Opinnäytetyö (AMK)

Tietojenkäsittely

2022

Tommi Mäkelä

ALOITTELEVAN OHJELMISTOKEHITTÄJÄN OPPIMISPÄIVÄKIRJA

– Päiväkirjamuotoinen opinnäytetyö

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tietojenkäsittely

2022 | 58 sivua

Tommi Mäkelä

Aloittelevan ohjelmistokehittäjän oppimispäiväkirja

- Päiväkirjamuotoinen opinnäytetyö

Tämän opinnäytetyön tarkoituksena oli seurata ja tukea aloittelevan ohjelmistokehittäjän oppimista sekä kasvattaa ammatillista osaamista mobiilisovelluksen jatkokehitystyön yhteydessä. Työn tavoitteena oli myös antaa kuva aloittelevan ohjelmistokehittäjän työstä sekä vastata tekijän laatimiin kysymyksiin.

Kaikki tavoitteet pyrittiin saavuttamaan päiväkirjamuotoisen tutkielman muodossa, jota kirjoitettiin seitsemän viikon ajanjaksolta. Tutkielmassa analysoitiin itsereflektoimalla ohjelmistokehittäjän suoriutumista työtehtävissään sekä perehtymällä ketterän oppimisen menetelmien ja ohjelmistotuotannon käytäntöjen teoriaosuuksiin.

Työn kaikki asetetut tavoitteet saavutettiin, ja niiden seurauksena on tapahtunut huomattavaa ammatillista kehitystä. Prosessin aikana on myös havaittu, että päiväkirjamuotoinen opinnäytetyö on toiminut loistavana ketterän oppimisen työkaluna aloittelevalle ohjelmistokehittäjälle.

Asiasanat:

ketterä oppiminen, reflektointi, clean code, koodin katselmointi, vaatimusmäärittely

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Business Information Technology

2022 | 58

Tommi Mäkelä

A junior software developer’s learning diary

- Diary thesis

The purpose of this thesis was to monitor and support the learning of a junior software developer and to increase his professional competence in development of an existing mobile application. The aim of the work was also to give a picture of the work of a junior software developer and to answer the questions prepared by the author.

All goals were achieved in the form of a diary written over seven weeks. The thesis analyzed the self - reflection of a software developer 's performance in his work tasks and studied the theoretical parts of agile learning methods and software production practices.

All the set goals of the work were achieved and as a result there has been significant professional development. During the process, it has also been found that a diary-based thesis has served as a great tool for agile learning for the junior software developer.

Keywords:

agile learning, reflection, clean code, code review, requirements specification

Sisältö

Kuvat

Käytetyt lyhenteet tai sanasto

API	Application Programming Interface. Ohjelmointirajapinta, joka mahdollistaa järjestelmien keskustella keskenään.
Back end	Ohjelmiston taustajärjestelmä.
Front end	Ohjelmiston käyttäjäjärjestelmä.
GIT	Hajautettu versionhallintajärjestelmä/versionhallinnan työkalu.
JavaScript	Dynaamisesti tyypitetty ja tulkattava monipuolinen ohjelmointikieli.
Kehityshaara	Hajautetun versionhallintajärjestelmän yksi kehityslinja.
React	Metan (facebook) luoma ilmainen ja vapaan lähdekoodin käyttöliittymän kehittämiseen tarkoitettu JavaScript kirjasto.
React Native	Metan (facebook) luoma ilmainen ja vapaan lähdekoodin JavaScript-pohjainen mobiilisovelluskehys.
Tiketti	Kirjattu työtehtävä.
UI-suunnittelija	User Interface Design. Käyttöliittymäsuunnittelu.
UX-suunnittelija	User Experience Design. Käyttäjäkokemussuunnittelu.

1 Johdanto

Jatkuva oppiminen ja osaamisen kehittäminen on edellytys nykypäivän työelämässä ja varsinkin nopeasti kehittyvillä aloilla niiden merkitys vain kasvaa. Ohjelmistoalalla oppiminen on välttämätöntä, sillä työkaluja ja teknologiaa kehitetään jatkuvasti nopealla syklillä. Ohjelmistokehittäjän on siis pysyttävä tässä muutoksessa mukana, mikäli haluaa ylläpitää osaamistaan ja pysyä alalla.

> *Eilen vielä käyttökelpoista osaamista ei ehkä huomisen tuotteessa tai palvelussa enää tarvita, tai toiminto, johon osaaja palkattiin, poistuu tai siirtyy robotille ja algoritmille, jolloin osaajan on nopeasti kyettävä siirtymään uuteen tehtävään ja opittava uudet vaatimukset.* (Otala & Meklin 2021.)

Tässä päiväkirjamuotoisessa opinnäytetyössä tarkastellaan aloittelevan ohjelmistokehittäjän oppimista ja taitojen kehittymistä mobiilisovellusprojektin jatkokehitystyön yhteydessä. Päiväkirjaa kirjoitetaan seitsemän viikon ajalta aikavälillä 11.4.2022 – 27.5.2022. Päiväkirjaseurannan aikana perehdytään myös ohjelmistotuotannon elinkaaren eri vaiheisiin ja menetelmiin. Opinnäytetyön tietoperustana hyödynnetään ketterän oppimisen ja ohjelmistotuotannon käytäntöihin liittyvää kirjallisuutta sekä muita aiheeseen sopivia ja vapaasti saatavilla olevia aineistoja.

Työn ensisijaisena tavoitteena on tukea ja kehittää opiskelijan oppimista ja osaamista aloittelevan ohjelmistokehittäjän roolissa. Toisena tavoitteena on perehtyä keskeisiin ohjelmistotuotannon käytäntöihin, joita projektityössä esiintyy. Kolmantena tavoitteena on tuottaa kokemusten perusteella kuva ohjelmistokehittäjän työstä ja työtehtävistä.

