OULUN AMMATTIKORKEAKOULU

Sami Kouva

Kehittyminen ohjelmistokehittäjäksi

Päiväkirjamuotoinen opinnäytetyö

Kehittyminen ohjelmistokehittäjäksi

Päiväkirjamuotoinen opinnäytetyö

Sami Kouva
Opinnäytetyö
Kevät 2022
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Sami Kouva
Opinnäytetyön nimi: Kehittyminen ohjelmistokehittäjäksi
Työn ohjaaja: Tuula Harju
Työn valmistumislukukausi ja -vuosi: Kevät 2022 Sivumäärä:26

Tämän opinnäytetyön tarkoituksena oli auttaa sen tekijää kehittymään ohjelmistokehittäjänä. Opinnäytetyö toimi myös työvälineenä reflektoida työskentelyä erinäisten teemojen ja tietoperustan avulla. Tietoperustassa käytettiin ohjelmistokehittäjien verkkoartikkeleita ja blogeja sekä kirjallisuutta koskien etätyötä ja opiskelua. Työpaikka, jossa olin harjoittelijana opinnäytetyön aikana, on Esko Systems Oy, jonka päätoimipiste on Oulussa. Lähipäivinä työskentelin Oulun toimipisteellä, mutta opinnäytetyön kirjoittamisen aikaan työskentelin pääasiallisesti etänä.

Työnkuvani on täysipäiväistä ohjelmistokehitystä, ja olin ehtinyt työskennellä yrityksessä jo 14 viikkoa, kun aloin pitämään päiväkirjaa. Päiväkirjani on jaettu viiteen osaan, joista käytetään termiä pyrähdys. Pyrähdyksissä käsittelen sen aikana tekemiäni asioita tietopohjaa hyödyntäen ja reflektoin omaa työskentelyäni ja uusia oppimiani asioita. Päiväkirjaa on pidetty kymmenen viikkoa.

Opinnäytetyölle asetetut tavoitteet saavutettiin ja se oli hyvä työkalu omalle oppimiselleni sen pakottaessa pysähtymään tarkistelemaan tehtyä työtä tarkemmin kuin vain raportoida, että työt tuli tehtyä. Opinnäytetyössä havaitsemiani ja oppimiani asioita tulen käyttämään henkilökohtaisesti tulevaisuudessa ja toivon, että siitä on myös hyötyä yritykselle, jossa tein opinnäytetyön.

Avainsanat: ohjelmistokehittäjä, päiväkirja, reflektio, junior-ohjelmistokehittäjä

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Sami Kouva
Title of thesis: Growing to be a software developer
Supervisor: Tuula Harju
Term and year when the thesis was submitted: Spring 2022 Number of pages: 26

Goal of this thesis was to help my growth as software developer. Thesis worked as a tool for me to use to reflect my work with help of themes and theoretical base. For theoretical base thesis uses web articles, blogs from software developers and literature concerning distance work and studying. The company is Esko Systems Oy which has its main office in Oulu. I had office days there but mostly I worked from home when writing the thesis.

My job as an intern is full day software developing and before my journal keeping, I have been working for this company for 14 weeks. My journal has been divided in five parts called sprints. In these parts I go through my activities during the sprint with the help of theoretical base and I reflect my own work and the things I learned during the sprint. I keep my journal for ten weeks.

Goals for the thesis were met and it was a good tool for my learning. It forced me to focus on observing my work more than only reporting the work hours and tasks. I will use the things I noticed and learned during my thesis in the future and hopefully it will help the company as much it helped me.

Keywords: software development, journal, reflection, junior developer

SISÄLLYS

1 JOHDANTO

1.1 Opinnäytetyön tavoitteet ja tarkoitus

Tämän päiväkirjamuotoisen opinnäytetyön tavoitteena on seurata tekijänsä kasvua ohjelmistokehittäjän rooliin Esko Systems Oy yrityksen palveluksessa. Kirjoitan päiväkirjaa kymmenen viikon ajan. Työaikani on 37,5 tuntia viikossa saldoliukumalla. Työtehtäväni päiväkirjan aikana on ohjelmistokehitys Javascriptiä ja Reactia sekä Reactin eri kirjastoja käyttäen.

Merkintöjeni kautta käsiteltäviä teemoja tulevat olemaan juniori-ohjelmistokehittäjän työnkuvan vaatimukset ja niihin vastaaminen sekä työskentely pääasiallisesti toisen juniorkehittäjän kanssa. Tämän jälkeen käsittelen työparini toiseen tiimiin siirtymistä ja sen vaikutuksia työ- ja oppimisprosessilleni. Käsittelen myös Material-UI React -komponenttikirjaston parissa työskentelemisen haasteita.

1.2 Opinnäytetyön keskeiset käsitteet

Pyrähdys – Ketterän menetelmän lyhyehkö ajanjakso, joka rytmittää kehitystyötä.

Scrum – Yksi ketteristä menetelmistä työskennellä ohjelmointialalla määrittää tiimi kokoonpanoa ja seurantakokous määriä ja rakenteita.

JavaScript – ” JavaScript on alun perin Netscapen kehittämä, pääasiassa verkkoympäristössä käytettävä dynaaminen komentokieli.”(Wikipedia 2022.)

Material-UI – React-komponenttikirjasto. Tässä kirjastossa löytyy useita eri valmiskomponentteja: nappeja, tekstinsyöttökenttiä ja paljon muuta. Näihin komponentteihin on rakennettu valmiita toimintoja tietyille komennoille, joka helpottaa koodausta.

Front-end – Selainpuolen toiminnallisuus ohjelmoinnissa.

Back-end – Palvelinpuolen toiminnallisuus ohjelmoinnissa.

React – Avoin lähdekoodi JavaScript-kirjaston front-end toiminnallisuuksille.

Css – Cascading Style Sheets koodaus kieli, jolla määritellään erilaisia graafisia tyylejä koodi komponenteille.

Scss – Esikäännettävä skripti kieli laajentaa ja helpottaa css määrityksiä.

Branch – Ohjelmiston kehityshaara. Yleinen käytäntö ohjelmoinnissa on, että uusia ominaisuuksia tehdään omissa kehityshaaroissa, jotka yhdistetään valmiiksi saamisen jälkeen pääkehityshaaran kanssa.

Merge – Kehityshaarojen yhdistäminen.

