutapahtumamme noudatti yllä olevaa esimerkkikaavaa suhteellisen tarkasti.

Kyseessä oli jo kolmas PI-suunnittelutapahtuma, johon osallistuin tässä tiimissä. Tiesin siis jo odottaa mielenkiintoisia keskusteluja ja case-esimerkkejä kehitystyön taustalle. Muuten SAFe ei varsinaisesti ollut minulle ennestään kovin tuttu viitekehys. Ennen tapahtumaa kertasin SAFen ja PI-suunnittelutapahtuman käytäntöjä, saadakseni tapahtumasta enemmän irti. Kävin myös huolella läpi oman sovellustiimin teemat ja tulevat työtehtävät, jotta myös minulla olisi mahdollisesti jotain sanottavaa breakout-kokousten yhteydessä. Tapahtuman yhtenä tuotoksena oli kaikkien tiimien yhteinen Miro-taulu, josta oli nähtävissä tiimien kehitystehtävät aikataulutettuna. Suunnittelutapahtumassa käytiin myös läpi tiimissä kesän aikana tapahtuneita muutoksia, jotka tulisivat väistämättä vaikuttamaan kaikkien tiimien aikatauluihin. Ongelmien tunnistaminen ja niihin reagoiminen toi tiimin keskuuteen rauhaa ja varmuutta.

Frontend-kehitykseen liittyen opiskelin tällä viikolla Redux-tilanhallintakirjaston käyttöä. Redux-tilanhallintakirjastoa voi käyttää eri ohjelmistokehysten, kuten Reactin tai Angularin kanssa. Reduxin avulla jokin tila, esimerkiksi asiakasnumero on koko sovelluksen käytössä, kun tilaa on pelkällä Reactilla käsitelty lähinnä komponenttikohtaisesti tai propsien välityksellä. (Codesphere 1.4.2022.) Ottamalla käyttöön taas Redux Persist-kirjaston, tila voidaan tallentaa sovelluksen välimuistiin. Näin ollen sovellus muistaa sen, vaikka palvelusta kirjautuisi välillä ulos. (Briggs 3.6.2022)

3.2 Seurantaviikko 2

Seurantaviikkojen raportointi jatkui viikon tauon jälkeen. Tämän viikon tavoitteena oli selkeyttää itselle Git-versionhallintatyökalun käyttöä sekä luoda rutiinia merge-konfliktien ratkaisuun. Lisäksi tavoitteen oli oppia lisää TypeScriptin tyyppimäärityksistä.

Maanantai 19.9.2022

Aamulla palauttelin mieleen mihin olinkaan jäänyt toissaviikolla kesken jääneen tiketin yksi kanssa. Opiskelin TypeScriptin Record tyyppimääritystä ja sain koodissa olleen ongelman ratkaistua. Puskin toimivan version koodista versionhallintaan. Tämän jälkeen pidin palaverin kokeneemman kollegan kanssa, jolle oli tullut mieleen lisämäärityksiä tiketin yksi puitteissa tehtäville töille. Kirjasin uudet vaatimukset ja niitä varten tarpeelliset toimenpiteet itselleni ylös ja valmistauduin myös perustelemaan tuoteomistajalle, miksi työmääräarvio saattaisi venyä hieman alkuperäisestä. Kysyin kollegalta apua myös muutamassa ilmenneessä TypeScript ongelmassa, jotka estivät etenemisen tehtävässä.

Harmittavan usein työni tuntuu tyssäävän TypeScriptin tyypityksiin ja datarakenteisiin, joista en selvästikään ole aivan riittävän perillä, joten esimerkiksi jokin verkkokurssi aiheesta tulisi tarpeeseen. Tässä suhteessa konsulttina työskentely juniorina on haastavaa, kun ajoittain tarvitsisi enemmänkin aikaa opiskeluun. Koska kuitenkin tekee laskutettavaa työtä asiakkaalle, joutuu usein vain nopeasti selvittämään kyseessä olevan ongelman ja mahdollinen opiskelu jää vapaa-ajalle.

Tiistai 20.9.2022

Tänään jatkoin jälleen työskentelyä tiketin yksi parissa. Olin saanut tiketin alkuperäisen määritelmän mukaisen toiminnallisuuden valmiiksi, ja nyt lähdin toteuttamaan lisämäärityksen mukaista toiminnallisuutta. Kyseessä oli valintanapin valinnan jälkeen avautuva modaali, jonka avulla varmistetaan, että valittua arvoa halutaan todella muuttaa. Designerimme lomaviikon vuoksi en vielä saanut varmistusta oliko tämä käyttäjäkokemuksen puitteissa paras ratkaisu ongelman ratkaisemiseksi.

Päivän aikana toteutui muutama palaveri. Ensimmäisessä palaverissa käsittelimme kehittäjien kesken tarpeellisia ylläpitotoimia, joita tulisi tehdä lähitulevaisuudessa. Ylläpitotoimille määriteltiin alustavat työmääräarviot sekä järjestettiin prioriteettijärjestykseen. Kirjattiin ylös myös riskit, jotka seuraisivat, mikäli esimerkiksi jonkin riippuvuuden versionumeroa ei päivitettäisi ajoissa. Näin ollen työt olisi helpompi perustella liiketoiminnan edustajille, jotka määrittelevät kehitysjonon prioriteettijärjestystä. Toinen palaveri koski kehittämäni sovelluksen uuteen maahan lanseerauksen seuraavaa vaihetta. Kävimme läpi missä mennään ensimmäisen vaiheen kanssa ja mitkä ovat seuraavat toimenpiteet. Mietimme yhdessä myös olisiko jotain, jonka tekemisen voisi aloittaa heti, ja toisaalta mitä riippuvuuksia tekemiselle on. Minulle nimitettiin sovelluksen uuteen käyttökieleen liittyvä tiketti (jatkossa tiketti 2), jonka puitteissa minun tulisi luoda tiedosto, johon liiketoiminnan edustajat tekevät käännökset.

Keskiviikko 21.9.2022

Tänään jatkoin edelleen työskentelyä tiketin yksi parissa. TypeScriptin Record tuotti edelleen päänvaivaa, sillä Recordilla määriteltyihin tietoihin tuntui hieman hankalalta päästä käsiksi, joten jouduin jälleen googlaamaan asiaa. Lopulta löysin tarvittavat tiedot googlesta ja sain koodin toimimaan halutulla tavalla. Lisäksi opin taas hieman lisää TypeScriptin Record-tyypistä ja kuinka siihen voidaan lisätä Partial-määritys, jolloin osa määritellyistä avaimista voi olla valinnaisia. Katsoin myös toiminnallisuuden läpi kokeneemman kollegan kanssa ja hänenkin mielestään kaikki vaikutti hyvältä. Päätin jättää koodin viimeistelyn ja siivoamisen seuraavalle päivälle ja keskittyä loppupäivän tiketin kaksi käännöstiedoston valmisteluun. Yritin miettiä kompromissia tiedostomuodosta, johon uuden maan liiketoiminnan edustajat tekisivät käännökset, sillä käännökset ovat sovelluksen sisällä JSON-muodossa, ja liiketoiminta tekee käännökset yleensä Excel-tiedostoon.

Torstai 22.9.2022

Tänään viimeistelin tiketin yksi koodikatselmointia varten. Viimeistely tarkoitti muuttujien nimien yhtenäistämistä ja refaktorointia, turhien koodirivien ja kommenttien poistamista ja koodin kommitointia versionhallintaan sopivissa osissa. Lisäksi otin käyttöön feature flagin, jolla estettiin tekemieni muutosten näkyminen muualla kuin kehitys- ja testiympäristöissä. Lopuksi ratkaisin vielä kasan merge-konflikteja, jotka aiheutuivat siitä, että versionhallinnan master-haaraan oli vihdoin mergetty useampi aiempi tiketti, jotka olivat odottaneet vuoroaan. Merge-konfliktien ratkominen on aiemmin ollut minulle jännittävää, jopa hieman pelottavaa, mutta nyt vihdoin toistojen ja hyvien ohjeiden myötä onnistuin siinä ilman virheitä. Jäin odottamaan, että kokeneemmalla kollegalla olisi aikaa katselmoida koodini. Katselmoinnin jälkeen oli odotettavissa, että tulisi tehdä vielä joitain korjauksia, mutta olin tyytyväinen, kun sain työn valmiiksi arvioidussa viidessä työpäivässä.

Perjantai 23.9.2022

Tänään kollegani katselmoi tiketin yksi koodin ja antoi siitä palautetta. Palaute liittyi lähinnä koodin luettavuuteen, jonka arvioimiseen oma silmäni ei ole vielä niin harjaantunut. Kaikki korjausehdotukset kävivät kuitenkin järkeen ja opin taas hieman lisää luettavan koodin tuottamisesta. Monesti miellän itse luettavan koodin yhtä suureksi kuin mahdollisimman tiiviiseen muotoon kirjoitettu koodi, esimerkiksi korvaamalla moniriviset ehdolliset funktiot uudentyyppisillä tiiviimpää muotoon kirjoitetuilla versioilla. Joskus kuitenkin koodin luettavuuden kannalta on perusteltavaa käyttää vanhaa if - else rakennetta, varsinkin jos ehto on monimutkainen. Kollegani antama palaute oli perustelevaa ja hän oli antanut konkreettisia esimerkkejä, miten koodia voisi parantaa. Tämän perusteella oli helppo refaktoroida koodia ja oppia samalla lisää.

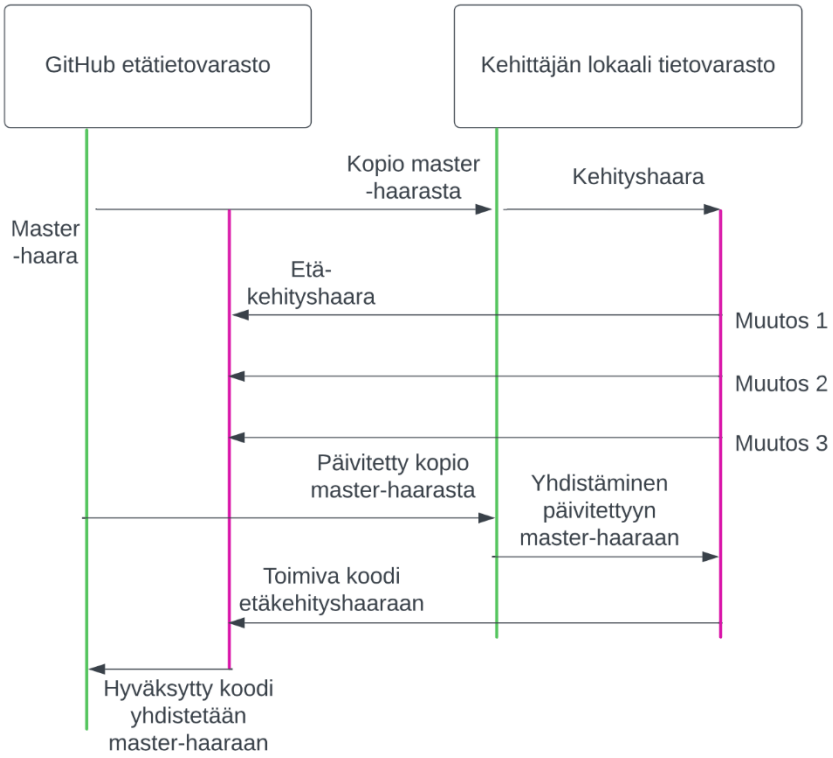
Viikkoanalyysi 2

Tämän viikon ensimmäisenä tavoitteena oli sujuvoittaa Git-versionhallintatyökalun käyttämistä ja merge-konfliktien ratkaisua. Tavoite toteutui, sillä loin itselleni kattavat muistiinpanot aiheesta ja harjoittelin komentoja käytännössä. Puskin omalla paikallisella kehityshaarallani tekemäni muutokset koodiin versionhallinnan etähaaraan. Ennen kuin koodi voidaan yhdistää versionhallinnalla master-haaraan, eli siihen haaraan, joka on nähtävissä testiympäristössä, tulee tarkastaa mahdolliset merge-konfliktit. Konflikteja saattaa syntyä, kun useampi kehittäjä kehittää omia kehityshaarojaan, joissa saatetaan muokata samoja tiedostoja. Merge-konfliktin ratkaisutilanteessa kehittäjä katsoo, että omat tekemät muutokset sopivat muiden kehittäjien muutosten kanssa ja että koodi toimii edelleen odotetusti.

Tiimissä käytetään Git-versionhallintatyökalua ja koodit tallennetaan sen avulla GitHub-verkkosivuston kautta hallinnoitavalle alustalle. Git-versionhallintatyökalun avulla useat kehittäjät voivat työskennellä saman koodikannan parissa samanaikaisesti ja tehtyjä muutoksia voidaan seurata helposti. GitHub on taas palvelu, joka toimii koodin etätietovarastona. (Sovelluskontti Oy s.a.)

Työn kulku on havainnollistettu tarkemmin kuvassa 2 sivulla 17. GitHub toimii koodikannan etätietovarastona. GitHubissa sijaitsee niin testiympäristön master-haara kuin kehittäjien omat etäkehityshaarat. Aloitettaessa uuden toiminnallisuuden kehittämistä, kehittäjä lataa viimeisimmän version master-haarasta omalle koneelleen. Sitten master-haaran päälle ruvetaan kehittämään uutta, luomalla paikallinen kehityshaara. Paikallisessa kehityshaarassa tehdyt muutokset pusketaan etätietovarastolla sijaitsevaan etäkehityshaaraan. Kun haluttu toiminnallisuus on saatu valmiiksi, ladataan master-haara vielä kerran omalle koneelle, sillä joku toinen kehittäjä on saattanut päivittää sitä. Master-haara yhdistetään omaan kehityshaaraan ja ratkotaan mahdollisesti ilmenneet merge-

konfliktit. Sitten kehityshaara pusketaan vielä kertaalleen etätietovaraston etäkehityshaaraan. Toinen kehittäjä katselmoi koodin, ja kun tarvittavat muutokset on tehty, se hyväksytään. Sen jälkeen etäkehityshaaran koodi voidaan mergetä eli yhdistää etätietovaraston master-haaraan. Master-haaran koodi pyörii testiympäristössä ja kun se on käynyt läpi testausprosessin, se viedään tuotantoon.