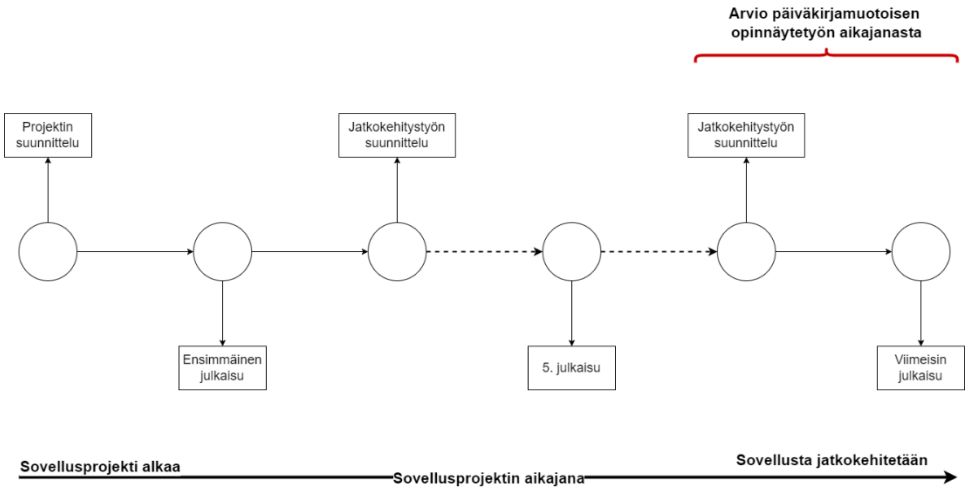
Työn lopuksi pyritään saamaan vastaukset myös seuraaviin kysymyksiin:

- Onko oppimispäiväkirjan kirjoitusten jälkeen tapahtunut ammatillista kehitystä?
- Minkälaisia menetelmiä tai työkaluja hyödynnettiin ammatillisen kehityksen tukena?
- Minkälaisia ominaisuuksia vaaditaan aloittelevalta ohjelmistokehittäjältä?

2 Toimintaympäristön kartoitus

2.1 Nykytilanteen kuvaus ja työtehtävät

Työpaikallani toimin aloittelevan ohjelmistokehittäjän roolissa sekä harjoittelijana. Osallistun jo laajasti käytössä olevan matka- ja kululaskun mobiilisovelluksen jatkokehitysprojektiin kokeneemman ohjelmistokehittäjän sekä projektipäällikön kanssa. Kuvassa 1 on havainnollistettu projektin aikajana.



Kuva 1. Projektin aikajana.

Työtehtäväni tulevat koostumaan seuraavanlaisista tehtävistä:

- mobiilisovelluksen front end -ohjelmistokehitystyöt
- dokumentaation kirjoittaminen
- rajapintojen suunnittelu
- vaatimusten määrittely.

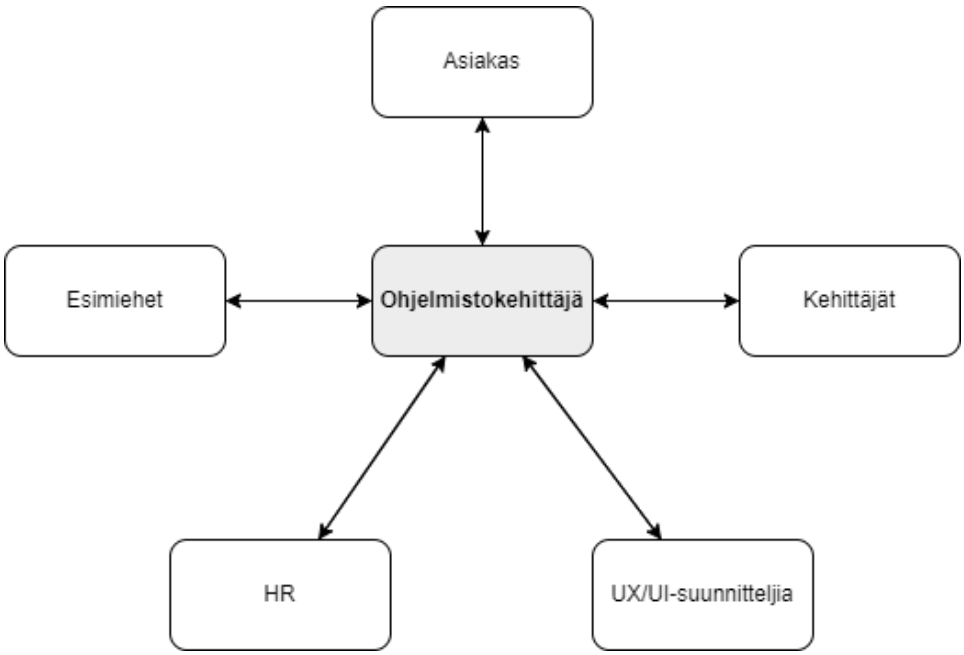
Työtehtävissäni vaaditaan osaamista seuraavanlaisista pääteknologioista ja kirjasto/kehyksistä:

- JavaScript
- React- ja React Native
- GIT

Näiden lisäksi työtehtävien työkalupakkiin kuuluvat erilaiset työkalut, ohjelmistot ja kehitysympäristöt, joiden avulla suunnitellaan, kommunikoidaan tai kehitetään sovellusta.

2.2 Sidosryhmät ja vuorovaikutus

Kuvassa 2 on esitetty omaan työhöni keskeisimmät ja eniten vaikuttavimmat sidosryhmät.



Kuva 2. Työn keskeisimmät sidosryhmät.

Kyseisessä projektissa toimin pääasiallisena kehittäjänä kokeneemman ohjelmistokehittäjän (esimies & kehittäjä) ja projektipäällikön (esimies) kanssa. Näiden henkilöiden kanssa vuorovaikutusta tapahtuu lähes päivittäin vapaan keskustelun tai eri viestintävälineiden ja palaverien muodossa.

Vuorovaikutus asiakkaan kanssa tapahtuu ketterästi eri viestintävälineiden ja palaverien muodossa. Koska kyseisessä projektissa asiakas vastaa taustajärjestelmän ja visuaalisen ilmeen suunnittelusta, vuorovaikutusta tapahtuu myös asiakkaan back end -kehittäjien sekä UX/UI-suunnitteljian kanssa.

Osastomme tiimi järjestää myös säännöllisesti joka toiselle viikolle aktiviteettia, jolloin vuorovaikutusta tapahtuu muiden kehittäjien kanssa vapaan keskustelun muodossa.

Henkilöstöhallinnon kanssa vuorovaikutusta syntyy satunnaisesti keskustelun tai sähköpostin muodossa.

Toimistolla palaverien ulkopuolella osastomme muiden kehittäjien ja oman tiimin kanssa vuorovaikutus näkyy vapaana keskusteluna.

3 Päiväkirjaraportointi

3.1 Viikko 1

3.1.1 Päiväkirjamerkinnät

Maanantai 11.4.2022

Aloitin viikkoni etäpäivänä kiinnostavan työtehtävän parissa, sillä päivän tehtävänä oli jatkaa edellisellä viikolla kesken jäänyttä sovelluksen jatkokehitystyön suunnittelua.

Työtehtävänä oli muodostaa dokumentaatiota asiakkaan vaatimuksista ja kartoittaa iso kuva rakennettavasta uudesta toiminnallisuudesta. Tehtäviin kuului myös uuden työkalun opettelua rajapintojen suunnittelua ja hahmottamista varten.