2 NYKYTILANTEEN KUVAUS

2.1 Yritys ja työympäristö Esko Systems Oy

Olen harjoittelijana Esko Systems nimisessä yrityksessä. Esko System Oy on julkisomisteinen in-house-yhtiö, joka kehittää potilastietojärjestelmää, joka on käytössä jo yli 10 000 käyttäjällä (Esko Systems 2022 a.). Esko Systemsin omistajia on tällä hetkellä Pohjois-Pohjanmaan sairaanhoitopiiri, Lapin sairaanhoitopiiri, Länsi-Pohjan sairaanhoitopiiri, Oulun kaupunki, 2M-IT Oy, Kainuun sote, Keski-Pohjanmaan sosiaali- ja terveyspalvelukuntayhtymä Soite, LapIT Oy ja Istekki Oy. Esko Systemsillä on toimipiste Oulussa ja Rovaniemellä. (Esko Systems 2022 b.)

2.2 Sidosryhmät

Sidosryhmiä työpaikallani ovat eri Esko-järjestelmän osien kehitystiimit, tietohallinto meille itsellemme, tietohallinto asiakkaille sekä erinäisistä koulutuksista vastaavat yritykset. Kanssakäymiseni muiden tiimien kanssa tapahtuu lähinnä henkilöstöinfoissa ja inkrementtidemoissa, joissa esitellään ja käydään läpi tiimien tekemiä uusia ominaisuuksia tai muutoksia muille yrityksen jäsenille eli pääasiallinen kanssakäyminen tapahtuu oman tiimin kanssa. Oman tiimin välinen kanssakäyminen tapahtuu Microsoft Teams kokouksina, sähköpostitse tai fyysisinä kokouksina työpaikan kokoustiloissa. Koko tiimin välinen keskustelu tapahtuu pyrähdyksen aikana olevissa seurantakokouksissa, joista enemmän pyrähdys ja sen rakenne osiossa.

Ulkopuolisia sidosryhmiä ovat omistaja-asiakkaat ja muiden saman alan palvelutarjoajien henkilöstö. Lisäksi ulkopuolisia sidosryhmiä ovat erilaiset laadunvalvontaelimet. Olen vain välillisesti yhteydessä laadunvalvontaelimiin, sillä en itse palauta eri sertifikaatteja koskevia dokumentteja, mutta osallistun niiden täyttämiseen tiimini jäsenenä. Omistaja-asiakkaisiin oma yhteys tulisi olemaan vain häiriötilanteissa tikettipäivystäjän ominaisuudessa.

2.3 Nykytilanteeni

Tutustuin Esko Systemsiin osana koulumme yritysprojektia, jossa opiskelijoille tarjotaan mahdollisuutta tehdä yhteistyötä erilaisten yritysten kanssa. Työskentelin projektin parissa noin kaksi kuukautta ja yhteistyö oli sen verran onnistunutta, että yritys tarjosi minulle mahdollisuuden harjoitteluun ja polun työelämään ohjelmistokehityksessä. Yksi tärkeimmistä syistä hakeutumiselleni tietojenkäsittelyn tradenomin koulutukseen oli kerryttää taitoja ohjelmistokehityksessä ja sen tulevassa opiskelussa, joten mahdollisuus päästä jatko-oppimaan ja työskentelemään ohjelmistokehityksen parissa opintojen ohella,oli enemmän kuin tervetullutta.

Työtehtäväni ovat päiväkirjan pitämisen aikana ja pari kuukautta ennen sitä ohjelmistokehitystä pääasiallisesti JavaScriptillä, Reactilla ja Material-UI-kirjastolla. Työnimikkeeni on harjoittelija ja kehitän itseäni tekemään junior-ohjelmistokehittäjän töitä yrityksessä. Päiväkirjamerkintöjeni alkaessa olen työskennellyt jo 14 viikkoa tässä yrityksessä, joten teknologiat ja käytänteet ovat tulleet jo tutuiksi ja näin ollen minun on mahdollista työskennellä suurimmaksi osaksi etänä.

3 OPINNÄYTETYÖSSÄ KÄSITELTÄVIÄ TEEMOJA

Työskentely tiimissä

Ensimmäinen pyrähdys tulee käsittelemään tiimissä työskentelyn tärkeyttä, jota reflektoin Senior-tason ohjelmistokehittäjän Obiyen (2020) artikkelin avulla Käsittelen pyrähdyksen ajalta kokemuksia eri taitotason omaavien tiiminjäsenten kanssa ja miten nämä kokemukset ovat vaikuttaneet minuun ohjelmistokehittäjäksi kehittyessä.

Etätyöskentelyn hyödyt ja haasteet

Päivittäisten toimieni ohella tulen tarkastelemaan etätyön mukana tulevia hyötyjä ja haasteita sekä sitä, miten ne näkyvät omassa työskentelyssäni. Pyrin tarkkailemaan omaa käytöstäni etätöissä verrattuna toimistotyöhön. Etätyöhön liittyvissä haasteissa käytän tietoperustana Haapakoski, Niemelä & Yrjölän (2020) kirjaa ”Läsnä etänä seitsemän oppituntia tulevaisuuden työelämästä”.

Kehittyminen junior-tason ohjelmoijana

Tulen seuraamaan omaa kehitystäni ohjelmoijana ja tarkkailemaan osaamisessani tapahtuvia muutoksia. Tietoperustana tässä tulee olemaan Niko Ollikan (2020) arviointi eri taitotasoisten ohjelmoijien kyvyistä. Ollikka (2020) antaa osviittaa mitkä ominaisuudet kuuluvat minkäkin taitotason omaavalle ohjelmoijalle, joten kokeilen löytää omasta työstäni kehittyneitä taitoja ja taitoja, jotka vaativat vielä kehitystä.

React ohjelmakehityksessä

Käyn läpi omia kokemuksiani Reactista ja reflektoin omia kokemuksiani toisen ohjelmakehittäjän kokemuksiin Reactin parissa. Cezary Goralski (2022) on blogissaan analysoinut Reactia ja kokemuksiaan sen kanssa, joten käytän hänen kokemuksiaan ja näkemyksiään tietopohjana.

Dokumentaatio ohjelmistokehityksessä

Tämä teema tulee käsittelemään dokumentaation tärkeyttä ja ongelmia ketterien menetelmien ohjelmistokehityksessä sekä omia kokemuksiani kommentoinnin ja muun dokumentaation parissa. Henri Laine (2021) on käsitellyt dokumentaation tärkeyttä blogi-kirjoituksessaan.

Koulutuksen ja työmaailman tarpeiden kohtaaminen

Yksi teemoista, joita pohdin opinnäytetyön aikana, tulee olemaan oma kokemukseni koulutukseni riittävyydestä työelämän tarpeisiin. Tietopohjana tässä tulen käyttämään Kallosen & Kuhmoisen (2021) teosta ”Jatkuva oppiminen työelämän tärkein taito”.