Kuva 2. Työn kulku etätietovaraston ja lokaalin tietovaraston välillä

Versionhallinta säilöö nimensä mukaisesti versioita koodista. Versionhallinnassa on nähtävillä jokainen kuhunkin tiedostoon tehty muutos, sen tekijä, päivämäärä ja kehittäjän muutoksesta kirjoittama viesti. Versionhallinnan avulla, mikäli jokin menisi pieleen, voidaan aina palata aiempaan toimivaan versioon koodista. Versionhallinta mahdollistaa usean kehittäjän samanaikaisen työskentelyn ohjelmiston parissa. Tämä toteutetaan luomalla kehityshaaroja, jotka ovat itsenäisiä toimivia versioita ohjelmistosta, joihin on lisätty esimerkiksi jokin uusi toiminnallisuus. Valmistuttuaan kehityshaarat yhdistetään päähaaraan. Kehityshaarat voidaan yhdistää myös niihin liittyviin tiketteihin, jotka sijaitsevat tiketinhallintapalvelussa kuten Jira-sovelluksessa. (Atlassian s.a.)

Git-versionhallintatyökalun on keittänyt omiin tarpeisiinsa myös Linuxin kehittäjä tunnettu Linus Torvalds. Gittiä käytetään laajasti sovelluskehityksessä, mutta sen avulla voidaan säilöä versioita myös esimerkiksi tekstimuotoisista koulutöistä. Git tulee asentaa omalle tietokoneelle ja mikäli sen

avulla halutaan säilöä projekteja GitHubiin, tulee sinne luoda käyttäjätunnukset. (Helsingin Yliopisto s.a.)

Tiimimme säilöö projekteja GitHubissa, joten siellä sijaitsevat projektit ovat valmiiksi Git-projekteja. Projektin tulee olla Git-projekti, jotta sitä voidaan hallinnoida Git-komennoilla. Mikäli projekti ei vielä olisi Git-projekti, voitaisiin se luoda ajamalla projektikansiossa komento “git init”. (Helsingin Yliopisto s.a.) Kuvassa 3 sivulla 19 on kuvattu tavanomainen työnkulku Git-komentoineen. Komentojen pohjalla on oletus, että projekti on kloonattu omalle koneelle.

Kuvan 3 ensimmäisessä koodiblokissa kuvataan uuden kehityshaaran aloittamista omalla koneella. Ensin siirrytään omassa kehitysympäristössä master-haaraan ja päivitetään se vastaamaan etätietovaraston master-haaraa. Tämän pohjalle luodaan uusi kehityshaara ja lopuksi siirrytään siihen ja aloitetaan kehitys.

Kuvan 3 toisessa koodiblokissa kuvataan kehityksen aikana tarvittavia komentoja. Tehtyjä muutoksia ja eroa vanhaan koodiin voi tarkastella komennoilla “git status” ja “git diff”. Komennon “git add” avulla lisätään kaikki tai valikoidut muutokset valmiiksi kommitointia varten. Kehitys suoritetaan järkevä kokonaisuus kerrallaan, jonka jälkeen koodi kommitoidaan ja pusketaan kuvaavan viestin kera etäkehityshaaralle. Sopivan kokonaisuuden voi ajatella niin että kommit-viesti on helppo muodostaa, esimerkiksi “Lisää lähetyspainike palautelomakkeelle”.

Kuvan 3 kolmannessa koodiblokissa kuvataan tilannetta, kun kehitetty ominaisuus on valmiina ja se halutaan yhdistää master-haaraan. Master-haaraan yhdistellään kehityshaaroja yleensä aina niiden valmistuttua, joten master-haara on saattanut muuttua paljonkin siitä mitä se oli kehitystyön alkaessa. Ensin päivitetään lokaali master-haara vastaamaan etätietovaraston master-haaraa. Sitten rebase-komennolla yhdistetään oma kehitetty koodi ja lokaali master-haara toisiinsa. Mikäli useat kehittäjät ovat tehneet muutoksia samoihin tiedostoihin, saattaa niihin tulla merge-konflikteja. Konfliktit tulee ratkaista ja lisätä muutokset omaan etäkehityshaaraan. Komennolla –force-with-lease puskettavien kommittien sisältöä muutetaan niin, että ne sisältävät sekä ajantasaisen master-haaran sisällön että kehittäjän tekemät muutokset.

Työn kulkuun tarpeelliset Git-komennot on kuvattu seuraavassa kuvassa 3.

```
git checkout master
git pull
git branch uusi-kehityshaara
git checkout uusi-kehityshaara

git status
git diff
git add .
git commit -m "Kerro mitä muutoksia tehty"
git push origin uusi-kehityshaara

git checkout master
git pull
git checkout uusi-kehityshaara
git rebase master
git add tiedostoJossaRatkaistuKonflikti
git rebase --continue
git push origin uusi-kehityshaara --force-with-lease
```

Kuva 3. Työn kulkuun liittyvät Git-komennot

Viikon toinen tavoite liittyi TypeScriptin tyypitysten syvempään ymmärtämiseen. Tavoite täyttyi osittain, sillä tyypityksiä on lukuisia. Ehdin kuitenkin opiskella ja ottaa käyttöön käytännössä kuvassa 4 sivulla 20 esitetyn Record-tyypityksen.

TypeScript on JavaScript-ohjelmointikielen päälle rakennettu tyypitetty ohjelmointikieli. Tyypitetyissä ohjelmointikielissä muuttujille asetetaan datatyypit, esimerkiksi merkkijono tai luku, joka voi auttaa vähentämään virheitä koodissa. (Microsoft 2022.) Stack Overflow:n kehittäjille teettämän tutkimuksen mukaan TypeScript oli viidenneksi suosituin ohjelmointikieli vuonna 2022 (Ramel 28.6.2022). Seuratessani työpaikkailmoituksia ja keskustellessani opiskelukollegoideni kanssa olen havainnut, että TypeScriptiä käytetään projekteissa paljon ja sen osaaminen on etu työmarkkinoilla. Olen aiemmin käyttänyt tyypitetyistä ohjelmointikielistä Javaa, jonka ansiosta tyypitysten käyttö on minulle teoriassa ennestään tuttua. TypeScriptiin liittyviä opintoja tai työkokemusta sen kanssa työskentelystä minulla ei ennen nykyistä projektia ollut. TypeScriptin käyttö on tuntunut ajoittain haastavalta, sillä minulla ei ole ollut riittävää osaamista siihen liittyvistä datarakenteista tai tyypitysten syntaksista. TypeScriptin ominaisuuksiin kuuluu virheiden näyttäminen koodieditorissa, joka on sekä auttanut paljon kielen opiskelussa että aiheuttanut ihmettelyä virheiden syistä.

Tällä viikolla opin uutena asiana käyttämään TypeScriptin Record-tyypitystä ja sekä Partial-määritystä. Record-tyypityksen avulla määritellään tyypit avain arvo -pareille kun taas Partial-määrityksellä voidaan ilmaista jonkin arvon olevan valinnainen (Microsoft 19.12.2022). Määritysten käyttöä koodissa on esitelty kuvassa 4 sivulla 20. Record-tyypityksellä on ilmaistu, että avaimena toimii

koiran nimi ja arvona koiran tiedot sisältävä objekti. Partial-määritys puolestaan sallii koiran tietojen osittaisen esittämisen tai muokkaamisen.

```
interface DogInfo {
  age: number;
  breed: string;
}

type DogName = "musti" | "sulo" | "laikku";

const dogs: Record<DogName, Partial<DogInfo>> = {
  musti: { age: 3, breed: "Labradorinnoutaja" },
  sulo: { age: 5, breed: "Villakoira" },
  laikku: { age: 8 },
};
```

Kuva 4. TypeScript Record-tyypityksen käyttö

3.3 Seurantaviikko 3

Tämän viikon tavoite oli ymmärtää tarvittavat vaiheet, jotka kehittäjän tulee suorittaa ennen pull requestin avaamista GitHub-palvelussa. Lisäksi tavoitteena oli edistää muiden kehittäjien työtä tekemällä koodikatselmointeja.

Maanantai 26.9.2022

Tänään minulle ei löytynyt uutta tikettiä, jonka parissa olisin voinut työskennellä. Kehitysjono on välillä tyhjillään, sillä tikettien valmistelu on oma prosessinsa. Keskityin siis toisten kehittäjien koodien katselmointiin. Katselmointia vailla olevat koodit löytyvät tiimimme Slack kanavalta. Viestin yhteyteen on mainittu sekä tiketti, että linkki GitHubin pull requestiin. Aloitan katselmoinnin aina tutustumalla huolella tikettiin, jotta saan kokonaiskäsityksen mitä tiketin puitteissa on ollut tarkoitus tehdä. Usein kehittäjät kirjoittavat jonkinlaiset testausohjeet pull requestin yhteyteen, joten tutustun myös niihin. Sitten otan tiketin kehityshaaran omalle koneelleni ja rupean tutkimaan koodia ja toiminnallisuutta tarkemmin. Mikäli ilmenee jotain korjausta vaativaa, GitHubissa voi jättää kommentteja kyseisen koodipätkän kohdalle. Jos taas ei löydy korjattavaa, tyypillisesti kuittaa koodin hyväksytyksi saatesanoilla “Testattu paikallisesti, toimii odotetusti”.

Tiistai 27.9.2022

Tänään otin yhteyttä lomalta palanneeseen designeriin ja kysyin varmistusta tiketille yksi kehittämäni modaalin käyttäjäystävällisyydestä. Designer hyväksyi toteutukseni ja antoi muutaman vinkin,

miten varmistusmodaalin infotekstiä kannattaisi muokata, jotta se olisi riittävän informatiivinen käyttäjälle. Toteutin vielä ehdotetut muutokset ja lähetin tiketin yksi koodin uudelleen katselmoitavaksi. Lisäksi kirjoitin tiketille kattavat testausohjeet, jotta sitä testaavan henkilön olisi helppo todentaa, että kaikki toimii odotetusti. Jatkoin myös tiketin kaksi käännöstiedoston muokkaamista liiketoiminnalle ymmärrettävään muotoon sekä tutustuin kollegan kanssa SAP:iin. Kehittämämme sovellus on ikään kuin parempi käyttöliittymä SAP:ille, joka pyörii kuitenkin kaiken taustalla. Oli siis todella mielenkiintoista päästä näkemään miten jokin asia, esimerkiksi tilauksen luominen voitiin tehdä SAP:in päässä. Iltapäivällä huomasin, että olin jälleen saanut parannusehdotuksia tiketille yksi, mutta jätin korjaukset seuraavalle päivälle.

Keskiviikko 28.9.2022

Tänään korjasin tiketille yksi vielä viimeiset pienet asiat mitä kollegani oli koodikatselmoinnin puitteissa pyytänyt. Kollegani kävi kanssani läpi JavaScriptin async/await-funktiota, kun kehittämäni toiminnallisuus päivitti sovelluksen tilaa ja local storagea epäloogisesti.

Meillä oli myös palaveri liittyen kehittämämme sovelluksen lanseeraukseen uuteen maahan. Määrittelimme tähän liittyen tulevien workshoppien aiheet ja listasimme valmiiksi seikat, joista meidän pitäisi kysyä kyseisen maan liiketoiminnalta tai muilta osaajilta. Löysin lisäksi pienen virheen liittyen kollegani kehittämään toiminnallisuuteen, josta loin tiketin lisätietoineen, lähetin tuoteomistajalle hyväksyttäväksi ja nimitin tiketille oikean vastuuhenkilön.

Torstai 29.9.2022

Tiimini kehitysjono oli edelleen tyhjä, joten teimme kukin tahoillamme vähän sitä sun tätä. Eilen viimeistelemääni tikettiin yksi oli eilen masteriin mergettyjen pull requestien johdosta tullut vielä yksi merge-konflikti, jonka ratkaisin aamulla. Tämän jälkeen tikettini ja pull requestini vihdoin hyväksyttiin. Työstin tuoteomistajan pyynnöstä tiketin kaksi käännöstiedosto Exceliin ohjeet liiketoiminnalle, miten Excel tulisi täyttää. Lisäksi selvittelin tuotannossa ilmennyttä ongelmaa, jonka käyttäjä oli raportoinut meille suoraan. Lueskelin lokeja käyttämästämme lokipalvelusta sekä pilvipalvelun lokeista, mutta asia ei vielä selvinnyt, ja lisäselvittely siirtyi seuraavalle päivälle. Tein myös muutaman pienen koodikatselmoinnin. Onneksi voi auttaa muiden töiden edistymisessä, sillä välillä kun itsellä ei ole avoimia tikettejä työn alla.

Perjantai 30.9.2022

Tänään jatkoin pääasiassa koodikatselmointien tekemistä. Tuotannossa olleen ongelman selvittely siirtyi vielä maanantaille, sillä en kokenut oloani riittävän itsevarmaksi soittaakseni ongelmasta raportoineelle käyttäjälle itsekseni. Pyysin kollegalta tukea opetellakseni lisää lokien lukemista tiimin käyttämästä lokipalvelusta ja tuotannon tietokannan tutkimista, mutta sen enempää en päässyt eteenpäin.