Työtehtävät olivat selkeitä eikä suurempia ongelmia esiintynyt päivän aikana, joten pystyin työskentelemään rauhassa ja itsenäisesti.

Tiistai 12.4.2022

Tänään palasin työskentelemään toimistolle, ja oli mukava nähdä taas muitakin työkavereita paikan päällä. Työpisteen asettelun ja aamurutiinien jälkeen avustin vielä tiimiläistäni ohjelmistoympäristönsä ongelman ratkaisussa, koska olin itsekin kohdannut samanlaisen ongelman aikaisemmin.

Aamulla pidimme lyhyen palaverin mentorini kanssa, jossa kävimme dokumentaatiotani läpi ja asetimme täksi päiväksi uudet tavoitteet. Tavoitteena oli jatkaa loppupäivän ajan dokumentaation kehitystä ohjeistusten mukaisesti.

Keskiviikko 13.4.2022

Tämän päivän aamurutiinien joukkoon kuului pääsiäismunien etsiminen, sillä meidän toimistollamme oli järjestetty tällainen aktiviteetti. Pääsiäismunajahdin jälkeen kävimme mentorini kanssa läpi vielä edellisen päivän tuloksia, jonka jälkeen asetimme täksi päiväksi uudet tavoitteet.

Tämän päivän tehtävänä oli korjata rajapintasuunnitteluun liittyvät asiat, viimeistellä dokumentaatiota sekä suunnitella alustavasti tikettejä.

Iltapäivällä ennen työpäivän päättymistä meillä oli vielä säännöllinen palaveri mentorini kanssa, jossa keskustelimme pikaisesti kehityksestäni.

Tämän viikon työtehtävien suorituksista sain hyvää palautetta. Dokumentaationi oli laadukasta ja olen pystynyt työskentelemään paljon itsenäisemmin. Rajapintojen suunnittelussa oli kuitenkin huomattavissa kokemuksen puutetta, mutta muuten olen selviytynyt hyvin niiden parissa. Näistä palautteista jäi erittäin positiivinen mieli ja näihin tunnelmiin oli mukava päättää työpäivä.

Torstai 14.4.2022

Ensimmäinen viikkoni päiväkirjassa päättyy etätyön merkeissä jo torstaina, sillä seuraavana päivänä onkin pyhäpäivä tiedossa.

Teimme aamulla mentorini kanssa viimeiset korjaukset dokumentaatioihin, joiden pohjalta lähdimme työstämään työmääräarviota. Työmääräarvion tekeminen oli mielestäni jännittävää, sillä jouduin itse arvioimaan työni laajuuden dokumentaationi pohjalta. Arvostan ja pidän kyllä siitä, että saan paljon vastuuta, mutta olen tehnyt näitä töitä elämäni aikana vasta vähän reilun kuukauden. Tässä ei kuitenkaan ollut vielä kaikki, sillä sain vielä kuulla, että pääsisin aloittamaan toteutusta jo tämän päivän aikana!

Vielä ennen päivän ja viikon päättymistä loin uudet kehityshaarat versionhallintajärjestelmään ja otin uuden tikettini työn alle.

Perjantai 15.4.2022

Pitkäperjantai

3.1.2 Vaatimusten määrittely

Ohjelmistotuotannon (engl. software engineering) yksi keskeisimmistä osa-alueista käsittelee vaatimusten määrittelyä (engl. software requirements). Tutkimusten mukaan jopa 60 % ohjelmistoprojektien epäonnistumisista johtuvat juuri vaatimuksissa olevista virheistä. (Haikala & Mikkonen 2011, 61)

Vaatimusten määrittelyn tarkoituksena on tuottaa hallinnoitua dokumentaatiota siitä, että miten rakennettavan ohjelmiston tulisi toimia asiakkaan mielestä. (Luukkainen 2022a)

Vaatimuksia olisi myös syytä suunnitella lähtökohtaisesti kevyesti, helposti hallinnoitavaksi ja riittävällä tarkkuudella sekä jäljitettävyydellä, sillä vaatimuksiin saattaa tulla yllättäviä muutoksia, lisäyksiä ja tarkennuksia projektin elinkaaren aikana. Turvallisuuskriittisimmissä toteutuksissa suunnittelun tulisi olla tietenkin yksityiskohtaisempi ja tarkempi riskien minimoimiseksi. (Haikala & Mikkonen 2011, 22)

Ohjelmiston vaatimukset erotellaan kahteen luokkaan, toiminnallisiin sekä ei-toiminnallisiin vaatimuksiin. Näistä ensimmäinen luokka käsittelee kaikkia ohjelmiston tarjoamia toimintoja ja toinen luokka taas keskittyy ohjelmiston laatuvaatimuksiin sekä toimintaympäristön rajoitteisiin. (Luukkainen 2022a) Viikon työtehtävät keskittyivät toiminnallisiin vaatimuksiin, joten emme käsittele tässä teoriaosuudessa ei-toiminnallisia vaatimuksia.

Toiminnallisten vaatimusten kuvaamisen työvälineenä ketterissä menetelmissä käytetään käyttäjätarinoita (engl. user story). Kyseessä on tapa kuvailla lyhyesti ja selkeästi asiakkaan kannalta arvoa tuottavia toiminnallisuuksia. (Cohn 2004.)

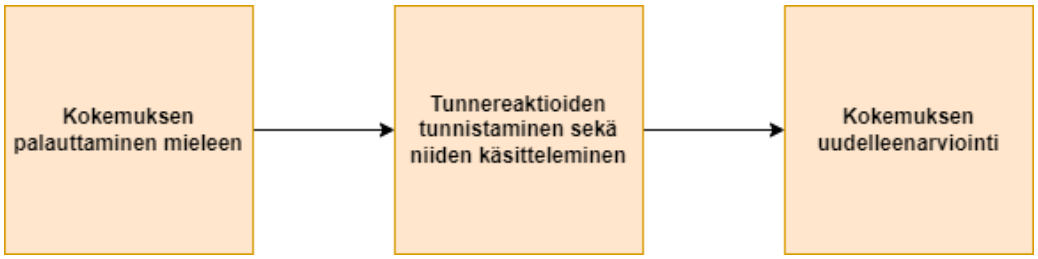
Käyttötapaus (engl. use case) on myös yksi vaatimusten määrittelyyn, tunnistamiseen ja mallintamiseen käytetty työväline. Sen avulla pyritään tunnistamaan skenaarioperusteisesti arvoa tuottavia toiminnallisuuksia käyttäjän kannalta. Tapauksilla pyritään kuvaamaan tekstimuotoisesti toimintojen ja tapahtumien järjestystä. (Haikala & Mikkonen 2011, 79-83)