4 TYÖTEHTÄVIEN KUVAUS JA OPPIMINEN PÄIVÄKIRJAMERKINTÖINÄ

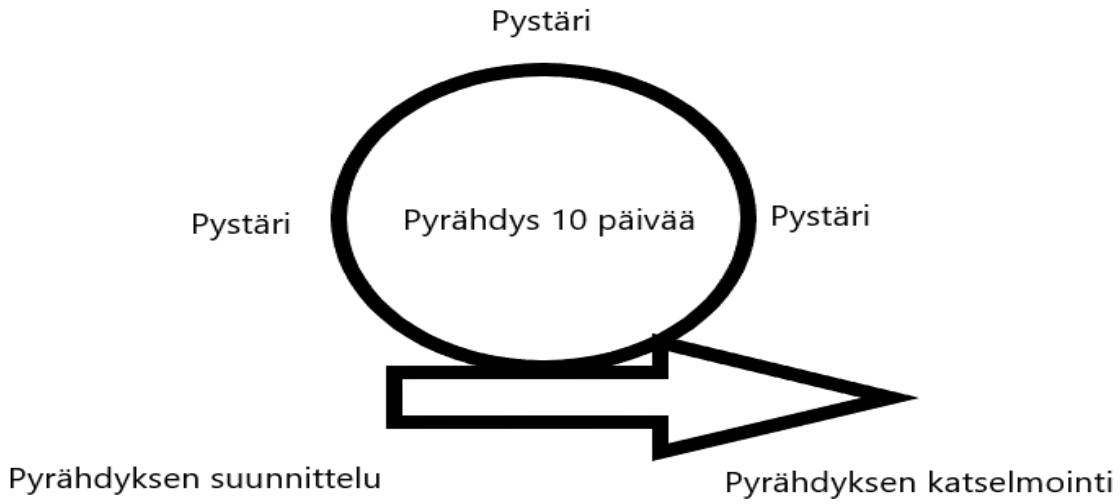
4.1 Päiväkirjan muoto

Päiväkirjani pituus tulee olemaan kymmenen viikkoa, joka on jaoteltu viiteen sprinttiin eli pyrähdykseen. Tulen käsittelemään pyrähdyksessä yhtä tai useampaa teemaa tietoperustaan peilaten. Pyrin pyrähdyksen aikana tarkkailemaan tietyllä teemalla työskentelyäni ja reflektoimaan pyrähdysten aikana kokemiani asioita.

4.2 Pyrähdys ja sen rakenne

Ohjelmistokehityksessä työpaikallani kehitystyömme tapahtuu ketteriä menetelmiä käyttäen scrum-viitekehyksessä: ”Scrum-termi viittaa rugbyn aloitusryhmitykseen. Ketterän kehityksen yhteydessä se tarkoittaa yleensä seisten pidettäviä lyhyitä kokouksia, joissa käydään läpi tulevan päivän tehtävät.”(Itewiki 2022.)

Työpaikallani sprintistä käytämme termiä pyrähdys, jonka rakenne näkyy kuviossa 1.



KUVIO 1 PYRÄHDYKSEN RAKENNE

Pyrähdys suunnitellaan yleensä torstaina suunnittelupalaverissa, jossa jokainen vuorollaan arvioi työmäärän, jonka hän pystyy suorittamaan pyrähdyksen aikana. Pyrähdys kestää 1,5 viikkoa eli noin kymmenen päivää. Pyrähdyksessä on työskentelyn seurantakokouksia eli pystäreitä maanantaina ja torstaina. Pyrähdyksen katselmointikokous on yleensä torstaisin, jossa tarkastellaan kaikki tehtävät, joita tiimi on ehtinyt tehdä. Katselmointikokouksessa harjoitetaan retrospektiiviä: mikä meni hyvin ja missä voisi parantaa. Pyrähdyksiemme seurantakokoukset ovat hieman pitempiä kuin monissa scrum-ympäristöissä pidettävät päivittäiset scrum-kokoukset, joista käytetään daily-nimitystä. Tämä johtuu lähinnä siitä, että järjestämme näitä vain kahdesti viikossa ja pyrähdykselle näitä tulee yleensä kolme.

4.3 Pyrähdys 1 (7.3 - 23.3.2022)

Ensimmäinen pyrähdykseni tässä päiväkirjamuotoisessa opinnäytetyössä on rakenteeltaan hieman erilainen kuin tavalliset pyrähdykset, joista olen maininnut luvussa 4.2. Tämä pyrähdys kesti lomien vuoksi kolme viikkoa. Pyrähdyksen pituuteen oli varauduttu pitemmällä pyrähdyksen suunnittelukokouksella, jossa tehtäviä oli jaoteltu koko pyrähdyksen mitalle ja keskusteltu auki ketkä työntekijät ovat milloinkin töissä, jotta voin junior-kehittäjänä kysyä neuvoa tarpeen vaatiessa. Pyrähdyksen aikana tein valmiiksi Material-UI-pohjaisen kontekstimenukomponentin alusta loppuun: ulkoasun, aktivointimääreet sekä kontekstimenukomponentin kytkemisen olemassa olevaan koodipohjaan ja vanhemman version back-end-kytkentäfunktioiden määrittelyt toimimaan työparini kanssa tehtyyn versioon. Kontekstimenu on siis pop-up-menu, joka ilmestyy jostakin käyttäjän interaktiosta tässä tapauksessa hiiren oikeasta napista klikaten tietyssä osiossa ohjelmistoa.

Material-UI on ollut suhteellisen pitkään jo olemassa, mikä tarkoittaa, että jos sen kanssa tulee ongelmia, yleensä siihen samaan ongelmaan on joku aikaisemminkin törmännyt ja keksinyt ratkaisun. Pitkäaikaisuus on myös ongelmallista spesifeihin ongelmiin törmätessä, sillä osa ongelmien neuvoista koskee Class-mallisia React-komponentteja, joihin tehdyt ohjeet ja ratkaisut eivät ole suoraa sopivia funktionaalisen komponenttiversion vastaaviin. Tästä seuraa, että ratkaisun löytämiseksi saattaa joutua kahlaamaan läpi sekä Material-UI-dokumentaatiota että myös lukuisia verkkolähteitä Youtube-videoista Stack Overflow -neuvoihin tai jonkun yksittäisen kehittäjän blogipostauksiin.