Viikkoanalyysi 3

Tämän viikon tavoite täyttyi, sillä keskityin enimmäkseen koodikatselmointeihin ja pull requestien analysointiin, oman sovellustiimin kehitysjonon ollessa tyhjä. Koodikatselmoinnit ovat itsessään myös tärkeää työtä, jota jokaisen kehittäjän tiimissämme tulee tehdä. Tyypillisesti koodit katselmoidaan oman kehitystiimin sisällä, niin että toinen kehittäjä samasta kehitystiimistä tekee katselmoinnin. On usein helpompaa katselmoida koodia, jonka parissa itsekin työskentelee. Lisäksi kehitystiimin sisällä kaikki kehittäjät ovat olleet tikettien määrittelypalavereissa, joten on helpompi ymmärtää, toimiiko kehitetty ominaisuus odotetulla tavalla. Toki kaikki määrittelyt on kirjattu tiketille, joten kenen tahansa pitäisi pystyä ymmärtämään tiketin sisältö. Joillain kehittäjillä saattaa myös olla erikoisosaamista tiettyjen osa-alueiden, esimerkiksi testien kirjoittamisen suhteen. Silloin voi pyytää tällaista henkilöä suoraan katselmoimaan omaa työtään.

Katselmointiin valmiit koodit julkaistaan joka tapauksessa tiimimme Slack-kanavalla, josta kuka tahansa tiimimme kehittäjistä voi napata katselmoitavaa. Slack-viesti sisältää linkin tiketille sekä GitHubin pull requestiin. Pull request avataan GitHubiin siinä vaiheessa, kun kehitetty toiminnallisuus on kehittäjän mielestä valmis ja hän on puskenut koodin omaan etäkehityshaaraansa.

Ennen pull requestin avaamista kehittäjän tulee käydä läpi tarkistuslista, jotta ilmiselvät virheet tulevat korjattu ennen katselmointia (Coudouy 4.11.2021; Tiimin sisäinen ohjeistus):

- Onko koodi tyyliltään projektin yleisten sääntöjen mukaista? Ilmoittaako lintteri eli automaattinen koodin laaduntarkastaja virheistä?
- Menevätkö automaattitestit läpi?
- Onko toiminnallisuus sellainen, jolle pitäisi kirjoittaa lisää testejä?
- Pitääkö jotain dokumentaation osaa päivittää uuden toiminnallisuuden myötä?
- Onko koodiin kirjoitettu kommentteja, etenkin vaikeammin ymmärrettävän logiikan yhteyteen?
- Onko muuttujat ja metodit nimetty yhtenäisesti ja ymmärrettävästi?
- Onko konsoliin ilmestynyt uuden toiminnallisuuden myötä virheilmoituksia?
- Onko testausohjeet kirjoitettu pull requestin ja tiketin yhteyteen?

- Onko tiketti linkitetty pull requestille?
- Onko mahdolliset konfliktit master-haaran kanssa korjattu?

Pull request avataan projektin etätietovarastossa GitHubissa. Pull requestin avaamisen yhteydessä GitHub avaa lomakkeen, jolla pull requestille voi antaa otsikon ja kirjoittaa sille testausohjeet. Avatun pull requestin avulla on helppo tarkastella tehtyjä koodimuutoksia, sillä GitHub avaan näkymän, jossa uusi ja vanha koodi ovat vierekkäin. Koodien kohdalle voi jättää kommentteja ja parannusehdotuksia joko tekstin tai koodin muodossa. Pull requestin avaaminen laukaisee myös projektiin liitetyt automaattitestit ja antaa varoituksen, mikäli ne eivät menisi läpi. Mahdolliset merge-konfliktit master-haaran kanssa näkyvät pull requestin yhteydessä. Kun koodin katselmoija on mahdollisten muutospyyntöjen korjausten jälkeen hyväksynyt pull requestin, se voidaan yhdistää master-haaraan. Pull requestin voi avata myös luonnoksena, mikäli kehittäjä haluaa palautetta koodista, mutta toiminnallisuus ei syystä tai toisesta ole vielä valmis yhdistettäväksi master-haaran kanssa. (GitHub 2022.)

3.4 Seurantaviikko 4

Tämän viikon tavoitteena oli selkeyttää ymmärrystä tikettien määrittelyprosessista sekä miten tiketit kulkevat Jira-tehtävienhallintaohjelmistossa.

Maanantai 3.10.2022

Tänään otimme kollegani kanssa yhteyttä viime viikolla ilmenneen tuotanto-ongelman raportoineeseen käyttäjään. Saimme käyttäjältä hyvää lisätietoa ongelmasta, missä tilanteessa se tapahtui ja kuinka usein se on toistunut. Testasimme myös itse sovelluksessa, saisimmeko ongelman toistettua. Tämän onnistuttua loin ongelmasta virhetiketin (jatkossa tiketti kolme), johon tarvittiin kuitenkin vielä ensitietojen lisäksi konseptointia ja designerin panosta. Lähetin keskeneräisen tiketin tuoteomistajalle, joka hoitaisi asiaa eteenpäin.

Tiistai 4.10.2022

Tänään dailyssa kävimme designerin kanssa läpi tiketin kolme käyttäjäongelmaa. Tikettiä varten oli tarkoitus järjestää vielä erillinen määrittelypalaveri, mutta kerroimme designerille jo tässä vaiheessa mistä ongelmassa on kyse, jotta hän ehtii valmistella luonnoksen ratkaisusta palaveria varten. Ongelma liittyi tilauksen luomiseen ja tilausriveille lisättävän tiedon tallentamiseen. Mikäli tallennus epäonnistuisi esimerkiksi verkkovirheen vuoksi, sovellus ei nykyisellään anna käyttäjälle riittävän informatiivista virheviestiä. Designerin haluttiin antavan ehdotus virheviestin ulkoasusta, sekä sen sijainnista sivulla.

Sain tänään myös testattavakseni kaksi kollegan kehittämää tikettiä. Molemmat liittyivät Rest-rajapintoihin, niille tehtäviin kyselyihin parametreineen ja niistä saataviin vastauksiin. Yleensä tikettien testausvaiheen hoitavat tuoteomistajat tai testaajat, mutta varsinkin todella tekniset tiketit tulevat usein kehittäjien testattavaksi.

Keskiviikko 5.10.2022

Tänään paneuduin syvemmin Rest-rajapintojen testaukseen. Kyseessä oli uusi kehitetty rajapinta (jatkossa tiketti neljä), jonka antamaa vastausta vertasin toisen rajapinnan tuottamaan vastaukseen. Rajapinnat oli kehitetty kahden erillisen palvelun päälle, mutta niiden vastausten odotettiin olevan identtiset. Uuden rajapinnan vastauksesta puuttui tärkeitä osia, joita selvittelin pitkän tovin kahden kollegani kanssa. Testaukseen käytin Swagger API-työkalua.

Lisäksi meillä oli palaveri kehitystiimin ja designerin kanssa tikettiin kolme liittyen. Designer oli valmistellut kaksi ehdotusta, joita kävimme läpi. Molemmissa ehdotuksissa designer suositteli virheviestin näyttämistä mahdollisimman lähellä virheen aiheuttajaa, eli esimerkiksi tuoterivin kohdalla, jossa virhe aiheutui. Sovelluksen teknisestä kokonaisuudesta eniten tietävä kollegani ei pitänyt ehdotettuja ratkaisuja teknisesti järkevänä toteuttaa. Kyseessä oleva virhe toistuu tietojemme mukaan harvoin, ja designerin ehdottama ratkaisu olisi vaatinut mittavaa refaktorointia sovelluksen tilan hallintaan. Annoimme designerille vielä lisätietoa toteutettavissa olevista mahdollisuuksista ja päätimme jatkaa keskustelua vielä myöhemmin.

Torstai 6.10.2022

Tänään jatkoin Rest-rajapinnan testaukseen liittyvän tiketin neljä kanssa. Etsin ja dokumentoin rajapinnan kehittäjälle hänen tarvitsemaansa lisätietoa. Tähän meni suurin osa päivästä, sillä tarvittava tieto oli kaivettava projektin koodikannasta, jossa tietoa oli useammassa paikassa.

Testaajamme löysi kehittämästämme sovelluksesta virheen, jonka perusteella hän loi virhetiketin. Virhe oli tällä kertaa nopea ja helppo korjata, joten tein sen alta pois. Päivän päätteeksi meillä oli vielä palaveri liittyen sovelluksen uuteen maahan lanseerauksesta. Palaverissa kävimme läpi käyttäjätapauksia, joilla tulevat käyttäjät tulevat käyttämään sovellusta. Tämä oli erittäin mielenkiintoista ja antoi vähän esimakua mitä kaikkea kehitysjonoon tulevaisuudessa tulisi.

Perjantai 7.10.2022

Tänään kävimme jälleen designerin kanssa läpi alkuviikolla luodun tiketin kolme ratkaisuehdotusta. Designer oli tehnyt kollegani kanssa käydyn keskustelun perusteella uuden ehdotuksen, joka pohjautui enemmän siihen mikä olisi teknisesti helposti mahdollista. Ratkaisussa virheviesti tulisi sivun

ylälaitaan ja sen sisältö olisi informatiivisempi kuin ennen. Designerin mielestä virheviestin olisi ollut hyvä tulla siihen kohtaan sivua, jossa virhe todella tapahtuu, mutta hän ymmärsi teknisen haasteen. Virheen ei myöskään uskottu toistuvan säännöllisesti tai edes kovin usein, joten ratkaisun ei tarvinnut olla niin loppuun saakka hiottu.

Osallistuin tänään lounastauolla Mimmit koodaa -yhteisön webinaariin aiheena tutustuminen design-maailmaan. Webinaaria vetänyt designer kertoi muun muassa siitä, kuinka koodarin ja designerin yhteistyö on tärkeää suunniteltaessa ui-elementtejä, jotta voidaan yhteistyössä keskustella ehdotetuista ratkaisuista niin designin kuin teknisen toteutuksen näkökulmasta (Korpela 10.10.2022, 1:00-1:02 h). Totesin tämän saman tänään käytännössä, pelkällä hienolla suunnitelmalla ei merkitystä, jos sitä ei voida järkevällä työpanoksella toteuttaa.

Myöhemmin täytin vielä tiketille kolme päivitetyt määritykset, sekä lisäsin linkin designeihin. Lisäksi pidimme pienen palaverin kokeneemman kollegani kanssa aiheesta, sillä tämä tiketti nimitettiin minulle. Palaverissa kävimme korkealla tasolla läpi tiketin toteutustapaan liittyviä teknisiä seikkoja ja minulle heti heränneitä kysymyksiä. Tämän kaltainen nopea palaveri on mielestäni hyvä tapa aloittaa kehittämistyöt, sillä se selkeyttää ratkaistavaa ongelmaa ja antaa suuntaviivat mistä lähteä liikkeelle.

Viikkoanalyysi 4

Tämän viikon tavoite täyttyi, sillä työtehtävät liittyivät paljon tikettien luomiseen, määrittelyiden päivittämiseen ja eri tiloissa olevien tikettien etenemiseen Kanban-taululla. Kanban oli minulle tuttu työväline edellisestä työpaikasta sekä opintojen ajalta. Nykyinen tiimini käyttää Kanbania kuitenkin aiempaa kokemustani monipuolisemmin, joten halusin oppia itsekin käyttämään sitä, sekä sen alustana toimivaa Jira-ohjelmistoa sujuvasti.

Tiimissämme käytetään tikettien hallintaan Jira-ohjelmistoa. Jira on Australialaisen Atlassian-yrityksen vuonna 2002 lanseeraama ohjelmisto, jota voi käyttää muun muassa projektinhallintaan, tikettien hallintaan ja service deskin apuna (Varshney 23.11.2021). Kehitystiimimme tarkastelee tikettien etenemistä tiimikohtaisilla Kanban-tauluilla. Kanban-projektityötaulun avulla on mahdollista tarkastella visuaalisesti työn etenemistä työlistalta tiimikohtaisesti määriteltyjen välivaiheiden kautta valmiiseen. Kanban lisää työn läpinäkyvyyttä ja siitä on helposti havaittavissa syntyneet pullonkaulat. (Koskinen 26.3.2021.)

Tiimissämme tikettejä luovat useimmiten tuoteomistajat ja kehityksen johtajat. Myös testaajat luovat tikettejä löytämistään virheistä, sekä jokaisella kehittäjällä on myös oikeus luoda tikettejä tarvittaessa. Jira-järjestelmään on mahdollista luoda monen tyyppisiä tikettejä, joita tiimi käyttää omien tarpeidensa mukaan. Tyypillisesti tikettejä luodaan hierarkkisesti kolmelle eri tasolle. Epic eli eepos

on hierarkian ylimpänä ja kuvaa esimerkiksi jotain suurta kokonaisuutta. Meidän kehitystiimimme tapauksessa tämä voisi olla esimerkiksi “Sovelluksen lanseeraus uuteen maahan”. Eeposten alla on Featuret eli uudet ominaisuudet tai toiminnallisuudet. Meidän tiimimme tapauksessa näitä voisi olla esimerkiksi “Tuotetietojen haku” ja “Asiakastietojen haku”. Uudet ominaisuudet paloitellaan vielä Storeihin eli tarinoihin, jotka ovat kehittäjän kannalta sopivia kokonaisuuksia. Meidän tiimimme tapauksessa näitä voisi olla esimerkiksi “Tuotetietojen rajapinta välittää uuden liiketoiminta-alueen tietoja” ja “Tuotetietojen esitys kaupassa lokaalilla kielellä”. Mikäli testausvaiheessa kehitetyistä Storeista löytyy virheitä, voidaan luoda virhetikettejä. Meidän tiimimme tapauksessa tällainen voisi olla esimerkiksi “Tuotetietojen otsikoista kohdasta hinta puuttuu käännös”. (Atlassian 2022a.)