Käyttäjätarinoiden työmääräarvion suunnittelussa työkaluna voidaan käyttää suhteelliseen kokoon perustuvaa estimointia. Estimoinnin yksikkö story point kuvaa toiminnallisuuden toteuttamisen vaikeutta. Sillä ei siis pyritä arvioimaan tarkasti toiminnallisuuden toteuttamiseen kuluvaa aikaa, vaan antamaan karkea ja suhteellinen arvio, jota voidaan hyödyntää mm. kustannusten arvioinnissa. (Luukkainen 2022a)

3.1.3 Analyysiraportti

Tässä ja tulevissa analyyseissä hyödynnetään itsereflektointia. Itsereflektoinnilla pyritään tunnistamaan ja ymmärtämään sanatonta tietoa uusien kokemuksien syvällisellä pohdinnalla. Tämän menetelmän avulla on mahdollista saada uutta tietoa, joka mahdollistaa uusien näkökulmien löytämiseen sekä itsensä kehittämiseen. (Sironen 2021.)

Kuvassa 3 on esitetty reflektioprosessi, jota pyritään hyödyntämään viikoittaisissa analyyseissä.



Kuva 3. Reflektioprosessi. (Sironen 2021.)

Viikon tavoitteena oli saada jonkinlainen työmääräarvio ja käsitys projektin laajuudesta sekä työtehtävistä. Työtehtävät koostuivat suurimmalta osin mielenkiintoisista suunnittelutyön tehtävistä, joita olivat mm:

- asiakkaan vaatimusten määrittelyä
- rajapintojen suunnittelua
- käyttöliittymäluonnoksien (rautalankamallien) suunnittelua
- työmääräarviointia
- työtehtävien (tikettien) suunnittelua.

Viikon alussa minulla oli vielä hieman sekava kuva projektin laajuudesta ja työtehtävistä, mutta loppuviikosta olinkin hyvin perillä siitä, että mitä tulen tekemään seuraavina viikkoina. Viikko oli siis erittäin opettava ja

merkityksellinen projektin kannalta. Myös kaikki asetetut vaatimukset saavutettiin.

Sain myös itse toteuttaa omalla tyylilläni dokumentaatiota asiakkaan suppeasta toiminnallisista vaatimuksista sekä käyttöliittymäluonnoksien mallinnuksista. Minulla oli tästä työtehtävästä jonkinlainen mielikuva päässä, sillä olimme joskus koulussakin käsitellyt jonkin verran aihetta aikaisemminkin. Tästä sain myös mentoriltani palautetta, että työn jälki oli laadukasta, joka kohotti itseluottamustani suorituksesta.

Haasteita kohtasin rajapintojen suunnittelussa, josta sain myös palautetta. Aihe ja työkalu olivat minulle uusi asia ja niitä piti opetella työn ohella. Tässä tehtävässä minun piti myös pyytää neuvoja mentoriltani päivittäisissä katsauksissa, mutta muuten selviydyin kohtuullisen hyvin tehtävistäni.

Yhteenvetona voisi sanoa, että

- Sain paremman käsityksen projektin laajuudesta ja tulevista työtehtävistä.
- Osasin hyödyntää ja soveltaa aikaisemmin opittua tietoa.
- Pystyin työskentelemään suurimmaksi osaksi itsenäisesti.
- Olin oma-aloitteinen tehtävien suhteen.
- Hallitsin työvälineiden käytön riittävällä tasolla.
- Työtehtävät olivat sopivia eikä kuormituksia esiintynyt.
- Rajapintojen suunnittelussa tarvitaan vielä harjoitusta.

3.2 Viikko 2

3.2.1 Päiväkirjamerkinnät

Maanantai 18.4.2022

Pääsiäispäivä

Tiistai 19.4.2022

Uusi viikko alkaa levättyjen pääsiäispyhien jälkeen.

Ehdin aamurutiinien jälkeen työstämään suunnittelemiani tikettejä noin tunnin verran, minkä jälkeen ryntäsin sovittuun palaveriin. Palaverissa käsittelimme dokumentaatiotani.

Palaverin jälkeen kävimme yhdessä muiden kehittäjien kanssa syömässä talossa sijaitsevassa lounasravintolassa.

Loppupäivä kului koodaamisen merkeissä ja tämä tuntuikin hyvinkin luontevalta minusta, sillä tikettien työtehtävät olivat vielä hyvin yksinkertaisia ja suoraviivaisia. Pärjäsin työtehtävistä myös itsenäisesti ja työnsin päivän päätteeksi lähdekoodini versionhallintaan tarkastettavaksi.

Keskiviikko 20.4.2022

Työpäivä lyhyesti:

- Kävin hakemassa kupin kahvia.
- Tein koodikatselmukset edellisen päivän tuotoksista mentorin kanssa.
- Korjasin katselmuksessa löytyneet epäkohdat.
- Kävin syömässä.
- Otin uusia tikettejä työn alle.
- Osallistuin tiimipalaveriin.

- Työnsin päivän tuotokset tarkastukseen versionhallintajärjestelmään.
- Poistuin työpaikalta.

Asiat, joihin pitäisi perehtyä paremmin tämän päivän perusteella:

- pull/merge requestin työnkulku.

Torstai 21.4.2022

Tänään oli huomattavissa pientä väsymystä johtuen huonosti nukutusta yöstä, joka vaikutti myös keskittymiseen.

Päivän aikana tuli käytettyä 5.30 tuntia koodaamiseen, joista noin puolet ajasta koostui korjaustoimenpiteistä. Loppupäivä koostui kahdesta palavereista.

Parannettavaa:

- Puhtaamman koodin tuottaminen.
- Juurisyiden etsiminen ongelmien esiintyessä.

Perjantai 22.4.2022

Työviikon viimeisenä päivänä tehtävälistalla oli ottaa uusia tikettejä työn alle. Kävimme myös mentorini kanssa lyhyen katsauksen projektin etenemisestä.

Tämän päivän haasteet kohdistuivat lähinnä sovelluksen tilanhallintaan ja ongelmien ratkaisemiseen. Tämän viikon aikaisemmat tiketit ovatkin olleet suhteellisen suoraviivaisia, mutta tästä eteenpäin työtehtävät tulevat olemaan enemmän monimutkaisempia.

Päivän ja viikon päätteeksi lähetin vielä viimeiset muutokset versionhallintaan katselmoitavaksi.