Komponentin rakentaminen ei ollut niin yksinkertaista kuin aluksi kuvittelin, vaikka minulla oli mallina Class-tyylinen React-komponentti, josta lähdin tekemään Funktio-mallista komponenttia. Oman haasteen komponentin rakentamiseen toi se, että rakensin komponenttiani työparini samalla rakentaman komponentin sisään. Jouduin keskustelemaan hänen kanssaan lähes päivittäin siitä, millaisia muutoksia hän tekee alkuperäiseen komponenttimalliin ja toisinpäin.

Työparin kanssa työskentelyn hyviä puolia on ongelmien puiminen yhdessä ja tunne siitä, että on jotenkin helpompaa kysyä niin sanottuja ”typeriä” kysymyksiä, kun kyseessä on saman taitotason omaava henkilö. Senior-kehittäjien kanssa keskustelu antaa nopeammin vastauksia joihinkin tarkempiin ongelmiin, mutta kynnys heiltä neuvon kysymiseen on hieman korkeampi. Toisaalta tämä on omasta mielestäni ihan hyväkin asia ohjelmoijana kehittyessä. Siten oppii itse hakemaan ratkaisuja ongelmiin.

Tiimityöskentelyn hyviä puolia, jotka toimivat tiiminjäsenten taitotasosta riippumatta: Koodin läpikäyminen tiimin sisällä noudattaen tiimissä tai firmassa sovittuja hyviä tapoja auttaa pitämään virheet kurissa sekä on oppimiskokemus jokaiselle tiiminjäsenelle. Yhteiset läpikäynnit pitävät muut paremmin selvillä muiden tiiminjäsenten koodista mikä helpottaa virheiden korjausta, vaikka tiiminjäsen ei olisi paikalla. Eritasoisten tiiminjäsenten vuorovaikutus nopeuttaa prosessia ja ratkaisuja ongelmiin sillä joku muu tiimistä on saattanut ratkaista samankaltaisen, ellei jopa saman ongelman. (Obiye 2020.)

Esimerkkinä mainittakoon tapaus, jossa saadaksemme testejä toimimaan, olimme yrittäneet ratkaista ongelmia, jotka aiheuttivat back-end-liitäntöjen virheilmoituksia. Komponenttitestausten aikana työparini ei ollut löytänyt ratkaisua erään ohjelma-asetuksen kanssa ilmenneeseen ongelmaan muutamaan tuntiin tietämättä, että minä olin törmännyt ratkaisuun netissä päivää aikaisemmin ja laittanut asetuksen kohdilleen omaan testausympäristööni. Vastavuoroisesti olin myös itse kuvitellut erään tekemäni komponentin olevan valmis ja toimiva. Työparini kuitenkin huomasi, että komponentin toiminta on vain näennäistä eli se toimisi vain kehitysympäristössä, mutta ei tallentaisi asioita.

Komponentit, joiden parissa työskentelemme, käsittelevät salassa pidettävää tietoa. Tämän vuoksi moni testausvaihe pakotti meidät olemaan toimistolla sisäverkossa, jotta komponenttien liitännät

toimivat ja voimme päivittää testeihin vaadittavia tietoja. Vaikka työparini kanssa olemme työskennelleet lähinnä front-end-toiminnallisuuksien parissa, pitää meidän kuitenkin itse liittää komponenttimme back-endiin, mikä tarkoittaa joidenkin vanhempien back-end-kutsufunktioiden päivittämistä.

Pyrähdyksen alussa meillä oli mahdollisuus keskustella koko ajan senior-kehittäjän kanssa, mutta lomien vuoksi avun saamisen mahdollisuus pieneni, joten itseohjautuvuus ja suunnitelmallisuus sen suhteen minä päivänä mitäkin komponentin osaa voi tehdä tai minkä osan tulee olla valmiina millekin päivälle testauksen tai neuvojen kysymisen suhteen, kehittyi näinä kolmena viikkona huomattavasti.

Noin 30 % komponentin sisällöstä piti testata toimistolla komponenttiin liittyvien ohjelmien ja salassa pidettävien tietojen vuoksi ja testattavia asioita olivat muun muassa:

- Välittyykö komponentin kautta data oikein sekä komponentille itselleen että välittääkö se ne seuraavaan osioon oikein.
- Tunnistaako komponentti kohteensa kaikki ominaisuudet.

Pyrähdyksen lopussa yhdistin (merge) oman kehityshaarani (branch) työparini kehityshaaran kanssa. Tässä menikin sitten hieman pitempi aika, sillä olimme molemmat omissa töissämme työstäneet samoja tiedostoja. Istuimme Teams-kokouksessa pari tuntia ja kävimme tiedosto tiedostolta läpi jokaisen osion. Ratkaisimme kaikki yhdistämisestä seuraavat konfliktit eli mitä muutoksia pidetään tai muokataan kummankin työstä.

Tälle pyrähdykselle sijoittui työajanseurantaohjelman koulutus ja kokouksissa meni noin seitsemän tuntia, joka on vähän yli keskitason normaaliin määrään verrattuna. Toivottavasti ensi pyrähdyksessä kokouksia on hieman kevyemmin.

4.4 Pyrähdys 2 (24.3 - 6.4.2022)

Pyrähdys alkoi bugin huomaamisella ja korjaamisella. Työparini kanssa erikseen tehdyt komponentit eivät toimineetkaan täysin ristiin, sillä molemmissa oli pop-up-toiminnallisuuksia, jotka eivät saaneet toimia päällekkäin. Ensimmäisenä päivänä tein pari tuntia ylityötä sillä olimme tietoisia, että pyrähdys tulee loppumaan inkrementti-demoon, jossa eri tiimit esittelevät tekemiään ohjelmistomuutoksia, joten meidän piti varmistaa tekemiemme osioiden toimivuus. Inkrementti-demoon olisi

siis vielä puolitoista viikkoa, ja työparini alkoi pikkuhiljaa irrottautua tiimimme työskentelystä, koska hän aloitti siirtymisen toiseen tiimiin.

Pyrähdyksen alkuun myös kuului seurantakokous esimiehen kanssa, jossa sain arvion viimeisen neljän kuukauden aikana tekemästä työstäni. Arvio oli positiivinen sekä lupaili jatkuvuutta tässä työpaikassa.

Tässä pyrähdyksessä aloin tekemään erilaista versiota Material-Ui-tooltip-komponentista. Tämä komponentti toimii osana käyttöliittymää ja on selittävä tekstikenttä, joka ilmestyy näkyviin, kun tiettyä osaa ruutu näkymästä osoitetaan hiirellä. Komponenttia kuvataan myös sanoilla ”työkaluvinkki, vihjelaatikko, vihjelappu (Suomisanakirja 2022). Tulen tässä opinnäytetyössä viittaamaan komponenttiin tooltippina koska se on komponentin nimi ja se termi, jota käytämme työssämme. Olimme aikaisemmin ottaneet tooltipin käyttöön projektisamme, mutta tarvitsimme siitä ulkonäöllisesti ja toiminnallisesti erilaisen version.