Tiketin luomisen jälkeen tuoteomistaja tarkistaa sen tiedot ja lisää muun muassa erilaisiin raportointeihin tarvittavat teemat ja merkinnät. Taulukossa 1 on kuvattu tiketin workflow eli elinkaari Kanban-taululla eri tiloineen ennen valmistumista. Jirasta löytyy valmiina paljon eri tiloja ja niitä voidaan myös luoda tiimin tarpeen mukaan lisää (Atlassian 2022b). Tuoteomistaja merkitsee tiketin “valmiina kehitettäväksi”, jonka jälkeen tiketti tulee näkyville kehittäjien Kanban-taululle ja sen voi ottaa työn alle. Kaikki käynnissä olevat kehitystyöt ovat tilassa “käynnissä” ja valmistuttuaan ne siirtyvät tilaan “katselmoitavana”. Kun katselmointi on hyväksytty, tiketti mergetään projektin master-haaran kanssa ja kehitetty ominaisuus tulee näkyviin sovelluksen testiympäristöön. Megetyt tiketit asetetaan tilaan “valmiina testattavaksi” josta testaajat ottavat tiketit työn alle “testattavana” tilassa. Kun testaaja on hyväksynyt tiketin, siirtyy se hyväksymistestaukseen liiketoiminnan edustajille. Kun liiketoiminnan edustajat ovat hyväksyneet muutokset tiketti asetetaan tilaan “valmiina julkaistavaksi” odottamaan seuraavaa julkaisuajankohtaa. Kun julkaisu on saatu tehtyä tiketti suljetaan asettamalla se tilaan “valmis”.

Taulukko 1. Tiketin elinkaari Kanban-taululla

Valmiina kehitet-täväksi	Käynnissä	Katselmoitavana	Katselmointi hyväksytty	Valmiina testat-tavaksi	Testattavana	Hyväksymis-testaus	Hyväksymis-testaus hyväksytty	Valmiina julkais-tavaksi	Valmis

3.5 Seurantaviikko 5

Tämän viikon tavoitteena oli syventää osaamista frontend-kehityksen osa-alueilla ja saavuttaa ymmärrystä keinoista ratkaista erilaisia sivun asemointiin liittyviä ongelmia.

Maanantai 10.10.2022

Tänään pääsin kunnolla työstämään tikettiä kolme. Ensimmäinen ratkaistava ongelma oli virheviestin siirtäminen sivun alalaidasta näkyvämmälle paikalle navigointivalikon alle. Siirto itsessään onnistui helposti, mutta ongelmia aiheutti alla olevien komponenttien asemointi sivulla. Virheviestejä on mahdollista olla useita päällekkäin, ja niiden alla olevien komponenttien tulisi tässä tapauksessa siirtyä responsiivisesti alaspäin. Aiemmin komponenttien asemointiin käytetty laskenta oli kovakoodattu, sillä sivun ylälaidassa olevat elementit olivat aina saman korkuisia. Pyörittelin erilaisia ratkaisuja käyttäen hyödyksi selaimen kehittäjän työkaluja, jossa CSS-määrityksiä pääsee muokkaamaan reaaliaikaisesti, eikä näin ollen tarvitse koskea varsinaiseen koodiin lainkaan. Lopulta jouduin kysymään apua kollegalta, mutta emme vielä saaneet ratkaistua ongelmaa.

Tiistai 11.10.2022

Tänään keskityin edelleen tiketin kolme työstämiseen. Muokkasin virheviestin manuaalisesti suljettavaksi ja tarkistin että virheviesti-komponentin CSS-määritykset vastasivat designeissa määriteltyjä. Virheviestin lisääminen ja poistaminen aiheuttivat edelleen ongelmia niiden alla sijaitsevien komponenttien responsiiviseen asemointiin sivulla. Syyksi paljastui lopulta Reactin renderöintijärjestys, eli missä järjestyksessä React piirtää komponentit sivulle. Lähdin kokeilemaan Reactin useEffect-hookkia, jonka avulla voidaan laukaista komponenttien uudelleen renderöinti, kun määritelty tila muuttuu. Tässä tapauksessa halusin laukaista uudelleen renderöinnin, kun virheviestien määrä muuttuisi. Tämä toimi, kun virheviestejä lisättiin, mutta ei silloin kun virheviestejä poistettiin. En saanut ongelmaa ratkaistua tämän päivän aikana.

Keskiviikko 12.10.2022

Tänään jatkoin edelleen tiketin kolme parissa. Kokeneemman kollegan kehotuksesta luovuin sittenkin yrityksestä manipuloida Reactin renderöintijärjestystä. Hän oli myös etsinyt tietoa aiheesta, eikä ollut keksinyt järkevää tapaa toteuttaa sitä. Komponenttien asemointiin vaikuttava laskenta päätettiin toteuttaa manuaalisesti, jolloin CSS-määrityksiä varten saatiin varmasti oikea arvo. Tämä ei kollegan mukaan ollut paras ratkaisu, mutta miettisimme sitä myöhemmin vielä yhdessä. Sain lisämäärityksenä tiketille pyynnön lisätä aikaleiman virheviestiin. Tämän lisääminen muutti viestin sisällön kuitenkin jo melko pitkäksi ja vaikeasti hahmotettavaksi. Otin yhteyttä designeriin, jonka kanssa yhdessä mietimme viestin sisällölle selkeän rakenteen.

Näiden lisäksi tein koodikatselmoinnin virheestä, jonka testaajamme löysi ja kollegani korjasi. Samalla opettelin git stashin käytön. Git stash on Git-versiohallintatyökalun komento, jolla omassa kehityshaarassa tehdyt muutokset saadaan talteen, ilman että niitä pusketaan versionhallinnan etäkehityshaaralle. Tämä oli tarpeellinen taito, jonka olin halunnut oppia jo jonkin aikaa.

Torstai 13.10.2022

Tänään jatkoin tiketin kolme työstämistä. Muokkasin aikaleiman käyttäjälle näytettävää muotoa yhtenäiseksi sovelluksessa muuallakin käytetyn tavan kanssa. Päivämäärän ja kellonajan lisäksi kellonaikaan haluttiin näkyviin sekunnit, jotta tapahtunut virhe olisi helppo löytää lokeista. Virheviesteissä oli toinen Reactin renderöintiin liittyvä ongelma, jonka vuoksi ne tulivat toisinaan näytölle tuplana. Ongelman ratkaisun etsiminen vei enemmän aikaa kuin odotin, mutta kokeneemman kollegan avustuksella sain sen ratkaistua. Nyt viestit suodatettiin niin sisällön kuin aikaleiman perusteella, jotta kahta samansisältöistä ja riittävän lähellä toisiaan ajallisesti olevaa viestiä ei päätyisi ruudulle.

Kehittämämme sovellus oli tänään testaajalla perusteellisessa testauksessa, sillä muutaman kehitetyn toiminnallisuuden julkaisu lähestyi. Testaaja on todella hyvä löytämään virheitä ja autoin ison osan päivästä katselmoimaan ja testaamaan kollegani tekemiä korjauksia niihin.

Perjantai 14.10.2022

Testaaja oli löytänyt uusia virheitä, jotka odottivat aamulla työjonossa. Ryhdyin tutkimaan niistä yhtä (jatkossa tiketti viisi). Virhetiketit kiilaavat yleensä tärkeysjärjestyksessä kehitysjonon kärkeen, sillä virheiden korjaamisella saattaa olla kriittinen merkitys sovellusversion julkaisuajankohtaan. Sovelluksessa oli ongelmana, että vieritettäessä sivun navigointivalikkoa, sen ruudun ulkopuolelle jäänyt osa hävitti jostain syystä taustavärinsä. Navigointivalikko muuttui vieritettäväksi, mikäli sen alla olevassa palkissa oli auki liian monta ostoskoria ruudun kokoon nähden. Tunnistin, että ongelma ei ollut uusi, eikä näin ollen kriittinen tulevan julkaisun kannalta. Tein muutamia kokeiluja CSS-määrityksiä muuttamalla, mutta en saanut virhettä suoraan ratkaistua.

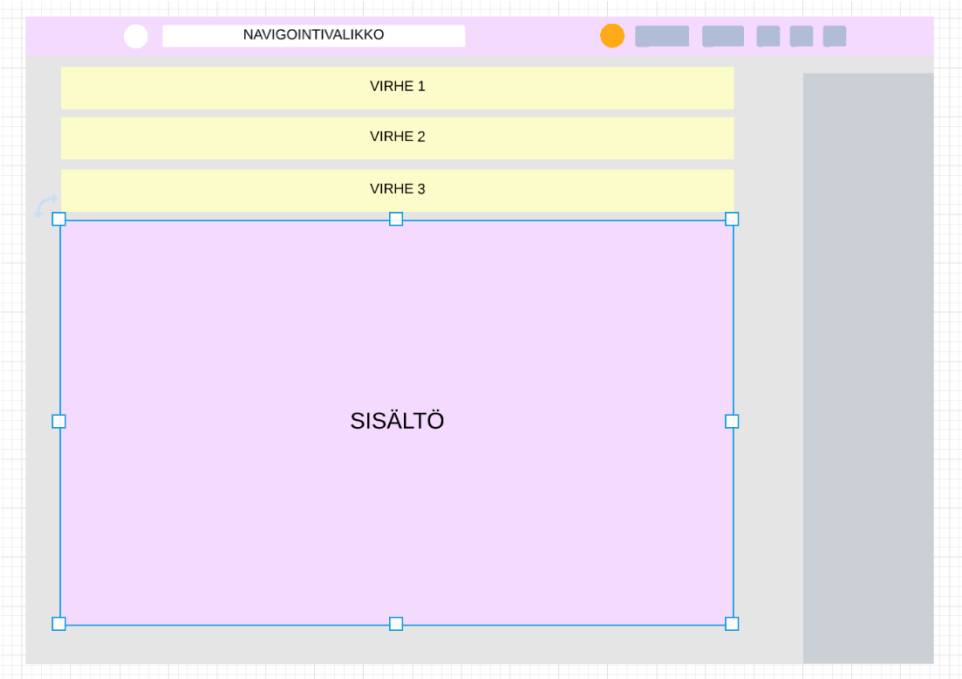
Jatkoin myös hieman tikettiä kolme ja sain kirjoitettua komponenttikohtaisesti räätälöidyt virheviestit aikaisemman geneerisen virheviestin tilalle. Tätä oli toivottu tiketin määrittelyn yhteydessä, jotta virheviesti antaisi enemmän informaatiota niin käyttäjälle kuin virhettä selvittävälle kehittäjälle.

Viikkoanalyysi 5

Tämän viikon tavoite täyttyi osittain, sillä tein suurimmaksi osaksi frontend-kehitystä ja kehityin joillain osa-alueilla. Pääsin syventämään Reactin ja CSS:n lisäksi versionhallinnan osaamistani.

Ensimmäinen opiskelemani asia oli Reactin renderöintijärjestys. React piirtää sovelluksen komponentit ruudulle, kun komponentti ladataan ensimmäistä kertaa tai kun komponenttien esittämässä datassa tapahtuu muutoksia (Makarevich 2.8.2021). Kuva 5 sivulla 29 esittää mukaelman

sovelluksen ulkoasusta. Kun sovellus ladataan ensimmäisen kerran, vaaleanpunaisella esitetyt navigointivalikko sekä sovelluksen sisältö piirtyvät ruudulle. Virheviestin paikka on esitetty keltaisella ja niitä on mahdollista olla useita allekkain. Navigointivalikko pysyy aina näkyvillä ja on tietyn korkuinen. Alla olevan sisällön paikka ruudulla on määritelty käyttäen hyödyksi navigointivalikon korkeutta, asettamalla sisällölle CSS-määritys “top”. Top-määritys asettaa komponentin yläpuolelle annetun pikselimäärän verran tilaa ulomman komponentin yläreunasta (W3Schools s.a.b). Kehittämieni virheviestien myötä tuo top-pikselimäärä olisi vaihdellut aina virheviestien lukumäärästä riippuen ja vaatinut lisää kovakoodattua laskentaa.



Kuva 5. Kehitettävän sovelluksen viitteellinen ulkoasu

Seuraavaksi opin lisää Reactin hookeista, jotka ovat funktioita, joilla voidaan vaikuttaa muun muassa React-sovelluksen tilaan (Meta Platforms 2022a). Reactin useRef-hookin avulla voidaan lukea suoraan DOMista sisällön yläpuolinen korkeus, jolloin manuaalisesta korkeuden laskennasta voitaisiin luopua. DOM eli document object model on standardi, jolla sovellusten dokumenttipuuta, eli HTML tai XML-elementtejä voidaan tarkastella ja muokata (Mozilla s.a.a). Reactin useEffect-hookin sisälle määritelty toiminnallisuus laukaistaan aina kun komponentti on renderöity, tai vain kun sille annettujen parametrien arvot muuttuvat (Meta Platforms 2022b). Pyrkimyksenäni oli saada sovellus toimimaan seuraavassa järjestyksessä:

1. Sovellus latautuu, useRef lukee sovelluksen yläosan (navigointivalikon) korkeuden ja asettaa luvun sisällön top-määritykselle.
2. Virhe tapahtuu, virheviesti piirtyy navigointivalikon alle.

3. Sovellus latautuu uudelleen ja koska sovelluksen yläosan (navigointivalikko + virheviesti) korkeus on muuttunut, useEffect asettaa useRef-hookilla luetun korkeuden sisällön top-määritykselle.

Yllä kuvattu lähestymistapa toimi hyvin, kun virheviestejä lisättiin ruudulle, mutta toistaiseksi tuntemattomasta syystä ei silloin kun niitä suljettiin. Kokeneempi kollegani arveli syyn olevan Reactin renderöintijärjestyksessä, mutta en löytänyt ongelmaan ratkaisua tämän viikon aikana.

Kolmantena uutena asiana tällä viikolla opiskelin lisää CSS-määrityksien käyttöä. Ongelmanani oli virheviestille annetut määritykset, joiden tuli olla voimassa vain silloin kun virheviesti näkyi ruudulla. Opin käyttämään :not:empty –selektoria, jolla ilmaistiin, että komponentin CSS-määritykset olisivat voimassa vain silloin kun sillä on sisältöä (Kravets 1.8.2018).