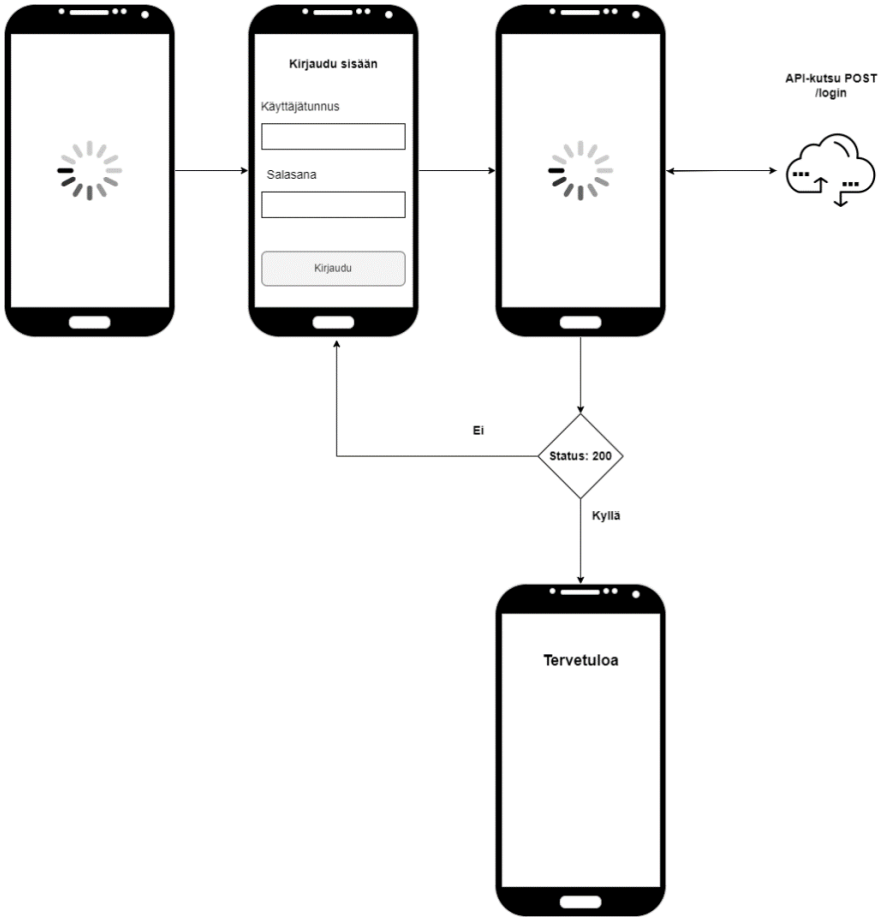
3.2.2 Arkkitehtuurin suunnittelu

Arkkitehtuurisuunnittelu on suunnittelun osa-alue, jonka tarkoituksena on hallita monimutkaista kokonaisuutta hahmottelemalla ohjelman rakenne karkealla tasolla. Kyseessä on hallinnoitu dokumentaatio, joka tuo esille ohjelman keskeisimpien rakennekomponenttien hierarkian sekä niiden väliset kommunikaatio ja rajapintojen määrittelyt. (Luukkainen 2022b)

Suunnittelun toteutukset tulisi olla myös niin selkeitä ja ymmärrettäviä, jotta jokainen projektin kanssa työskentelevä työntekijä ymmärtäisi mistä on kyse. (Haikala & Mikkonen 2011, 180)

> *Kovista yrityksistä huolimatta ohjelmistojen arkkitehtuurien kuvaamiselle ei ole onnistuttu kehittämään mitään yleisesti käytössä olevaa notaatiota.* (Luukkainen 2022b)

Ohjelmiston rakenteen toiminnallisuuden suunnittelun työkaluna voidaan käyttää rautalankamallia (engl. wireframe), joka on esitetty kuvassa 4. Kyseessä on nopeasti ja kustannustehokkaasti toteutettu visuaalinen kuvaus, jonka avulla voidaan hahmottaa sovelluksen toiminnallisuuksia jättämällä ulkoasu yksinkertaiseksi. (Aronen 2017.)



Kuva 4. Rautalankamalli, jossa havainnollistetaan sovellukseen kirjautumista.

3.2.3 Analyysiraportti

Tämän analyysin menetelmänä käytetään henkilökohtaista SWOT-analyysiä, joka on esitetty kuvassa 5.

SWOT-analyysiä käytetään yleensä ongelmanratkaisun työkaluna, mutta sitä voidaan hyödyntää myös oppimisen apuvälineenä. Kyseessä on nelikenttämalli, jossa analysoidaan vahvuuksia ja heikkouksia sekä niihin liittyviä ulkoisia mahdollisuuksia ja uhkia. (Hannonen 2012.)

Kuva 5. Henkilökohtainen SWOT-analyysi.

3.3 Viikko 3

3.3.1 Päiväkirjamerkinnät

Maanantai 25.4.2022

Uusi työviikko alkoi uusien haasteiden kera. Työpäivästä jäi vähän sellainen tunne, että mikään ei kunnolla lähtenyt käyntiin. Syynä tähän saattoi olla se, että päivän tikettien kohdalla en päässyt etenemään ja toiseksi siirryin saman projektin toiselle pitkälle vietyyn kehityshaaraan, josta minulla ei ole kunnolla otetta.

Minulla oli myös palaveri tämän toisen kehityshaaran back end -kehittäjän kanssa, jonka kanssa minun piti testailla sovelluksen toimivuutta. Ikävä kyllä suurin osa ajastamme kului ongelmien selvittelyyn, koska sovellus ei jostain syystä toiminut minun tietokoneellani.

Päivään kuului myös palaveri opinnäytetyön ohjaajani kanssa sekä viimeisimmän merge-requestin epäkohtien korjaustoimenpiteet. Jouduin myös pyytämään apua mentoriltani päivän aikana.

Tiistai 26.4.2022

Tämä päivä lähti käyntiin huomattavasti paremmin. Edellisen päivän kompuroinnit työn alla olevien tikettien suhteen johtuivat siitä, ettei asiakas ollut antanut vielä tarkempia kuvauksia käyttöliittymän lopullisesta rakenteesta. Tämän takia en voinut myöskään lähteä kehittämään niissä olevia toiminnallisuuksia sen pidemmälle. Minun piti siis vetäytyä tästä kehityshaarasta toistaiseksi pois ja keskittyä muihin toiminnallisuuksien kehittämiseen. Tämän oivallettuani lähdin suunnittelemaan päivän uusia tavoitteita ja työtehtäviä.

Loppupäivä koostuikin mobiilisovelluksen muistinhallinnan ja api-kutsujen kehitystöistä. Vaikka ongelmia esiintyi päivän aikana, niin selvisin niistä hyvin ja itsenäisesti.