Tämän komponentin parissa työskentelyn aikana tuli hyvin selväksi valmiskomponenttien haitat ja hyödyt, sillä lähes koko pyrähdys meni selvitellessä ja toteuttaessa komponentin ulkonäöllisiä määrityksiä. Material-UI-tooltip-komponenttiin on rakennettu perustyylit, jotka menevät prioriteetissa ohitse css-määreistä. Suoraan komponentille annettavat tyylimääreet eivät vaikuttaneet yksittäisen tooltipin ulkomuotoon millään lailla ilman paria erillistä riippuvuutta Material-UI kirjastosta ja tämäkin informaatio oli jokseenkin hankalasti kaivettavissa dokumentaatiosta.

Yksi tämän pyrähdyksen haasteista oli kokousten suuri määrä ja pituus. Kokouksiin meni lähes 10 tuntia aikaa. Kokoukset ovat välttämätön osa työtä ja jokaisella kokouksella oli paikkansa tässä pyrähdyksessä mutta ajallisesti ne veivät lähes 1,5 työpäivää.

Yksi mukavimmista ja mielenkiintoisimmista ei-normaaliin työkuvaan sijoittuvista asioista oli yhden tiimimme ohjelman joukkotestaus ja siitä raportointi. Tämä ei kuulu normaaliin työkuvaani, joten se toi miellyttävää vaihtelua. Jokaiselle tiimin jäsenelle jaettiin käyttäjärooli ohjelman sisällä ja testasimme eri osioiden toimivuutta ja kirjoitimme ylös käyttömukavuuden parannusehdotuksia. Tässä testauksessa testasin hetkellisesti väärää osiota, vaikka minulla oli käytössä laajat ja selkeät ohjeet mutta aloitusnäkymäni poikkesi esimerkistä, mikä sai minut pysymään väärässä osiossa.

Etäyhteyden välityksellä työskentely aiheuttaa väärinkäsityksen vaaran. Tavoitteiden selvyydellä ja vastuunjaolla on suurempi merkitys etätyöskentelyssä kuin, jos oltaisiin fyysisesti samassa tilassa työskentelemässä. Samassa tilassa olevilta on paljon helpompi ilmeiden ja muun käytöksen perusteella päätellä onko annetut ohjeet menneet perille. Myöskin kynnys esittää pikainen kysymys lähellä olevalle henkilölle tuntuu itsestäni pienemmältä kuin pyydellä puheenvuoroa ja jakaa omaa näkymää tai mitä kaikkea askelia pitääkin ottaa asioista varmistuakseen. (Haapakoski ym. 2020, 113.)

Etätyöskentelyssä tulee myös helpommin kuuntelijan rooli päälle: luonnollinen keskustelun vastavuoroisuus on kuitenkin useamman askeleen päässä kuin hakea katsekontakti ja esittää kysymys. Monissa työkaluissa on onneksi huomioitu viittaaminen tai puheenvuoron pyytäminen jollain toiminnolla. Tämä tapaus kuvaa hyvin kokemiani etätyöskentelyn haasteita. Toisaalta se hyöty, mikä saadaan siitä, että etätyönä voidaan testauttaa ohjelmistoa monenlaisen kokemustaustan omaavilla henkilöillä, on paljon suurempi kuin haitat ja ongelmat mitä etätyöstä seuraa.

Testauksen bugeista ja huomioista oli helppo ottaa kuvakaappauksia esimerkiksi Snipping tool-työkalulla. Se on todella hyödyllinen työväline muutenkin etätöissä ja varsinkin UI-puolella. Snipping-toolin tai vastaavan työkalun etuna Teams-puhelun tapaisiin ruudunjakomahdollisuuksin on se, että jopa hieman epästabiilimmalla verkkoyhteydellä onnistuu kuvan ja tekstin lähettäminen, ja videota ja mikrofonia ei tarvita.

Pyrähdys huipentui inkrementti-demoon. Siinä esitellään eri tiimien tuotoksia tietyltä aikaväliltä. Työparini kanssa esittelimme valmiiksi saamamme komponentin ja teemavärimuutokset, joita olimme työstäneet jo marraskuusta 2021 asti.

4.5 Pyrähdys 3 (7.4 - 21.4.2022)

Tässä pyrähdyksessä olen ensimmäistä kertaa ilman saman taitotason omaavaa työparia eli kaikki ongelmatilanteeni pitää ratkaista itsenäisesti tai kysymällä tiimin senior kehittäjiltä. Senior-kehittäjille osoittamani kysymykset työskentelyni aikana ovat lähinnä koskeneet ohjelman backend-kutsuja, ja mitä mikin osio vanhasta ohjelmasta pitää sisällään. Tässä pyrähdyksessä kommunikaatio senior-kehittäjille oli, hyvin vähäistä. Tooltip, jonka parissa työskentelin oli rakennettu Material-UI pohjalta ja tarkemmin sille perustalle, jota olimme työstäneet työparini kanssa, joten ensimmäistä

kertaa työskentelyssäni oli tilanne jossa minä tunsin työstämäni osion koodipohjan parhaiten jäljellä olevasta tiimistämme.

Työskentelyni on ollut tähän mennessä komponentti kerrallaan tapahtuvaa ja isoksi osaksi ”käännöstyötä” tarkoittaen sitä, että meillä oli alkuperäinen komponentti kaikkine toiminnallisuuksineen olemassa ja se piti tehdä uusiksi funktionaaliseksi React-komponentiksi Material-UI toiminnallisuuksia käyttäen. Tässä pyrähdyksessä työskentelystäni alkoi jo löytymään kykyä ennakoida ongelmakohtia ja soveltaa taitojani itsenäisesti, luvan kanssa tietenkin. Ongelmakohta, jonka havaitsin tässä pyrähdyksessä, oli Material-UI tooltip-komponentin versiossa, joka seuraa hiiren osoitinta osoittimen liikkuessa kohteen ylitse, johon tooltip on määritelty. Ongelma syntyi komponentissa, kun asia, johon tooltip on sidottu, tulostui näytölle tietyin edellytyksin, niin tooltip hukkasi välillä koordinaatit, jossa sen tulisi olla. Tästä seurasi, että tooltip hyppi ruudun vasempaan yläkulmaan, vaikka sen olisi pitänyt olla kursorin kohdalla. Tässä pyrähdyksessä lähinnä siirsin ongelmaa havaitsemalla, että ainoastaan yksi osio komponentissamme tarvitsee osoittimen mukana liikkuvan tooltipin. Muut tooltipit pärjäävät perinteisemmällä versiolla, joten tässä pyrähdyksessä teen valmiiksi nappi- tai tekstiobjektiin sidotut tooltipit ja jätän kursoria seuraavan tooltipin ensi pyrähdykseen.