Lopuksi opin vielä uuden taidon versionhallin käytöstä. Välillä vastaan saattaa tulla tilanne, jolloin työstää omaa koodia, mutta ei ole valmis puskemaan sitä etäkehityshaaralle. Väliin saattaa kiilata jokin kiireellinen virhekorjaus tai toisen kehittäjän koodin katselmointi. Tällöin oma koodi on saatava jotenkin talteen puskematta sitä etäkehityshaaralle, jotta voi jatkaa samasta pisteestä kiireellisemmän tehtävän valmistuttua. Tällaisissa tilanteissa voi käyttää versiohallinnan komentoa git stash, joka säilöö keskeneräiset muutokset. Säilötyt muutokset saadaan takaisin näkyviin komennolla git stash pop. (Helsingin yliopisto s.a.)

3.6 Seurantaviikko 6

Tämän viikon tavoitteena oli syventää ymmärrystä frontend-kehitykseen ja koodin luettavuuteen liittyen. Tavoitteena oli myös osallistua tiimin yhteiseen viestintään ja oppia lisää ammattimaisen kehittäjän roolista.

Maanantai 17.10.2022

Tänään jatkoin perjantaina aloittamani tiketin viisi tutkimusta. Tein edelleen kokeiluja sovelluksen CSS-määrityksiä muuttamalla selaimen kehittäjän työkalujen kautta. Samalla havaitsin, että sovellus ei skaalautunut kovin hyvin eri koikoisille ruuduille. Sovelluksella ei ole vaatimusta olla yhteensopiva mobiililaitteiden kanssa, mutta tiedämme että sitä käytetään niillä kuitenkin jonkin verran. Päätin ottaa tuoteomistajan kanssa puheeksi pitäisikö sovellukseen tehdä isompi refaktorointi, jotta sovelluksesta saataisiin nykyistä responsiivisempi. Esimerkiksi navigointivalikkoon voitaisiin ottaa kapealla ruudulla käyttöön niin sanottu hamburger-menu, jonka taakse navigointivaihtoehdot menisivät. Näin ollen ei olisi tarvetta vierittää navigointipalkkia, vaan se olisi responsiivinen ruudun koon mukaan.

Tiistai 18.10.2022

Tänään jatkoin tiketin kolme parissa. Tarkastelimme kokeneemman kollegan kanssa designerin näkemystä virheviestin ulkoasusta komponentin sisällä olevan tyhjän tilan ja tekstin välistysten suhteen. Huomasimme että virheviestin ulkoasu ei mukaillut muun sovelluksen ulkoasua. Otin yhteyttä designeriin ja havaitsimme, että hänen ehdotuksensa perustui Figma-suunnittelutyökalusta löytyneisiin versioihin sovelluksessa käytetyistä komponenteista, jotka eivät lähemmin tarkasteltuna olleet enää ajan tasalla. Korjasimme yhdessä suunnitelman sopivaksi ja muutin komponentit toteutukseeni päivittyneen suunnitelman mukaisiksi.

Pyysin tuoteomistajalta tiketille kolme tarvittavat käännökset eri kieliversioille. Lisäksi päätin vielä refaktoroida komponenttirakennetta, sillä mielestäni koodi alkoi olla vaikeasti ymmärrettävää. Sovelluksessa esitettiin aiemmin sekä virhe- että infoviestit saman komponentin kautta tietyssä kohtaa sivulla. Tekemäni muutoksen myötä virheviestit siirrettiin sivun ylälaitaan, mutta infoviestit jäivät edelleen vanhaan kohtaan sivua. Molemmat viestityypit käyttivät samaa Redux-tilanhallintakirjaston avulla luotua tilaa, eli ne kaikki tallentuivat samaan taulukkoon, mutta niiden ulkoasu ja asemointi sivulla oli täysin eri. Näin ollen päädyin erottelemaan erityyppiset viestit omiin komponentteihinsa, jotta välttyisin liian monimutkaisilta ehtoihin perustuvilta esitysmuodoilta.

Keskiviikko 19.10.2022

Tänään sain melkein viimeisteltyä tiketin kolme. Siirsin vielä muutamia useamman komponentin yhteisessä käytössä olevia koodeja aputiedostoon. Näin sain siivottua koodista pois toisteisuutta ja siitä tuli helpommin luettavaa. Refaktoroin myös muuttujien ja funktioiden nimiä kuvaamaan niiden todellista tehtävää. Lopuksi ajoin vielä testit ja niiden mentyä läpi puskin koodit versionhallinnan etäkehityshaaralle. Seuraavalle päivälle jäisi vielä koodin rebasetus lokaalin ajantasaisen masterhaaran kanssa, pull requestin avaaminen ja testausohjeiden kirjoittaminen.

Meillä oli tänään kuukausittainen kehittäjien yhteinen ylläpitotoimiin keskittyvä palaveri. Näiden palaverien tarkoitus oli päättää yhdessä ylläpito- ja refaktorointitoimien tärkeysjärjestys eri sovelluksissa. Tässä, kuten edellisessäkään vastaavassa palaverissa, osaamiseni tällä saralla ei riittänyt ottamaan kantaa esitettyihin asioihin. En kuitenkaan potenut tästä huonoa mieltä, sillä useimmat käsiteltävät asiat vaativat selvästi enemmän kokemusta kehittäjänä, sekä tietämystä muun muassa pilvipalveluiden asetuksiin liittyen. Ilokseni huomasin kuitenkin, että ymmärsin käsiteltävistä asioista jo huomattavasti enemmän, kuin edellisessä kokouksessa. Olisin voinut ottaa jopa jotain ylläpitotoimista työn alle. Palaverin jälkeen ymmärsin entistä paremmin, että ammattimaisen kehittäjän täytyy ymmärtää koodin lisäksi myös alustaa ja ympäristöä, jossa se pyörii, sekä ymmärtää projektiin liittyvät riippuvuudet ja tietoturvaseikat.

Torstai 20.10.2022

Tänään viimeistelin lisää tikettiä kolme. Latasin lokaalisti ajantasaisen master-haaran ja yhdistin omat koodini siihen. Sitten puskin koodin vielä kertaalleen etäkehityshaaralle ja avasin pull requestin. Pull requestin yhteyteen kirjoitin testausohjeet kehittäjän näkökulmasta ja tiketille taas testaajan ja tuoteomistajan näkökulmasta.

Tänään meillä oli lisäksi koko tiimin yhteinen Mid PI review-kokous. Kokouksen aloittivat liiketoiminnan edustajat eri maista. He kertoivat kuluneen syksyn tapahtumista ja taloudellisista luvuista. Kehittäjän kannalta kiinnostavin asia oli loppukäyttäjien palaute uusista kehitetyistä toiminnallisuuksista. Mikäli varsinaista palautetta ei ollut saatavilla, saimme kuitenkin nähdä analytiikkaan toiminnallisuuksien käyttöasteesta ja niiden tuomasta kassavirrasta.

Mid PI review-kokous sisälsi myös tuoteomistajien puheenvuoron, jossa käytiin läpi kunkin kehitystiimin PI-suunnitelman mukaisten kehitystöiden tilannetta. Näin kaikki tiimin jäsenet saivat näkyvyyttä, mitä mahdollisesti julkaistaan lähiaikoina ja mitä taas lähempänä PI:n loppua.

Perjantai 21.10.2022

Kollegani oli katselmoinut tiketin kolme koodini ja jättänyt muutosehdotuksia. Ilokseni muutosehdotukset olivat hyvin pieniä seikkoja, kuten virheviestin sisällön pientä viilausta. Varsinaista koodiin liittyvää huomautettavaa ei kerrankin ollut löytynyt! Kollegani oli kuitenkin keksinyt ratkaisun Reactin renderöintijärjestykseen liittyvään ongelmaan, jota olimme yhdessä yrittäneet ratkaista jo edellisellä viikolla. Virheviestin näyttämisessä käytettiin aikaisemmin pientä animaatiota, joka toi virheviestin näytölle voimistuen. Tämä loi virheviestin esittämiselle juuri sen mittaisen viiveen, että renderöintijärjestys meni sekaisin, joten päätimme poistaa animoinnin. Muutos oli hyvin pieni käyttäjän näkökulmasta ja virheviesti ilmestyi ruudulle edelleen siististi. Tällä muutoksella saatiin kuitenkin koodia siivottua ja yksinkertaistettua, sekä nyt koodi toimi viikkoanalyysissä 5 sivulla 29 esitetyn suunnitelman mukaisesti. Tein viimeiset ehdotusten mukaiset korjaukset, jonka jälkeen tiketin kolme pull request hyväksyttiin.

Viikkoanalyysi 6

Tämä viikon tavoitteet täyttyivät. Opin lisää sovelluksen responsiivisuudesta ja yhtenäisestä ulkoasusta, sekä opin myös lisää luettavan ja laadukkaan koodin tuottamisesta. Lisäksi osallistuin tiimini yhteisiin palavereihin, joissa käsiteltiin kehitysympäristöjen ajan tasalla pitämistä sekä kehitystöiden etenemistä PI:n alkupuolella.

Sovellustiimin kehittämä sovellus toimii välttävästi mobiililaitteilla, mutta havaitsin skaalautuvuudessa jonkin verran parannettavaa. Opiskelin responsiivisen kehittämisen periaatteita tarjotakseni tuoteomistajalle lisätietoa mahdollisuuksista ja ymmärtääkseni millaisen refaktorointityön muutokset vaatisivat. Responsiivinen sovellus toimii eri kokoisilla laitteilla sujuvasti. Yhä useampi käyttää sovelluksia mobiililaitteilla, joten responsiivisuutta osataan myös vaatia. Responsiivisuus voidaan saavuttaa muun muassa määrittelemällä erilaisia tyylimäärityksiä eri kokoisille näytöille. Responsiivinen sovellus käyttää sisällön esittämiseen näytöllä käytettävissä olevaa tilaa ja osa sisällöstä voidaan kapealla näytöllä piilottaa erilaisten painikkeiden alle. (Interaction Design Foundation 2022.) Yksi esimerkki tästä on usein mobiilisivuja selatessa navigointivalikoissa nähtävä hampurilaismenu, jonka taakse voidaan piilottaa navigointipalkin linkkejä, kuten kielivalinta tai uloskirjautuminen (Babich 26.5.2021).

Kehittäjien kuukausittaisessa palaverissa käsiteltiin päivitystä vaativia kirjastoja eri sovelluksissa ja teknisen velan paikkaamista. Tekninen velka on yleiskäsite sovelluksiin liittyvistä asioista, jotka vaikeuttavat sovelluksen jatkokehitystä tai ylläpitoa (Lehojärvi 12.2.2017). SAFe:n viitekehyksen kanssa työskentely tarkoittaa muun muassa ketterän ohjelmistokehityksen julistuksen periaatteiden omaksumista (Scaled Agile 27.9.2021). Ketterän ohjelmistokehityksen julistuksessa yksi periaatteista on “Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.” (Beck ym. 2001). Päivitettävien kohteiden tunnistaminen oli tärkeää, jotta välttämättömät ylläpitotyöt saadaan priorisoitua ennen uusien kehitystöiden aloitusta.

Sovelluksissa hyödynnetään kehityskielestä riippumatta avoimen lähdekoodin kirjastoja, joilla voidaan ratkoa erilaisia ongelmia ja säästää työmäärässä. Nämä käytetyt kirjastot saattavat puolestaan edelleen hyödyntää muita avoimen lähdekoodin kirjastoja. Kirjastoja kehitetään edelleen ja niihin saattaa tulla uusia ominaisuuksia tai jokin tietoturvan liittyvä virhe saatetaan korjata. Tästä seuraa tarve päivittää kirjastosta uusi versio sovellukseen ja kun kirjastoja on käytössä useita, saattavat päivitykset kasaantua. Usein päivitys tarkoittaa vain kirjaston versionumeron päivittämistä, mutta joissain tapauksissa päivitys saattaa vaatia sovelluksen mittavampaa refaktorointia. (Heikniemi syyskuu 2022.) Tiimin sovelluksissa oli muutama tällainen isompi päivitys tiedossa ja yhdessä sovittiin mihin sovellukseen päivitykset tehtäisiin ensimmäisenä. Kun mahdolliset päivityksen aiheuttamat ongelmat on ratkaistu yhdessä sovelluksessa, on päivitys sen jälkeen helpompi tehdä myös muihin sovelluksiin.

Sivusin luettavan ja laadukkaan koodin periaatteita viikkoanalyysissä 3 sivulla 22, listatessani seikkoja, joita kehittäjän tulee tarkistaa ennen pull requestin avaamista. Luettavan ja laadukkaan koodin tuottaminen on yksi tärkeimmistä oppimistavoitteistani ja opinkin siitä lisää tällä viikolla. Avasin yhdestä tiketistä pull requestin enkä saanut yhtään luettavuuteen liittyvää muutosehdotusta, joita

olin aiemmin saanut useita. Aloittelevana koodarina aloitan koodin kirjoittamisen usein niin että saan siitä ensin toimivaa ja vasta sen jälkeen kiinnitän huomiota luettavuuteen ja laatuun. Pyrkimyksenäni on parantaa tehokkuuttani jatkossa opettelemalla laadukkaan ja luettavan koodin periaatteet, jotta osaan tuottaa hyvää koodia heti alusta alkaen.

Luettava ja laadukas koodi on tärkeää ohjelmointikehityksessä, sillä koodia käytetään kommunikointivälineenä ihmisen ja tietokoneen välillä. Tietokone ei kuitenkaan välitä koodin ulkoasusta vaan luettavaa ja laadukasta koodia kirjoitetaan ihmisiä varten. Useimpia sovelluksia työstetään tiimeissä muiden kehittäjien kanssa, joten kaikkien tulee ymmärtää kirjoitettua koodia. Luettavaa ja laadukasta koodia on helppo ymmärtää vielä vuosienkin päästä, joka mahdollistaa sovelluksen sujuvan jatkokehityksen ja päivityksen. (Oikarinen 27.5.2020.)