Keskiviikko 27.4.2022

Tämän päivän tavoitteena oli saada testattua sovelluksen toimivuutta taustapalvelimen kanssa sekä jatkaa muistinhallinnan kehitystyötä.

Aamupäivä meni ongelmienselvittelyssä, sillä minun piti saada sovellus toimimaan Android- sekä iOS-laitteissa.

Iltapäivällä jatkoin sovelluksen muistinhallinnan jatkokehitystyötä ja loppupäivästä pidimme palaverin back end -kehittäjien kanssa, jossa testailimme sovelluksen toimivuutta fyysisillä laitteilla. Päivä sujui muuten hyvin ja selvisin itsenäisesti.

Torstai 28.4.2022

Aamupäivä alkoi edellisen päivän koodin korjaustoimenpiteillä, joiden jälkeen siirryin jatkamaan muistinhallinnan kehitystöitä.

Ennen lounasta oli järjestetty vielä mielenkiintoinen tietoiskutyyppinen palaveri, jonka aiheena oli vaatimusmäärittely. Olin tätä varten valmistellut pienen esityksen, sillä palaverin tarkoituksena oli herättää keskustelua tekemästäni dokumentaatiostani, jota sain esitellä muille.

Iltapäivä koostui muistinhallinnan jatkokehitystyöstä, jossa pärjäsin hyvin ja itsenäisesti.

Perjantai 29.4.2022

Tämän päivän tavoitteena oli vain selviytyä työpäivästä, sillä meillä oli edellisenä iltana järjestetty ohjelmaa osastomme puolesta ja iltakin venyi mukavasti.

3.3.2 Clean code

Puhtaalla koodilla (engl. clean code) tarkoitetaan lukijakeskeistä kehitystyyliä, jolla pyritään tuottamaan ylläpidettäviä ja laajennettavia ohjelmistoja, joita ihmisten on helppo ymmärtää ja muuttaa.

> *Any damn fool can write code that a computer can understand, the trick is to write code that humans can understand.* (Fowler 1998.)

Robert C. Martin mukaan jopa 90 % ohjelmointiin kuluvasta ajasta kuluu olemassa olevan koodin lukemiseen, joten koodin luettavuudella, selkeydellä ja tehokkuudella on merkittävä vaikutus ohjelmistokehityksen kustannuksiin pitkällä aikavälillä. (Martin 2008.)

Puhtaan koodin kirjoittaminenkin vaatii myös paljon enemmän keskittymistä ja aikaa, sillä kirjoittajan on huomioitava koodin luettavuutta, yksinkertaisuutta ja tehokkuutta. (Clean, high quality code: a guide on how to become a better programmer s.a.)

Syitä puhtaan koodin kirjoittamatta jättämiseen voivat olla osaamattomuus ja kiire. (Campbell 2019.) Kokeneemman ohjelmistokehittäjän haastattelun perusteella kävi myös ilmi, että puhdasta koodia voidaan pitää hyvänä mittarina työntekijän kehityksen seuraamiseksi.

Koska puhtaan ja laadukkaan koodin tuottaminen vie enemmän aikaa, niin joskus voi olla jopa asiakkaan kannalta parempi tehdä vähemmän laadukasta koodia, jolloin saadaan paljon aikaiseksi lyhyessä ajassa. Kyseessä on tällöin tarkoituksella tehty päätös lyhytaikaisen teknisen velan ottamisesta. (Luukkainen 2022b)

Puhtaalla koodilla pyritään siis hallitsemaan monimutkaisuutta tuottamalla laadukasta ja helposti omaksuttavaa koodia, joka pitkällä aikavälillä minimoisi teknistä velkaa myös tilanteissa, joissa ohjelmiston kehittäjätiimin kokoonpano muuttuisi projektin elinkaaren aikana. (Luukkainen 2022b)

Lääkkeitä puhtaan koodin tuottamiseen on lukuisia ja niiden käsitteet ovat hyvin laajoja, joten tässä teoriaosuudessa pyritään tuomaan muutama tärkeä käsite lyhyesti esille.

Yleiset periaatteet

Ohjelmakoodin toteuttamisessa voidaan hyödyntää erilaisia alan yleisiä ohjelmoinnin periaatteita, joita ovat mm.

SRP (engl. single responsibility principle), yhden vastuun periaatteella pyritään luomaan luokkia, metodeja tai funktioita joilla saa olla vain yksi vastuu eli syy muuttua.

Low Coupling, riippuvuuksien vähäisyyden periaatteella pyritään minimoimaan tarpeettomia riippuvuuksia luokkien ja olioiden välillä.

DRY (engl. don’t repeat yourself), toisteisuuden periaatteella pyritään minimoimaan saman koodin uudelleen kirjoittamisella.

Refaktorointi

Puhtaan koodin toteuttamisen yksi tärkeimmistä menetelmistä on refaktorointi. Kyseessä on siis prosessi, jossa tehdään muutoksia lähdekoodin rakenteeseen siten, että sen toiminnallisuus kuitenkin säilyy. (Fowler s.a.)

Refaktoroinnin tarkoituksena on tehdä koodista ymmärrettävämpää eli kehittää sen rakennetta luettavammaksi ja yksinkertaisemmaksi sekä tunnistaa ja eliminoida koodissa esiintyviä virheitä. (Vihlman 2022.)

Jotta koodi pysyisi hallitusti puhtaana ja laadukkaana, tulisi refaktorointia toteuttaa jatkuvasti ja tekemällä muutoksia aina pieni pala kerrallaan. (Luukkainen 2022b)

Koodihaju

Koodihajut (engl. code smell) ovat tunnistettavia merkkejä lähdekoodissa, jotka heikentävät koodin sisäistä laatua. Näitä voivat olla mm.

- turhat kommentit (voisiko koodia muuttaa niin, ettei kommentteja tarvita?)
- epäselvät nimeämiset muuttujissa, metodeissa ja luokissa
- pitkät parametrilistat metodeissa tai funktioissa
- toistuva koodi
- liian pitkät metodit tai funktiot.

Koodin katselmointi

Kokeneempien kehittäjien haastattelujen perusteella koodikatselmukset ovat tehokkaimpia lääkkeitä puhtaan koodin ylläpitämiseksi. Tällöin myös vastuu puhtaasta koodista siirtyy ja jakautuu katselmointia tekevälle toiselle kehittäjälle. Tulemme avaamaan koodin katselmointia seuraavassa teoriaosuudessa.

3.3.3 Analyysiraportti

Tässä analyysissä pohditaan tämän viikon työtehtäviä sekä selviytymistä niistä.