Tämän pyrähdyksen aikana päädyin tarkastelemaan omaa kehitystäni juniori-tason ohjelmistokehittäjänä. Ollikan (2022) mukaan senior-kehittäjä osaa ajatella koodiaan uudelleen käytettävyyden kautta, mutta juniorille toimiva ja hyvä koodi on lähes sama asia. Seniori-tason kehittäjä osaa myös kirjoittaa paremmin uudelleenkäytettävää koodia. Vaikka olen vasta puoli vuotta työskennellyt ohjelmoinnin parissa, olen havainnut tässä taidossa itselläni kehitystä, eli olen oppinut kirjoittamaan osaksi uudelleenkäytettävää koodia.

Tässä pyrähdyksessä tein tooltippien pohjan irralliseksi komponentiksi, joka ei ole sidottu vain tiettyyn komponenttiin, vaan sitä voidaan tarvittaessa kutsua useista eri komponenteista. Pystyin myös tekemään seuraavan pyrähdyksen tooltipin ulkonäöllisesti valmiiksi, mutta sen toiminnallisuudet jäävät seuraavaan pyrähdykseen.

Sain hankittua itselleni sähköpöydän helpottamaan etätyöskentelyäni ja se on helpottanut jumeja ja lihassärkyjä, jota kahdeksan tuntia päivässä tietokoneella työskentely on tuonut mukanaan. Olen ottanut tavakseni vähintään parin tunnein välein työskennellä seisaaltaan ja pyrkinyt vapaa-aikana tietokonepelejä pelaamaan seisaaltaan.

4.6 Pyrähdys 4 (21.4 - 4.5.2022)

Tässä pyrähdyksessä aloin työstämään tooltip-komponenttia, joka ei ollut rakennettu Material-UI tooltip-komponentilla. Käytin tooltipissa Material-UI Grid-komponenttia, joka auttaa minua tekemään komponentin visuaalisesta ilmeestä responsiivisen ja helpottaa eri elementtien paikalleen laittoa. Termiä responsiivinen käytetään ohjelmistokehityksessä viittaamaan siihen, että komponentti tai elementti muuttaa kokoaan käyttäjän laitteen näytön koon mukaan (eLuotsi 2022). Grid-komponentti mahdollistaa sen, että sille annetaan tiedoksi, kuinka paljon mikäkin elementti tulee viemään riviltä, joka on jaettu perukseltaan 12 osaan. Elementtien koko määräytyy prosentuaalisesti vertaamalla annettua lukua lukuun 12. Esimerkiksi, kun elementille annetaan kooksi kuusi, se vie puolet sen rivin tilasta, missä se on ja sille riville jää vielä kuusi tyhjää paikkaa. Luku 12 on näytön leveys, joten edellä mainittu koko kuusi vie puolet näytön tai muun elementtisäiliön leveydestä riippumatta sen koosta. Tämänkaltaista visuaalista ilmettä pystytään tekemään myös normaaleilla css-määreillä, mutta Grid työkalu nopeutti tätä työskentelyä huomattavasti.

Kokouksellisesti pyrähdys oli hyvin tavanomainen ja minun osaltani lähinnä raportoin oman työni etenemisestä. Jouduin ja pääsin työskentelemään todella itseohjautuvasti, sillä komponentti, josta tein uudempaa versiota oli tehty tavoilla, joita en voinut hyödyntää tämän komponentin tekemisessä. Arviolta 40 % pyrähdyksen työajasta meni tutkimustyössä. Tutkin sitä, miten tällainen komponentti tehdään. Loput työajasta meni komponentin tekemiseen.

Olen päässyt pyrähdyksissäni huomaamaan, kuinka paljon apua Reactilla koodaamiseen netistä löytyy, ja kuinka helppoa sen oppiminen on ollut omien kokemuksieni mukaan verrattuna esim Angulaariin. Angular on ohjelmistokehys, joka on monilta osin Reactin kaltainen, mutta nämä kaksi eivät ole laajudessaan suoraan vertailukelpoisia Reactin ollessa kirjasto Javascriptille, joka tarvitsee monia palasia saavuttaakseen samat asiat jota, Angularilla voi tehdä. Nämä kaksi kamppailevat kuitenkin samoissa toiminnallisuuksissa ja siksi näitä monesti verrataan toisiinsa.

Reactin suosiolla on positiivisia vaikutuksia liittyen liitännäiskirjastojen määrään ja ylimääräisen dokumentaation määrään. React ei ole vielä onnistunut tekemään muutoksia, jotka rikkoisivat vanhempia toimintoja välittömästi, vaan jokaiselle vanhemmalle ominaisuudelle annetaan paljon aikaa ottaa pois vanhasta koodista ja korvata se uudemmalla versiolla. (Goralski. 2022.)

Olenkin kehitystyössäni törmännyt vanhempiin tapoihin tehdä asioita Reactilla, ja olen tehnyt ne uusilla suositelluilla tavoilla uudestaan. Angularin kohdalla törmäsin useasti dokumentaatioita etsiessäni siihen, että neuvot olivat vanhentuneet suurien muutoksien vuoksi. Tämä on yksi syy siihen että React on Angularia suositumpi.

Dokumentaation puute ja teknologian kehitys on ollut myös minulla haasteena töissäni.
Kallonen & Kuhmonen (2021,144.) käsittelevät ongelmia, joita Singaporen korkeakoulu- ja aikuiskoulutusjärjestelmissä on havaittu. Siellä akateeminen opetus ei välttämättä pysy työelämän muutoksien perässä. Itsekin olen tehnyt samanlaisia havaintoja esimerkiksi JavaScriptiin, oli ehtinyt tulla suurempi päivitys sinä aikana, kun en ollut sitä ensimmäisen opiskeluvuoteni jälkeen käyttänyt ja mainitsemani Angularin suosion lasku osui juuri opiskelu aikaani. Oma vuosikurssini vielä päätyi opiskelemaan Angularia, mutta seuraavalle vuosikurssille opetetaan Reactia. Työskentelen yrityksessä Reactilla ja JavaScriptilla, joten jos olisin aloittanut vuoden myöhemmin, olisin ollut paremmassa asemassa töiden aloittamisen aikaan.