Luettavan ja laadukkaan koodin periaatteita löytyy netistä ja kirjallisuudesta paljon. Mayer on listannut 17 periaatetta kirjassaan, jotka tiivistäen ovat (Mayer 2022, luku 4):

- Mieti sovelluksen arkkitehtuuria. Ovatko kaikki komponentit ja tiedostot tarpeellisia tiedostorakenne selkeä?
- Voisiko sovellukseen ottaa käyttöön jonkin valmiin kirjaston, jolla osa itse rakennetuista toiminnallisuuksista saataisiin korvattua?
- Koodia kirjoitetaan ihmisten luettavaksi, joten noudata projektin käytäntöjä muun muassa sisennysten ja koodirivien pituuden suhteen. Noudata myös ohjelmointikielikohtaisia sisennyksiä ja tyyliperiaatteita.
- Nimeä muuttujat ja funktiot kuvaavasti sekä noudata ohjelmointikielikohtaista tyyliä.
- Selvennä monimutkaisia toiminnallisuuksia kommenteilla, mutta pyri kuitenkin itsensä selittävään koodiin esimerkiksi nimeämällä funktiot riittävän kuvaavasti. Poista myös vanhat koodit projektista, älä kommentoi niitä piiloon varmuuden vuoksi.
- Poista turha toisteisuus koodista luomalla uudelleen käytettäviä funktioita.
- Funktioiden tulee noudattaa yhden toiminnan periaatetta ja ne tulee nimetä sen mukaisesti. Monimutkaista toiminnallisuutta on helpompi seurata, kun se koostuu useammasta pienestä funktiosta eikä kaikkea toiminnallisuutta yritetä tehdä yhden ison funktion sisällä.
- Kirjoita testejä. Testien avulla voidaan varmistua, että uusi toiminaalisuus ei riko olemassa olevia toiminnallisuuksia.
- Älä kirjoita turhaa koodia, jota esimerkiksi arvelet tarvitsevasti seuraavassa tiketissä.
- Käytä koodieditorin työkaluja havaitaksesi liian monimutkaista koodia. Älä luo koodiluokkien välille turhia riippuvuuksia.
- Korjaa eli refaktoroi koodia saamasi palautteen perusteella tai jos löydät vanhaa huonoa koodia.

Koodi selkeyteen ja luettavuuteen liittyen opin tällä viikolla käyttämään tiivistä esitystapaa, eli ternary-operaattoria. Kuvassa 6 on havainnollistettu, kuinka operaattoria käyttämällä saadaan vähennettyä koodirivejä viidestä yhteen. Kyseistä operaattoria tulee käyttää harkiten ja jos sen käyttö tekee koodista vaikeasti luettavaa, on parempi käyttää if/else-rakennetta (Jraleman 7.1.2018).

```
var ohje;
if(vihreä){
  ohje = "mene"
}else{
  ohje = "pysähdy"
}

-> var ohje = vihreä ? "mene" : "pysähdy";
```

Kuva 6. Koodin tiivistäminen

3.7 Seurantaviikko 7

Tämän viikon tavoitteena oli ymmärtää CSS-määritysten muutosten vaikutus sovelluksen ulkoasuun eri selaimilla ja eri käyttöjärjestelmillä.

Maanantai 24.10.2022

Tänään keskustelin tuoteomistajan kanssa tiketin viisi vaihtoehdoista, olisiko sovellusta syytä refaktoroida enemmän responsiiviseksi vai hoitaisimmeko havaitun ongelman muulla tavalla. Tuoteomistajan mukaan emme lähtisi tässä vaiheessa optimoimaan sovellusta enempää mobiililaitteiden kanssa yhteensopivaksi, sillä pyyntö tälle ei ollut tullut liiketoiminnan taholta.

Jatkoin tiketin viisi kehitystä ja sain siihen myös vaatimukset täyttävän ratkaisun. Päädyin lopulta muuttamaan auki olevien ostoskorien esittävän palkin vieritettäväksi, mikäli ostoskoreja olisi auki enemmän kuin ruudulle mahtuisi. Näin ollen sivun navigaatiopalkki pysyisi joka tilanteessa paikoillaan, eikä menettäisi taustaväriään.

Tiistai 25.10.2022

Sain tikettiin viisi kehittämääni ratkaisuun palautetta. Ratkaisu toimi klassiseen tyyliin hienosti omalla koneella. Kollegani koneella oli kuitenkin päällä helppokäyttötoimintoihin lukeutuva asetus, joka piti kaikki saatavilla olevat vierityspalkit aina näkyvillä. Näkyvissä oleva vierityspalkki peitti ostoskoreja esittävän palkin sisältöä inhottavasti. Omalla Mac-koneellani vierityspalkki katosi aina näkyvistä, kun sitä ei aktiivisesti käytetty. Yritin ratkaista näkymäongelmaa ja törmäsin myös selainkohtaisiin eroihin. Päätin testata näkymän vielä varmuudeksi Windows-käyttöjärjestelmällä, sillä

sovellusta käytetään tietojemme mukaan useimmiten sillä. Havaitsin, että Windows-koneilla vierityspalkit olivat aina näkyvissä, vaikka järjestelmäasetuksista ei oltu erikseen laitettu päälle helppokäyttötoimintoa. Keskustelin vaihtoehdoista tuoteomistajan kanssa. Kaikilla alustoilla toimiva ratkaisu olisi edelleen vaatinut suurempaa refaktorointia, joten ratkaisu päätettiin hyväksyä sellaisenaan.

Keskiviikko 26.10.2022

Tänään jatkoin aiemmin aloittamaani tiketin neljä rajapinnan testausta, kun sen parissa työskentelevä kehittäjä oli saanut kehitystä eteenpäin. Testaus eteni hyvin ja vei suurimman osan päivästäni.

Torstai 27.10.2022

Olin käyttänyt useamman viikon aikana aina silloin tällöin aikaa tiketin neljä puitteissa luodun uuden rajapinnan testaukseen ja rajapinnan kehittäjän kanssa kommunikointiin. Ehdotinkin tuoteomistajalle, että kirjoittaisin rajapinnalle kunnolliset integraatiotestit, jotta voisimme olla myös jatkossa varmoja, että rajapinta toimii odotetulla tavalla. Rajapinta liittyi tuotteiden hinnoitteluun ja sen arvoja tuli verrata toisen rajapinnan palauttaviin arvoihin. Loin integraatiotestien kirjoittamiselle uuden tiketin (jatkossa tiketti kuusi). Kirjoitin tiketille vaatimusmäärittelyt ja rajaukset, jossa otin huomioon jo olemassa olevat aiheeseen liittyvät integraatiotestit, ja kuinka niitä tulisi laajentaa. Lisäsin tiketille lisäksi työmääräarvion työpäivissä tekemällä itselleni konkreettisen tehtävälistan ja miettimällä paljon sen toteutukseen arviolta menisi aikaa. Tehtävälistan ja arvion sain kasaan tutkimalla koodia, uusia tarvittavia datarakenteita ja suunnittelemalla uusia tarvittavia testitapauksia.

Perjantai 28.10.2022

Tänään lähdin työstämään tiketin numero kuusi integraatiotestejä. Ensin laajensin olemassa olevaa datarakennetta ottamaan huomioon uudet testattavat kentät, sitten uudelle rajapinnalle tehtävää kyselyä. Alkuun ajoin uutta rajapintaa vasten olemassa olevat testit ja varmistin että kaikki menevät läpi, sitten lähdin laajentamaan itse testejä vertailemaan myös datarakenteen uusia kenttiä. Etenin kenttä kerrallaan vertaillen kahden rajapinnan palauttamaa vastausta ja löysinkin metodillani heti eroavaisuuksia. Pyysin rajapinnan kehittäjää muun muassa korjaamaan päivämäärän esitysmuotoa. Uutta rajapintaa kehitettiin eri kehitystiimin toimesta SAP:in puolelle, joten kaikkia eroavaisuuksia ei ollut niin yksiselitteistä saada vastaamaan vertailtavan rajapinnan Java-koodilla tuotettuja arvoja.

Viikkoanalyysi 7

Tämän viikon tavoite täyttyi osittain, sillä opin yhden CSS-määrityksen vaikutuksen sovelluksen ulkoasuun eri käyttöjärjestelmissä ja selaimilla. Jatkoa varten päätin opiskella lisää vastaavista määrityksistä ja kerrata kokeneempien kollegoiden kanssa sovelluksen ulkoasuun liittyvien muutosten testaamisen käytännöt.

Ratkaistava ongelma liittyi ostoskoreja esittävän elementin yhteyteen liitettyyn vierityspalkkiin. Elementti asetettiin vieritettäväksi silloin kun sen sisältö ylitti käytettävissä olevan tilan määrittämällä CSS-määritys overflow:auto (Mozilla s.a.b). Riippuen selaimesta, käyttöjärjestelmästä ja oliko käyttöjärjestelmän helppokäyttötoiminnot kytketty päälle, vierityspalkki näytti erilaiselta. Mikäli vierityspalkki oli jatkuvasti näkyvissä, se peitti ostoskoreihin liittyvää tietoa ikävästi. Kollegan kehotuksesta yritin jonkin aikaa etsiä erilaisia CSS-määrityksiä, joilla vierityspalkki saataisiin näyttämään mahdollisimman hyvältä kaikilla alustoilla.

Mac-käyttöjärjestelmä piilottaa vierityspalkit automaattisesti silloin kun niitä ei käytetä, jota esimerkiksi Windows-käyttöjärjestelmä ei tee. Saadakseen todenmukaisemman kuvan sovelluksen ulkoasusta, Mac-käyttöjärjestelmällä kehittävän kehittäjän kannattaa asettaa vierityspalkit aina näkyvillä oleviksi (Valkhof 4.1.2021). Kun vierityspalkki on näkyvillä, sen ulkoasuun voi vaikuttaa esimerkiksi scrollbar-width CSS-määrityksellä. Tällä määrityksellä vierityspalkista voidaan tehdä ohuempi tai se voidaan asettaa läpinäkyväksi. Asetuksen ongelma on, että se toimii ainoastaan Firefox-selaimella. (Mozilla s.a.c.) Sovelluksen elementeille voidaan määrittää tällaisia eri selaimissa toimivia CSS-määrityksiä, mutta niille tulisi aina määrittää myös vaihtoehto, joka toimii muilla selaimilla. Eri selaimille tehdyt määritykset voivat tehdä koodista vaikeasti luettavaa ja päivitettävää, joten niitä tulisi mahdollisuuksien mukaan välttää. (Mozilla s.a.d.)

3.8 Seurantaviikko 8

Tämän viikon tavoitteena oli toimia tiimin hyödyllisenä jäsenenä, sekä oppia lisää luettavan koodin tuottamisesta.

Maanantai 31.10.2022

Tällä viikolla oli vuoroni olla kehittäjien first responder, eli joka esimerkiksi tuotanto-ongelman ilmaantuessa lähtisi ensimmäisenä selvittelemään tilannetta. First responder on kehittäjien kesken vuorotteleva rooli, jolla on tietyt velvollisuudet muun muassa pilvipalveluiden turvallisuuden tarkkailussa. Suurin osa rutiinitehtävistä painottuu alkuviikkoon, mutta tilanteen vaatiessa tehtävät saattavat täyttää suurimman osan työpäivästä ja peräti viikosta.

Tänään pääsin pienellä vaivalla first responder tehtävien kanssa ja pääsin kehittämään lisää tiketin kuusi integraatiotestejä. Uusi rajapinta ja sille tehtävä kysely palauttivat joukon hinnoitteluehtoja, jotka tulostettiin JSON-muodossa. Hinnoitteluehdot palautuivat kahdesta eri rajapinnasta sattumanvaraisessa järjestyksessä. Olisin halunnut, että hintaehdot tulostuisivat aakkosjärjestyksessä ja käytinkin paljon aikaa eri lähestymistavoin ratkaistakseni ongelma. Lopulta ratkaisin ongelman lukemalla hinnoitteluehdot mappeihin ja tulostamalla ne sieltä aakkosten mukaan lajitellussa järjestyksessä irrallaan muusta hinnoitteluvastauksesta.

Tiistai 1.11.2022

Tiketin kaksi uuden maan käännöstiedosto palautui minulle liiketoiminnan edustajilta. Lisäsin käännökset sovelluksen käännöstiedostoihin ja testasin vielä sovellusta, jotta kaikki tarvittava olisi käännetty. Sovellus oli päätetty ottaa uuteen maahan käyttöön vaiheittain ja ihan kaikkia ominaisuuksia ei oltu vielä ajan säästämiseksi ja sekaannusten välttämiseksi käännetty. Kaikki näytti hyvältä, joten siirsin tiketin eteenpäin katselmoitavaksi ja testattavaksi.

Lisäksi pääsin jatkamaan tiketin neljä rajapinnan testausta osittain kirjoittamieni integraatiotestien kautta ja myös palaverin välityksellä SAP-asiantuntijan kanssa. Vertasimme käyttötapauksia SAP:in ja sovelluksemme kautta ja mietimme mitä arvoja uuden rajapinnan tulisi missäkin tilanteessa palauttaa. Rajapinnan kehitys ja sen valmistuminen estivät kehitystiimini muiden töiden etenemistä, joten vaikka sen testaaminen ei ollut varsinaista sovelluskehitystä minun osaltani, työ oli tärkeää saada etenemään.