Viikon mieleenpainuvimmat aiheet olivat:

Sovelluksen toimivuuden testaus Android ja iOS-laitteilla yhdessä taustapalvelimen kanssa

Näiden tehtävien parissa koin jonkun verran epävarmuutta, sillä näiden toteutusten lähdekoodit olivat muiden tekemiä, enkä ollut ihan varma omista työtehtävistäni. Lisäksi koin ensimmäisessä palaverissa epäonnistumisia demoamisen suhteen.

Lähdekoodin ymmärtämisen tueksi käytin joskus aikaisemmin tekemääni rautalankamallia, joka visualisoi koodin suorituksen flowta. Tämä auttoi yllättävän paljon saamaan kuvan siitä, mitä koodissa tapahtui isossa kuvassa.

Sovelluksen uusimman kehityshaaran muistinhallinnan ja API-kutsujen kehitystyöt

Kehitystyötehtävät ja ongelmanratkaisut olivat näiden tehtävien parissa mielenkiintoisia ja mieluisia. Joihinkin ongelmiin sain vastaukset tutkimalla vanhempaa koodia, sillä ratkaisut olivat samantyyppisiä. Pyrin myös hyödyntämään refaktorointia uuden koodin luonnissa, jotta sain koodin rakenteesta selkeämpää.

Tietoisku ja esitys vaatimusmäärittelystä

Tietoiskussa oli mielenkiintoista kuulla eri asiantuntijoiden näkökulmia vaatimusten määrittelyn dokumentaation toteutuksesta sekä sain myös palautetta omasta dokumentaatiosta.

Tärkeitä pointteja tietoiskusta olivat seuraavat:

- Dokumentaation toteuttamiseen ei ole olemassa mitään jotain tiettyä tyyliä.
- Vaatimusmäärittely käsittelee vain vaatimuksia.
- Käyttäjäpolkujen suunnittelu ja mallinnus kuuluvat yleensä palvelumuotoiluun.
- Omassa dokumentaatiossa oli hyvin tuotu esille käyttäjätarinat ja käyttötapaukset.
- Aihe ja keskustelut olivat mielenkiintoisia osallistujien mielestä.

3.4 Viikko 4

3.4.1 Päiväkirjamerkinnät

Maanantai 2.5.2022

Uusi viikko ja uudet haasteet.

Tämän päivän tehtävälistalla oli selvittää sovelluksen testivaiheessa olevan kehityshaaran integraatio ongelmat.

Työtehtäviin kuului viestintää eri sidosryhmien kanssa ja ongelmien selvittelyä.

Ratkaisua ei syntynyt päivän aikana ja jouduin pyytämään apua myös mentoriltani.

Tiistai 3.5.2022

Päivän tavoitteena oli saada selville edellisen päivän integraatio ongelmien juurisyyt, sillä asiakas toivoi saavansa vastauksia ripeästi. Lopulta saimmekin selville mistä oli kyse ja loppupäivä koostuikin koodin korjaustoimenpiteistä.

Keskiviikko 4.5.2022

Kahden etätyöpäivän jälkeen oli mukava palata toimistolle ja nähdä tuttuja kasvoja paikan päällä. Myös alkuviikon ongelmienratkaisut saatiin vietyä sen verran pitkälle, että sain keskittyä tänään uusimman kehityshaaran työtehtäviin.

Tämän päivän tavoitteeksi asetettiin projektin uusimman toiminnallisuuden käyttöliittymän sekä käyttöliittymälogiikan kehitystyöt.

Käyttöliittymän kehitystöitä oli mukava tehdä, sillä saimme viime viikon perjantaina käyttöliittymän suunnittelijalta ensimmäisiä luonnoksia, jotka helpottivat työn tekoa.

Tavoitteet saavutettiin ja pärjäsin työn parissa myös itsenäisesti.

Torstai 5.5.2022

Tämän päivän työlistalla oli sovelluksen käyttöliittymän ja logiikan kehitystyöt. Pärjäsin työtehtävistä itsenäisesti eikä suurempia ongelmia esiintynyt.

Perjantai 6.5.2022

Aamu alkoi herkullisella aamupalalla, sillä toimistollamme oli järjestetty kattava aamupalatarjonta, jossa pääsi itsekin paistelemaan vohveleita.

Aamupalan jälkeen meillä oli heti palaveri back end -kehittäjien kanssa, jonka aiheena oli sovelluksen uusien toiminnallisuuksien rajapinnat.

Loppupäivä koostui sovelluksen integraatiohaaran viimeistelyistä, josta piti saada testiversio asiakkaalle ennen päivän ja viikon päättymistä. Tämä tavoite saavutettiin ja viikko saatiin pakettiin.

3.4.2 Koodin katselmointi

Koodin katselmoinnin tarkoituksena on parantaa koodin laatua havaitsemalla koodissa esiintyvät mahdolliset ongelmakohdat sekä virheet.

Menetelmä, jossa joku muu kuin ohjelmoija lukee koodin, on pidetty erittäin tehokkaana virheiden etsintäkeinona. Yleisesti on arveltu, että tarkastuksilla voidaan löytää jopa 50–80 % kaikista virheistä. (Haikala & Mikkonen 2011, 198)

Menetelmää suositaan hyödyntämään kehitystyön varhaisissa vaiheissa, jolloin testausta ei ole vielä mahdollista toteuttaa. Tämä ei kuitenkaan korvaa testausta. (Haikala & Mikkonen 2011, 198)

Mitä pidemmälle katselmoinnin toteuttamista siirretään, niin sitä suuremmiksi virheiden korjauskustannukset eli tekninen velka kasvavat. (Haikala & Mikkonen 2011, 198)

Pull request (PR) & Merge request (MR)

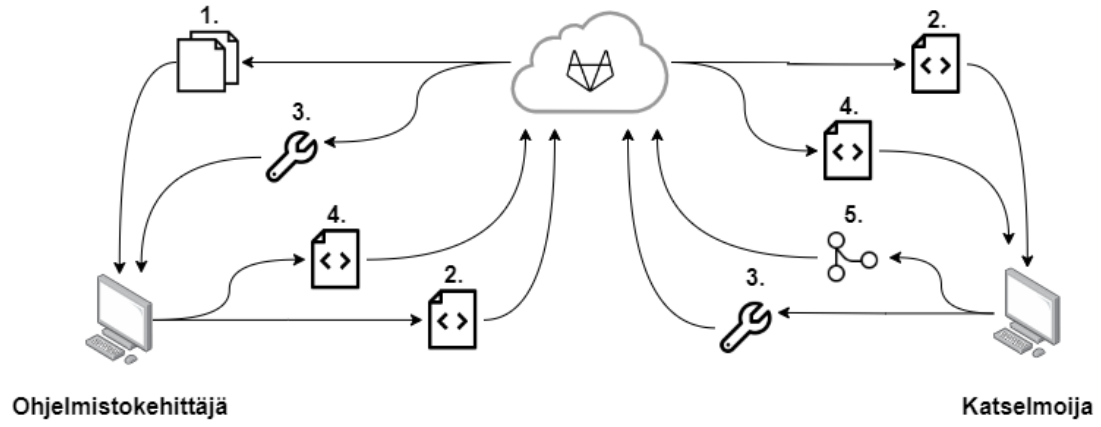
Yksi yleisimmistä työkaluista koodikatselmointien tekoon on hajautettujen versionhallintatyökalujen pull/merge requestit. (Luukkainen 2022c)

Kyseessä on toiminto, jossa pakotetaan toisen tai vanhemman ohjelmistokehittäjän hyväksymään kehityshaarassa tehdyt muutoksen, ennen kuin kehityshaara yhdistetään johonkin toiseen kehitettävään haaraan.