Reactilla työskentelyssä pidän siitä, kuinka helppoa on useEffect-tekniikalla lisätä ja poistaa tapahtuman kuuntelijoita (event listener) ohjelmaan, mikä itsellä tuntui välillä hieman sekavalta perus JavaScriptin parissa työskennellessä. Tapahtuman kuuntelijan tarkoitus on odottaa käyttäjältä tässä komponentissa sitä, että hiiri menee tiettyyn kohtaan ruutua, jolloin tooltip tulostuu ja sen paikka määrittyy hiiren liikkeen mukaan. UseEffect on yksi uudemmista React ”koukuista” (hook), jolla ohjelmalle annetaan määrityksiä siitä, milloin jokin asia tapahtuu. Tarkemmin selitettynä useEffectillä voidaan määrittää, että asia tapahtuu vain, jos toinen tai usea asia x tapahtuu, ja vain silloin.

Pyrähdyksen loppupuolella sain kuulla, että työpanokselleni on suurempi tarve toisen ohjelmiston parissa. Kokoustamme seuraavassa pyrähdyksessä siitä, mitä kaikkea ehdin vielä viedä loppuun tämän ohjelmiston parissa, ja mitä kaikkea suunnitelmistamme pitää dokumentoida seuraavalle tätä projektia jatkavalle henkilölle.

4.7 Pyrähdys 5 (5.5 - 19.5.2022)

Tässä pyrähdyksessä aloin kasamaan dokumentaatiota asioista, joita olimme suunnitelleet työmme jatkosta ja muuttamaan tekemiemme osioiden ulkonäköä vastaamaan uudempaa yhtenäistä väri- ja ikoni teemaa. Olimme tehneet vanhemman oppaan mukaiseksi käyttöliittymämme

ulkonäköä, joten olimme jo projektin alussa joutuneet miettimään Material-UI:ssa olevan teema-työkalun käyttöä. Teema-työkalu mahdollistaa sen, että eri elementeille, kuten napeille, tekstinsyöttökentille ja muille sellaisille annetaan perusarvot. Tämä helpottaa yhtenäisen ulkonäön pitämistä ja sen kerralla uudistamista.

Tästä teema-työkalusta oli myös hivenen haittaa, kun joissain elementeissä se vei tyylimääreen prioriteetissa ohi myöhemmin annettavista määreistä, mikä hankaloitti yksittäisten elementtien ulkonäön vaihtoa. Tähän vaikeutta toi lisäksi se, että Material-UI komponenteilla on vielä omat tyylisäädöksensä, jotka menevät vielä teema-työkalun tyylien ylitse, ja tämä pakotti meidät tekemään välillä kiertelyitä aikaisemmissa komponenteissamme. Jouduin ne kiertelyt paikallistamaan ja sinne vaihtaman uusia määreitä.

Kirjoitin dokumentaatioon itse tekemieni komponenttien riippuvaisuuksia helpottamaan seuraavan henkilön työskentelyä ja luettelin, mistä tietyt riippuvaisuudet ja tiedostot löytyvät sekä viittasin verkko-opiskelumateriaaleihin teknologioista, joita olimme käyttäneet osioissamme. Dokumentaation tekeminen oli myös oiva oppimiskokemus, sillä sain käytyä jokaisen tekemäni komponentin lävitse vielä kertaalleen. Tähän ei oikein ole ollut aikaa, sillä olen uudessa pyrähdyksessä siirtynyt aina uuteen komponenttiin. Yksi ohjelmistokehittäjän haasteista on se, että komponenttiin yleensä palataan vain virhetilanteissa, ja sitten pitäisikin muistaa mitä oli ajatellut ja tehnyt sen komponentin kohdalla.

Ketterien menetelmien osalta dokumentaatio tuottaa haasteita. Dokumentaatio jää helposti vähemmälle. Dokumentaatiota tehdään eri henkilöille ja näistä ensimmäinen on uusi kehittäjä. Uuden kehittäjän dokumentaation tarve liittyy arkkitehtuurin oppimiseen ja järjestelmään tutustumiseen (Laine 2021).

Olisin omassa oppimisessani arvostanut isompaa määrää dokumentaatiota ohjelmiston parissa työskennellessä, koska se vähentäisi myös tarvetta kysyä ja vaivata senior-kehittäjiä. Olen kuitenkin huomannut, että en itsekään ole täysin vielä oppinut kommentoimaan tai dokumentoimaan koodiani tarpeeksi.

En ehtinyt tässä pyrähdyksessä saamaan vielä täyttä varmuutta uudesta ohjelmistosta, jonka parissa tulen työskentelemään, sillä tiimimme kokee pienen uudelleenjärjestelyn. Tämä ei minua henkilökohtaisesti haittaa, sillä olen jo puolisen vuotta kehittänyt samaa koodipohjaa, joten uusien asioiden pariin siirtyminen tulee varmasti kehittämään minua ohjelmistokehittäjänä.

Pyrähdyksen lopussa on hieman haikeuttakin, sillä ulkonäkömuutokset ohjelmaan saivat sen näyttämään ja tuntumaan hieman vieraalta. Vaikka en itse ole koskaan pitänyt itseäni hirvittävän taiteellisena ihmisenä, kuuluu ohjelmistokehitykseen front-end-puolella komponenttien ulkonäölliset asiat. Kun tein komponentin alusta loppuun, niin loin niihin jonkinasteisen tunnesiteen.

Pyrähdyksen aikana minulle tuli entistä selkeämmäksi, kuinka tärkeää on kirjoittaa koodinsa siihen muotoon, että muutostarpeen sattuessa koodin parissa työskentelevän on helppo löytää se osa koodista, jonne muutokset tulee kohdistaa. Huomasin myös kehitystä muuttujien nimien käytössä omassa koodaamisessani: Aikaisimmissa koodeissani muuttujat olivat yksinkertaisempia, mutta mitä enemmän koodia olin tuottanut, ja mitä monimutkaisempien toiminnallisuuksien parissa jouduin työskentelemään sitä selkeämpiä merkkejä laitoin muuttujiini, jotta voin kohdentaa ongelmien ratkomista koodissani.

5 POHDINTA

Opinnäytetyöni pääasiallinen tarkoitus valmistumisen lisäksi on toimia ylimääräisenä oppimista ohjaavana ja parantavana työkaluna. Se, että jokaisen pyrähdyksen jälkeen saa pidettyä ylimääräisen retrospektiivin omassa rauhassa, auttoi minua sisäistämään tekemällä oppimiani asioita.