Keskiviikko 2.11.2022

Tänään jatkoin edelleen tiketin neljä rajapinnan testausta ja tiketin kuusi integraatiotestien kirjoittamista. Kävimme jälleen SAP-asiantuntijan kanssa läpi muutamaa hinnoitteluskenaariota, jotka vaikuttivat rajapinnan antamaan vastaukseen. Huomasin että omasta aiemmasta taustasta kaupan alalla, ja vieläpä rautakauppa-alan myyntitehtävistä oli hinnoittelun ymmärtämisessä todella suuri hyöty. Löysimme yhdessä vielä muutamia kohtia, jotka tuli korjata rajapinnan antamaan vastaukseen SAP:in päässä.

Torstai 3.11.2022

Tänään osa kehitettyjen sovellusten päivittäin ajettavista automaattitesteistä olivat menneet virhetilaan ja käytin osan päivästä niiden selvittelyyn. Tämä työ oli ensisijaisesti first responderin vastuulla, mutta toki kuka tahansa tiimistä sai auttaa, varsinkin jos tiesi heti syyn virhetilalle.

Lisäksi sain tehtäväkseni muutaman nopean feature flag tiketin, jotka liittyivät palvelun uuteen maahan julkaisun lähestymiseen. Feature flag asetetaan joillekin tietyille toiminnallisuuksille tai sovelluksen osioille, joiden ei haluta näkyvän kuin esimerkiksi testiympäristössä, tai tässä tapauksessa vain tietyn maan käyttäjille.

Perjantai 4.11.2022

Tänään sain tiketin kuusi integraatiotestit menemään kaikki läpi ja kehitystyö SAP:in päässä oli saatu päätökseen. Muokkasin integraatiotestejä vielä hieman ja avasin pull requestin. Integraatiotestini toimivat mainiosti ja testasivat oikeita asioita, mutta minulla oli hieman ongelmia koodin selkeyden ja logiikan kanssa. Kävimme kokeneemman kollegan kanssa Javan tietorakenteita kuten mappeja ja settejä ja kuinka niitä voisi hyödyntää testeissä. Sain myös jälleen kerran vinkkejä luettavan koodin kirjoittamiseen ja loinkin useita apumetodeja, joiden taakse piilotin hankalampaa logiikkaa. Olin jälleen todella kiitollinen kollegan avusta, sillä vaikka sain koodin itsenäisesti toimimaan oikein, ei luettavan koodin kirjoitusta voi oikein oppia kuin antamalla muiden lukea koodia ja saamalla siitä palautetta. Päivän lopuksi integraatiotestien pull request hyväksyttiin. Kävin myös läpi SAP-kehittäjän kanssa suunnitelman rajapinnan lopulliseksi testaamiseksi, joka tehtäisiin, kun kaikki rajapintaan liittyvät tiketit oltaisiin mergetty testiympäristöön.

Viikkoanalyysi 8

Tämän viikon tavoitteet täyttyivät. Hoidin suurimman osan vastuullani olleen first responder –roolin puitteissa ilmenneistä työtehtävistä itsenäisesti. Lisäksi autoin kehitystiimin työtä etenemään testaamalla muiden töiden tukkeena toiminutta rajapinnan kehitystä. Testaamisen ohessa kirjoitin rajapinnalle integraatiotestejä, joiden puitteissa opin jälleen lisää luettavan koodin tuottamisesta.

Viikkoanalyysissä 6 sivulla 34 listasin luettavan ja laadukkaan koodin periaatteita. Tällä viikolla opin lisää toisteisuuden poistosta ja funktioiden yhden toiminnan periaatteesta. Tiketillä kuusi kehittämiini integraatiotesteihin oli tullut jonkin verran toisteisuutta ja huonoja datarakenteita. Onneksi kokeneempi kollegani kävi kaikki parannusta vaativat kohdat kanssani läpi, niin että ymmärsin mitä tuli tehdä toisin.

Toisteisuuden havainnointiin ja poistoon voi soveltaa DRY-periaatetta (Don’t Repeat Yourself). Mikäli samaa koodia on kahdessa eri paikassa, tulee päivitykset tehdä myös kahteen paikkaan, jolloin on todennäköistä, että jommankumman päivitys jää jossain vaiheessa tekemättä. Myös koodin luettavuus kärsii, mikäli siinä on paljon toistuvaa koodia. Toisteisuutta saattaa syntyä esimerkiksi tilanteissa, jossa kaksi tiiminjäsentä tekee samoja muutoksia tietämättä toisen tekemisistä. Myös kehittäjän huolimattomuus, osaamattomuus tai suoranainen laiskuus ja koodin kopiointi voivat johtaa toisteisen koodin käyttöön. Koodin kommentit ovat yksi toisteisuuden muoto, josta on helppo

päästä eroon nimeämällä muuttujat ja funktiot kuvaavasti. Koodin toisteisuutta voi poistaa päättämällä datarakenteet ja luokat tarkasti, eristämällä toistuvat koodit funktioihin tai käyttämällä silmukkaa eli toistorakennetta. (Hunt & Thomas 1999, luku 2.)

Kirjoitettaessa funktioita tulee ottaa huomioon yhden toiminnan periaate, joka tarkoittaa, että funktiolla on vain yksi päätoiminnallisuus. Funktion sisällä voi olla muita funktioita, jotka suorittavat omia toiminnallisuuksiaan. Kun kaikki funktiot on nimetty kuvaavasti, on koodia helppo lukea ja seurata. Funktion argumenttien määrään kannattaa kiinnittää huomiota, sillä niiden käyttö vähentää funktion uudelleen käytettävyyttä ja vaikeuttaa testausta. (Martin 2008, luku 3.)

Toisteisuuden poisto tuntuu mielestäni loogiselta ja useimmiten osaan jo välttää sitä itsenäisesti. Funktioiden yhden toiminnan periaate tuntuu taas vaikeammalta. Myös Martin toteaa kirjassaan tämän olevan vaikeaa aloittelijoille, joten hyväksyn että tässäkin asiassa taito karttuu tekemällä ja saamalla palautetta koodista (Martin 2008, luku 3).

4 Pohdinta

Opinnäytetyöprojektin alkaessa olin työskennellyt kehittäjänä noin puolitoista vuotta ja sen päättyessä siihen oli tullut kahdeksan viikkoa lisää. Projektin aikana kehityin kaikilla oppimistavoitteiksi määrittelemilläni osa-alueilla, mutta mitään huimaa hyppäystä kehityksessä ei tapahtunut. Vertaan tätä ajanjaksoon ennen IT-alalla aloittamistani, jolloin suoritin Academic Workin Academy-intensiiviopinnot 12 viikon ajan. Kolmessa kuukaudessa opin lähes nollatilanteesta koodaamaan niin, että opintojen lopussa osasin rakentaa toimivan full stack –sovelluksen. Oppiminen on ollut myös tämän intensiiviperiodin jälkeen nousujohteista, mutta ei kuitenkaan aivan niin jyrkkää.

Opinnäytetyöprojektin alussa asetetut oppimistavoitteet ja niihin liittyvät seurantaviikot on esitetty taulukossa 2. Kuten luvun kolme alussa kerroin, seurantaviikot sisältävät vain oppimistavoitteisiin liittyvät päivätoteumat. Jokaiselle seurantaviikolle valitsin yhden tai useamman viikkotavoitteen oppimistavoitteiden joukosta, joka helpotti päivätoteumien rajaamista.

Taulukosta 2 voidaan havaita, että frontend-kehittäjän taitoja on käsitelty useimmilla viikoilla kuin muita tavoitteita. Täytyy kuitenkin ottaa huomioon, että osa-alueena frontend-kehitys on hyvin laaja kokonaisuus. Opinnäytetyöprojektin aikana kehityin frontend-kehityksessä kohtalaisesti, enimmäkseen TypeScriptin tyypitysten ja tietorakenteiden osalta. Sen sijaan koin kehittyneeni todella paljon työvälineiden käytössä, eli Git-versionhallinnassa ja Jira-tiketinhallintaohjelmistossa. Tämä helpottaa päivittäistä työtäni kehittäjänä. Kehitystä tapahtui myös luettavan ja laadukkaan koodin tuottamisessa. Kävin kattavasti läpi osa-alueen teoriaa ja onnistuin vähentämään koodikatselmoinnissa saamieni muutospyyntöjen määrää. Kehitystiimin jäsenenä toimimisessa kehityin hieman, sillä taitoni eivät valitettavasti usein riitä auttamaan esimerkiksi tiimissä ilmenneissä DevOps-ongelmissa. Pyrin kuitenkin jatkuvasti opiskelemaan lisää ja tekemään pienempiä tiimiä hyödyttäviä tehtäviä. Ketterän kehityksen osalta kehityin jonkin verran, käytyäni läpi Safe:n teoriaa ja toimintamalleja. Tiimissä ei kuitenkaan noudateta Safe:a aivan kirjaimellisesti, joten kehittyminen jäi enemmän teorian tasolle.

Taulukko 2. Peittomatriisi päiväkirjaopinnäytetyön tekstinsisäisistä kytköksistä

Oman ammatillisen kehittymisen tavoitteet	**Seurantaviikko**
1 a. Frontend-kehittäjän taidot	Viikot 1, 2, 5, 6 ja 7
1 b. Luettavan ja laadukkaan koodin tuottaminen	Viikot 3, 6 ja 7
1 c. Kehitystiimin jäsenenä toimiminen	Viikot 3, 6 ja 7
2 a. Ketterä kehitys	Viikot 1 ja 4
2 b. Työvälineet	Viikot 2, 4 ja 5

Opinnäytetyöprojektin aikana tiedonhakutaitoni paranivat ja opin löytämään apua eri ohjelmointikielten dokumentaatiosta. Opin jäsentelemään saamiani työtehtäviä ensin käyden kokeneemman kollegan kanssa läpi tehtävän tarkoituksen, sitten lukemalla koodia ja etsimällä kohdat muutoksille, sitten suorittamaan muutokset järkevissä kokonaisuuksissa Git-versionhallintaa ajatellen. Sisäistin laadukkaan ja luettavan koodin periaatteet. Opin välttämään turhat pr muutospyynnöt, eli nimeämään muuttujat ja metodit, käyttämään apumetodeja ja tekemään testejä uusille tärkeille toiminnoille. Opin myös aikatauluttamaan ja tauottamaan työtäni erilaisten keskittymistä lisäävien tekniikoiden avulla.

Päiväkirjamuotoinen opinnäytetyö ja työtehtävien raportointi toivat konkretiaa ajankäyttöön ja tehtävissä etenemiseen. Opin pyytämään apua nopeammin, mikäli en syystä tai toisesta päässyt työtehtävässä eteenpäin omin avuin. Havaitsin myös, kuinka rikkonaisia työpäivät usein olivat ja kuinka se vaikutti työtehtäviin keskittymiseen. Ryhtyessäni kirjoittamaan päiväkirjamuotoista opinnäytetyötä kuvittelin, että raportoinnit olisi helppo tehdä työn ohessa ja opinnäytetyö valmistuisi kuin itsestään. Viikkoanalyysien tuottaminen teoriaosuuksineen osoittautui kuitenkin yllättävän aikaa vieväksi, joka aiheutti haasteita aikataulussa pysymisen kannalta. Opin jatkossa jättämään aikatauluihin enemmän tilaa.

Opinnäytetyöprojektin aikana tapasin vanhoja opiskelukavereitani, joiden kanssa olin suorittanut kolmen kuukauden koodauksen intensiiviopinnot kesällä 2019. Perhevapaan vuoksi minulle oli kertynyt hieman vähemmän työkokemusta uudelta alalta kuin muille, mutta oivalsin kuitenkin, että nykyinen toimeksiantoni on ollut ammatillisen kehittymisen kannalta todella tehokas. Opiskelukaverini kertoivat omista toimeksiannoistaan ja kokemuksistaan uransa ensimetreillä ja tulimme siihen lopputulokseen, että saadulla palautteella on todella suuri merkitys kehittymisen kannalta. Joissain projekteissa saattoi olla niin kiire, että kukaan ei ehtinyt käydä kunnolla läpi kirjoitettua koodia saatikka antaa siitä rakentavaa palautetta. Koin olevani hyvässä asemassa, kun tiimistäni löytyi useampi kokenut kollega, joilla oli sekä aikaa että taitoa antaa palautetta ja kehitysehdotuksia sekä ennen kaikkea myös mielenkiintoa tehdä sitä. Opin arvostamaan ja hyödyntämään saamaani palautetta ja toisaalta osaan vaatia sitä myös jatkossa.

Junior kehittäjän rooli konsulttina on ollut ajoittain haasteellinen, oletusarvon ollessa että 100 % työstä laskutetaan asiakkaalta. Uran alkuvaiheessa tarvitsisi enemmän aikaa uusiin teknologioihin ja menetelmiin tutustumiseen sekä omaksumiseen. Toki työtehtävissäni ja tikettien työmääräarvioissa on otettu huomioon, että tuottavuuteni ei ole senior kehittäjän tasolla. Siitä huolimatta työmääräarviossa pysyminen on aiheuttanut toisinaan stressiä ja monesti olen opiskellut ja työstänyt tikettejä työajan ulkopuolellakin. Konsultin rooli itsessään sopii minulle hyvin, tulen helposti toimeen

ihmisten kanssa ja sujahdan osaksi työyhteisöä. Mikäli jatkan konsulttina työskentelyä, toivoisin konsulttitalolta mahdollisuutta itsenäiseen opiskeluun ja itsensä kehittämiseen, jotta asiakkaalle tuotettu arvo olisi mahdollisimman suuri. Hyvä tietoperusta tuo myös itsevarmuutta kehittäjän rooliin ja ennen kaikkea enemmän onnistumisen tunteita, jotka ovat miltei parasta koodaamisessa.