Koulutukseni ennen tietojenkäsittelyn tradenomia oli toimisto- ja talouspuolen merkonomi, joten tietokoneella istuminen oli tullut tutuksi. Se oli luonut työskentelytyylini listamaiseksi, selkeästi kohta kohdalta läpikäymiseksi, mikä alkoi muodostumaan joissakin komponenttien rakenteluissa ongelmaksi: luulin, että osio x tulisi olla valmiina ennen osio y:tä, koska ne olivat muissa näkemissäni koodeissa tietyssä järjestyksessä. Se, mikä osio koodista kannattaa tehdä ensiksi, riippuu monesta asiasta ja mitään valmista ohjetta koodin tekemiseen ei ole. Jouduin useasti pysähtymään ja pohtimaan komponenttieni toiminnallisuuksista sitä, että minkä varaan kaikki muut toiminnallisuudet rakentuvat ja aloittamaan siitä. Useasti rakensin myös komponentista kohtaa, jota piti muuttaa, että sain sen toimimaan toivotulla tavalla jälkeenpäin tekemäni osion kanssa.

Ohjelmistokomponenteissa luodaan ohjelmistokokonaisuuden sisällä viittauksia muihin, monesti useampiin tiedostoihin ja niiden kohta kohdalta läpikäyminen ei aina ollut mahdollista tai suotavaa. Se, että saa päässään luotua ohjelmistokokonaisuuden sellaiseksi, että voi sen tehdä kohta kohdalta ilman, että tarkistelee asioita tai joutuu tekemään osioita alusta, kun huomaakin että alkuperäinen suunnitelma ei olekaan paras tai edes toimiva ratkaisu, on enemmänkin senior-ohjelmistokehittäjän taitoa. Tässä vaiheessa en ole itse vielä päässyt tähän pisteeseen.

Muiden tekemästä koodista oppiminen loi omia haasteita pyrähdysten aikana, sillä minun piti myös opiskella niitä tapoja, joilla aikaisempi koodi oli tehty tietämättä sitä, että tulenko työskentelemään tulevaisuudessa samanlaisen koodipohjan parissa. En koe tätä aikaa millään lailla turhaksi, sillä vaikka koodin rakenne ja käytäntö on todella erilaista riippuen siitä, millä se on tehty, antaa se kontekstia jokaisen uuden koodipohjan parissa työskentelylle.

Itseohjautuvuuteni joutui koetukselle, kun työparini siirtyi toiseen tiimiin. Työtäni oli rytmittänyt työparini työstämien osioiden tarpeet ja toisinpäin, sekä kahdestaan käydyt keskustelut ja yhteistyö oppimisprosessista lähtien. Enemmän yksinään työskentely on pakottanut minut opettelemaan työpäivien rytmitystä ja suunnittelemaan työskentelyäni suurempina kokonaisuuksina, kun työparini

työskentelyn vauhti tai kohteen riippuvaisuudet ei ole ollut luomassa minulle raameja työskentelyn aikataulutukselle päivätasolla.

Pääasiallinen motivaationi opiskella tietojenkäsittelyn tradenomin tutkinto oli oppia oppimisprosessia koodaukselle, mutta sanoisin, että vasta työskennellessäni tässä yrityksessä olen onnistunut luomaan tottumuksen asioiden selvittelylle. Tähän kuuluu sekä kehitys- että korjaustyöt. On todella arvokasta päästä työskentelemään ohjelmiston pariin, jota on tehty pitemmän aikaa, koska sieltä näkee muiden kehittäjien tapaa koodata, jota voi soveltaa tai käyttää lisäopiskelun oppaana.

Itselle tärkein asia, jonka huomasin opinnäytetyötä kirjoittaessa ja mikä konkretisoitui viidennessä pyrähdyksessä, jossa kävin työstämääni ohjelmistoa läpi ilman jatkuvaa kiirettä, oli se, että olen todellakin ohjelmistokehittäjä, vaikkakin junior sellainen ja olen kehittänyt ohjelmistoa. Tämän konkretisoitumiseen meni hieman aikaa, sillä jokainen komponentti on vaatinut oppimista ja oppimisen hyödyntämistä. Ennen tuota pyrähdystä en ollut sisäistänyt sitä, että ei se uuden oppiminen ja sen hyödyntäminen lopu, kun valmistuu koulutuksestaan, vaan kun toimin ohjelmistokehittäjänä, en ainoastaan kehitä ohjelmistoa vaan myös itseäni. Kehittämällä omia tietojani ja taitojani opin paremmaksi ohjelmistokehittäjäksi.

LÄHTEET

eLuotsi 2022. Responsiivisuus. Hakupäivä 9.8.2022. https://www.eluotsi.fi/responsiivisuus/.

Esko Systems 2022a. Yhtiö. Hakupäivä 6.8.2022. https://eskosystems.fi/yritys/.

Esko Systems 2022b. Yhtiö/Omistajat ja hallitus. 6.8.2022. https://eskosystems.fi/yritys/omistajat-ja-hallitus-2/.

Goralski, Cezary, Pros and Cons of React. Hakupäivä 7.8.2022. https://thecodest.co/blog/pros-and-cons-of-react.

Haapakoski, Kati, Niemelä, Anna & Yrjölä, Elina 2020. Läsnä etänä seitsemän oppituntia tulevaisuuden työelämästä. Helsinki: Alma Talent.

Itewiki 2022- Ketterät menetelmät, agile, LEAN ja scrum. Hakupäivä 22.5.2022. https://www.itewiki.fi/opas/ketterat-menetelmat-agile-lean-ja-scrum/.

Kallonen, Tarja & Kuhmonen, Annemari 2021. Jatkuva oppiminen työelämän tärkein taito.
1. painos. Turenki: HansaBook.

Laine, Henri 2022. Havaintoja ketteryydestä ja ohjelmistojärjestelmien dokumentaatiosta. Hakupäivä 7.8.2022. https://blog.digia.com/havaintoja-ketteryydesta-ja-ohjelmistojarjestelmien-dokumentaatiosta.

Obiye, Richard 2022. Importance of teamwork in software development. Hakupäivä 22.5.2022. https://www.linkedin.com/pulse/importance-teamwork-software-development-richard-obiye.

Ollikka, Niko 2022. Miten juniori, keskitason ja seniori- ohjelmoijat eroavat? Hakupäivä 7.6.2022. https://zaibatsu.fi/miten-junioritason-keskitason-ja-senioritason-ohjelmoijat-eroavat/.

Suomisanakirja 2022. Tooltip. Hakupäivä 24.5.2022. https://www.suomisanakirja.fi/tooltip .

Wikipedia 2022. Javascript. Hakupäivä 7.8.2022. https://fi.wikipedia.org/wiki/JavaScript.