Oman työn analysointi on kehittänyt minua oman tason arvioinnissa ja heikkouksien ja vahvuuksien tunnistamisessa. Tästä on varmasti hyötyä seuraavaa projektia etsiessäni ja työhaastatteluihin valmistautuessa. Oikeastaan voitaisiin sanoa, että ammatillinen identiteettini kehittäjänä on selkeytynyt opinnäytetyöprojektin aikana. Analysoidessani työtehtäviäni olen ymmärtänyt mihin suuntaan haluan osaamistani jatkossa kehittää ja missä työtehtävissä voin hyödyntää vahvuuksiani parhaiten. Työn analysoinnista junior kehittäjän näkökulmasta saattaa olla hyötyä myös projektin johdolle, kun he suunnittelevat junioreiden perehdyttämistä jatkossa.

Vaihtaessani uraa IT-alalle tiesin, että alalla saisi jatkuvasti oppia lisää. Mielenkiintoni on tällä hetkellä frontend-kehittäjän työtehtävissä, sekä haluaisin myös oppia lisää palvelumuotoilusta ja designerin tehtävistä. Tavoitteenani on päästä kehittämään sovelluksia, joissa yhdistyy upea käyttäjäkokemus, saavutettavuus ja visuaalisuus.

Miten sitten kehittyä taitavaksi frontend-kehittäjäksi? Frontend-kehittäjän perustaitoja ovat ohjelmointikielet HTML, CSS ja JavaScript. Lisäksi tulisi hallita jokin ohjelmistokehys, kuten React tai Angular (W3Schools s.a.c). CodeBerry-koodauskoulu tutki artikkelissaan 70 frontend-kehittäjän työpaikkailmoitusta vuodelta 2021 ja listasi niissä yleisimmin mainitut taitovaatimukset. Edellä mainittujen lisäksi korkealla listalla olivat TypeScript, versionhallinnan käyttö, REST-rajapintojen tuntemus, testien kirjoittaminen, laadukas ja luettava koodi, debuggaus ja viestintätaidot niin tiimin kanssa kommunikoinnin kuin dokumentoinnin näkökulmista. (CodeBerry s.a.) Opinnäytetyöprojektin aikana opin lisää lähes kaikista edellä mainituista teknologioista ja kehittämisen tukitoiminnoista, mutta kaikesta voi oppia aina lisää. Tulevaisuudessa voin kehittää frontend-kehittäjän taitoja esimerkiksi suorittamalla verkkokursseja ja hakeutumalla frontend-kehittäjän työtehtäviin. Toiveeni olisi päästä kokeneen frontend-kehittäjän työpariksi ja mentoroitavaksi.

Designia ja palvelumuotoilua olen päättänyt opiskella tulevaisuudessa itsenäisesti lisää ja lähitulevaisuuden suunnitelmissani on suorittaa Googlen UX Design -sertifikaatti, joka toivottavasti auttaa tielläni koodaavaksi designeriksi. Parasta IT-alassa on, että koskaan ei ole valmis ja aina voi oppia lisää.

Lähteet

Atlassian s.a. What is version control? Atlassian Bitbucket. Luettavissa: https://www.atlassian.com/git/tutorials/what-is-version-control. Luettu: 3.10.2022.

Atlassian 2022a. What are issue types? Atlassian Support. Luettavissa: https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-types/. Luettu: 28.11.2022.

Atlassian 2022b. What are issue statuses, priorities, and resolutions? Atlassian support. Luettavissa: https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-statuses-priorities-and-resolutions/. Luettu: 28.11.2022.

Babich, N. 26.5.2021. What Is a Hamburger Menu? 5 Website Examples. Xd Ideas. Luettavissa: https://xd.adobe.com/ideas/principles/web-design/what-is-a-hamburger-menu/. Luettu: 7.12.2022.

Beck, K. Beedle, M. Bennekum, A. Cockburn, A. Cunningham, W. Fowler, M. Grenning, J. Highsmith, J. Hunt, A. Jeffries, R. Kern, J. Marick, B. Martin, R. Mellor, S. Schwaber, K. Sutherland, J. Thomas, D. 2001. Julistuksen takana olevat periaatteet. Ketterän ohjelmistokehityksen julistus. Luettavissa: https://agilemanifesto.org/iso/fi/principles.html. Luettu: 8.12.2022.

Briggs, T. 3.6.2022. Persist state with Redux Persist using Redux Toolkit in React. LogRocket. Luettavissa: https://blog.logrocket.com/persist-state-redux-persist-redux-toolkit-react/. Luettu: 30.11.2022.

CodeBerry s.a. Mitä front-end-devaajan tulee tietää vuona 2022? CodeBerry. Luettavissa: https://codeberryschool.com/blog/fi/front-end-devaaja-2022/. Luettu: 30.11.2022.

Codesphere 1.4.2022. The Beginner’s Guide to React Redux. Codesphere. Luettavissa: https://medium.com/codesphere-cloud/the-beginners-guide-to-react-redux-eba864e91bb4. Luettu: 30.11.2022.

Coudouy, A. 4.11.2021. GitHub pull request template. Engineering collaboration and tech news. Luettavissa: https://axolo.co/blog/p/part-3-github-pull-request-template. Luettu: 27.11.2022.

Eddie, J. 27.2.2022. FrontEnd vs. BackEnd vs. Full Stack Development: Choose The Right Stack. Geek Culture. Luettavissa: https://medium.com/geekculture/frontend-vs-backend-vs-full-stack-development-choose-the-right-stack-bb24799721b2. Luettu: 5.12.2022.

GitHub 2022. About pull requests. GitHub Docs. Luettavissa: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests. Luettu: 27.11.2022.

Harle, R. 3.5.2018. SAFe tulee ja pelastaa? CGI Suomi. Luettavissa: https://www.cgi.com/fi/fi/blogi/safe-tulee-ja-pelastaa. Luettu: 25.9.2022.

Heikniemi, J. Syyskuu 2022. Jatkuvien päivitysten aikakausi. Devisioona. Luettavissa: https://www.devisioona.fi/2022/09/jatkuvien-paivitysten-aikakausi/. Luettu: 7.12.2022.

Helsingin Yliopisto s.a. Osa 2 - Versionhallinta: Git ja Github. Tietokone työvälineenä - Lapio. Luettavissa: https://tkt-lapio.github.io/git/. Luettu: 2.10.2022.

Hietaniemi, J. 17.5.2016. SAFe tuo ketteryyttä XL-kokoisille. Gofore. Luettavissa: https://gofore.com/safe-ketteryytta-xl-kokoisille/. Luettu: 4.12.2022.

Hunt, A. & Thomas, D. 1999. The Pragmatic Programmer. E-kirja. Addison-Wesley Professional. Reading. Luettu: 5.1.2023.

Interaction Design Foundation 2022. Responsive Design: Best Practices. Luettavissa: https://www.interaction-design.org/literature/article/responsive-design-let-the-device-do-the-work. Luettu: 7.12.2022.

Jraleman 7.1.2018. Ternary operators vs if-else statements. Jraleman. Luettavissa: https://jraleman.medium.com/ternary-operators-vs-if-else-statements-6c26f7d034f7. Luettu: 5.1.2023.

Korpela, E. 10.10.2022. #MimmitKoodaa webinaari: Tutustu designin maailmaan - Mitä on käyttäjälähtöinen tuotekehitys? Videoitu webinaari. Katsottavissa: #MimmitKoodaa webinaari: Tutustu designin maailmaan - Mitä on käyttäjälähtöinen tuotekehitys? Katsottu: 10.10.2022.

Koskinen, I. 26.3.2021. Mikä on Kanban: Katsaus menetelmään ja sen käyttö ketterässä projektinhallinnassa. Visma. Luettavissa: https://psa.visma.fi/blog/mika-on-kanban-katsaus-menetelmaan-ja-sen-kayttoon-ketterassa-projektinhallinnassa/. Luettu: 28.11.2022.

Kravets, U. 1.8.2018. Solved with CSS! LOgical Styling Based on the Number of Given Elements. CSS-Tricks. Luettavissa: https://css-tricks.com/solved-with-css-logical-styling-based-on-the-number-of-given-elements/. Luettu: 30.11.2022.

Lehojärvi, J. 12.2.2017. Tekninen velka –tunnusta, tunnista ja rajoita. Sytyke. Luettavissa: https://www.sytyke.org/tapetilla/tekninen-velka-tunnusta-tunnista-ja-rajoita/. Luettu: 7.12.2022.

Makarevich, N. 2.8.2021. React re-renders guide: everything, all at once. Developer way. Luettavissa: https://www.developerway.com/posts/react-re-renders-guide. Luettu: 30.11.2022.

Martin, R. 2008. Clean Code: A Handbook of Agile Software Craftsmanship. E-kirja. Pearson. Lontoo. Luettu: 5.1.2023.

Mayer, C. 2022. The Art of Clean Code. No Starch Press. San Francisco. E-kirja. Luettu: 5.1.2023.

Meta Platforms 2022a. Hooks at a Glance. React. Luettavissa: https://reactjs.org/docs/hooks-overview.html. Luettu: 30.11.2022.

Meta Platforms 2022b. Using the Effect Hook. React. Luettavissa: https://reactjs.org/docs/hooks-effect.html#tip-optimizing-performance-by-skipping-effects. Luettu: 30.11.2022.

Microsoft 2022. TypeScript is JavaScript with syntax for types. Luettavissa: https://www.typescriptlang.org/. Luettu: 17.12.2022.

Microsoft 9.12.2022. Utility Types. Luettavissa: https://www.typescriptlang.org/docs/handbook/utility-types.html. Luettu: 17.12.2022.

Mozilla s.a.a. Introduction to the DOM. MDN Web Docs. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Luettu: 28.12.2022.

Mozilla s.a.b. Overflow. MDN Web Docs. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/CSS/overflow. Luettu: 28.12.2022.

Mozilla s.a.c. Scrollbar-width. MDN Web Docs. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/CSS/scrollbar-width. Luettu: 28.12.2022.

Mozilla s.a.d. Handling common HTML and CSS problems. MDN Web Docs. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/HTML_and_CSS. Luettu: 28.12.2022.

Oikarinen, L. 27.5.2020. Kolaa kaikille – miksi sinunkin kannattaa olla kiinnostunut laadukkaasta koodista. Profit Software. Luettavissa: https://profitsoftware.com/kolaa-kaikille-miksi-sinunkin-kannattaa-olla-kiinnostunut-laadukkaasta-koodista/. Luettu: 5.1.2023.

Ramel, D. 28.6.2022. TypeScript Vaults Ahead of Java to Crack Stack Overflow Top 5. Visual Studio Magazine. Luettavissa: https://visualstudiomagazine.com/articles/2022/06/28/typescript-so.aspx. Luettu: 17.12.2022.

Remes, M. 12.10.2021. Miten vakava on it-alan työvoimapula? Taloustaito. Luettavissa: https://www.taloustaito.fi/Rahat/miten-vakava-on-it-alan-tyovoimapula/#0fbddb2b. Luettu: 29.12.2022.

Scaled Agile 10.2.2021a. Business Owners. SAFe. Luettavissa: https://www.scaledagileframework.com/business-owners/. Luettu: 2.12.2022.

Scaled Agile 10.2.2021b. Iteration Execution. SAFe. Luettavissa: https://www.scaledagileframework.com/iteration-execution/. Luettu: 2.12.2022.

Scaled Agile 10.2.2021c. PI Planning. SAFe. Luettavissa: https://www.scaledagileframework.com/pi-planning/. Luettu: 25.9.2022.

Scaled Agile 27.9.2021. Lean-Agile Mindset. SAFe. Luettavissa: https://www.scaledagileframework.com/lean-agile-mindset/. Luettu: 8.12.2022.

Scaled Agile 6.9.2022. Program Increment. SAFe. Luettavissa: https://www.scaledagileframework.com/program-increment/. Luettu: 11.10.2022.

Sovelluskontti Oy s.a. Opas Gitin perusteisiin. Ohjelmistokehityksen menetelmät. Luettavissa: https://book.sovelluskontti.com/versionhallinta/opas-gitin-perusteisiin. Luettu: 2.10.2022.

Tolvanen, P. 30.3.2021. Etätyö sopii kokeneille asiantuntijoille, nuoret voivat kärsiä paljonkin. Intranet-ostajan opas –blogi. Luettavissa: https://intranet-ostajanopas.fi/2021/03/30/etatyo-sopii-kokeneille-asiantuntijoille-nuoret-voivat-karsia-paljonkin/. Luettu: 4.12.2022.

Ussa, E. 22.9.2022. Hei, sinä digiosaaja – Suomi tarvitsee sinua! FiCom. Luettavissa: https://ficom.fi/ajankohtaista/uutiset/hei-sina-digiosaaja-suomi-tarvitsee-sinua/. Luettu: 28.12.2022.

Valkhof, K. 4.1.2021. You want overflow: auto, not overflow: scroll. Kilian Valkhof. Luettavissa: https://kilianvalkhof.com/2021/css-html/you-want-overflow-auto-not-overflow-scroll/. Luettu: 28.12.2022.

Varshney, H. 23.11.2021. Jira Ticketing System Simplified 101. Hevo. Luettavissa: https://hevodata.com/learn/jira-ticket/. Luettu: 28.11.2022.

WS Finance Oy 19.7.2019. Ketkä ovat yrityksen tärkeimmät sidosryhmät? BusinessCredit. Luettavissa: https://www.businesscredit.fi/blog/keita-ovat-yrityksen-tarkeimmat-sidosryhmat. Luettu: 6.12.2022.

W3Schools s.a.a. What is Full Stack? W3 schools. Luettavissa: https://www.w3schools.com/whatis/whatis_fullstack.asp. Luettu: 4.12.2022.

W3Schools s.a.b. CSS top Property. W3 schools. Luettavissa: https://www.w3schools.com/cssref/pr_pos_top.php. Luettu: 30.11.2022.

W3Schools s.a.c. What is a Front-End Developer? Luettavissa: https://www.w3schools.com/whatis/whatis_frontenddev.asp. Luettu: 30.11.2022.