Termit pull ja merge tarkoittavat käytännössä samaa toimintoa, mutta nimet ovat tulleet eri pilvipalvelutarjoajan versionhallintatyökalujen palveluista. mm. GitHub käyttää termiä pull request ja GitLab taas merge request. (Danjou 2021.)

Kuvassa 6 on havainnollistettu GitLabin merge requestin työnkulku ja se etenee seuraavanlaisesti:

1. Ohjelmistokehittäjä luo uuden kopion kehitettävästä kehityshaarasta, eli forkkaa repositorion itselleen.
2. Ohjelmistokehittäjä tekee muutokset omaan kehityshaaraan ja tekee merge requestin hallinnoivalle kehittäjälle, joka suorittaa katselmoinnin.
3. Jos katselmoija havaitsee koodissa puutteita, katselmoija kirjoittaa ohjelmistokehittäjälle joukon korjausehdotuksia ja edetään vaiheeseen 4. Muussa tapauksessa edetään vaiheeseen 5.
4. Ohjelmistokehittäjä tekee muutokset omaan kehityshaaraan ja työntää kehityshaaran muutokset katselmoijalle tarkastettavaksi.
5. Jos katselmoija toteaa koodin olevan kaikin puolin kunnossa, yhdistetään eli mergetään kehityshaara johonkin tiettyyn kehityshaaraan. Tarvittaessa vaiheen 1 kehityshaara poistetaan tarpeettomana versionhallinnasta. Muussa tapauksessa palataan vaiheeseen 3.



Kuva 6. Gitlabin merge requestin työnkulku.

Pariohjelmointi

Yksi ketterän menetelmän eXtreme Programming (XP) parhaista käytänteistä tukee koodin katselmointia, jota kutsutaan pariohjelmoinniksi (engl. pair programming).

Menetelmässä laitetaan kaksi ohjelmistokehittäjää työskentelemään keskenään yhdellä koneella, joista toinen kirjoittaa koodia ja toinen tekee jatkuvaa katselmointia. Roolia vaihdetaan sopivin väliajoin, jolloin pari oppii toisiltaan. (Böckeler & Siessegger 2020.)

Tutkimuksien perusteella menetelmää on todettu parantavan koodin laatua sekä mm. työtyytyväisyyttä. (Luukkainen 2022c)

3.4.3 Analyysiraportti

Tässä analyysissä pohditaan viimeisen kolmen viikon ajanjaksolta koodikatselmuksien merkitystä suhteessa omaan työskentelyyn.

Ensimmäisellä viikolla pääsin aloittamaan ensimmäiset kehitystyöt eikä minulla ollut juurikaan kokemusta tai tietoa tulevasta työn kulusta. Pull/merge requestit ja koodikatselmoinnit olivat uusia asioita, joita piti siis opetella työn ja kiireen ohella.

Toisella viikolla kehitystyöt jatkuivat ja sain viikon aikana useamman merge requestin tehtyä. Katselmoinneista sain nopeasti korjauskehotteet, johon oli selkeästi kuvattu ongelmakohdat.

Kolmannella viikolla työtehtävät keskittyivät suurimmalta osin integraatiohaaran ongelmienratkaisuun. Näistäkin jokaisesta muutoksesta vietiin lähdekoodi katselmoitavaksi, joista sain pieniä parannusehdotuksia.

Kolmen viikon tarkastelujakson aikana koodini katselmoitiin siis useasti ja työnkulkukin alkoi rytmittymään hyvin, jolloin pääsin jatkuvasti tekemään jotain.

Katselmointien korjauskehotteet olivat erittäin opettavaisia, sillä mentorini osasi tuoda esiin myös omia ja tehokkaampia ratkaisuja esille.

Lyhyen katselmointijakson ja pienten lähdekoodin muutosten ansiosta ongelmakohdat löydettiin nopeasti ja lähdekoodi pysyi myös selkeänä.

Johtopäätöksenä voidaan todeta, että koodikatselmuksilla voidaan minimoida virheet sekä samalla parantaa koodin laatua ja selkeyttä.

3.5 Viikko 5

3.5.1 Päiväkirjamerkinnät

Maanantai 9.5.2022

Viikko alkoi etätyön merkeissä ja tämä päivä keskittyi vain API-rajapintojen dokumentaation viimeistelyihin. Päivän aikana piti tutkia asiakkaan käytössä olevien tietorakenteiden nimeämistyylejä ja korjata uusien rajapintojen tietorakenteiden nimet vastaaviksi.

Tiistai 10.5.2022

Palasin toimistolle ja asetin päivän tavoitteeksi saada korjattua kaikki katselmoinnissa ilmestyneet korjaustoimenpiteet.

Suurimpia ja aikaa vieviä toimenpiteitä oli jäsentää lähdekoodin rakenne selkeämmäksi. Tarkoitus oli pilkkoa yhdessä tiedostossa olevia toiminnallisuuksia omiksi moduuleiksi eli jakaa lähdekoodi useampaan tiedostoon.

Jouduin päivän aikana myös pyytämään apua mentoriltani, sillä jäin jumiin yhden mitättömän ongelman takia. Ongelman syynä oli lopulta se, että yritin vahingossa ladata komponenttia väärästä hakemistosta. Yleensä ongelmia pystyy helposti paikantamaan virheviestien tai editorin huomautusten avulla, mutta tällä kertaa nekään eivät auttaneet asiassa.

Päivän päätteeksi sain kaikki korjaukset tehtyä, joista sain tehtyä uuden merge requestin versionhallintaan.

Keskiviikko 11.5.2022

Tämän päivän työtehtäviä:

- työtehtävien suunnittelua
- tikettien suunnittelua
- lähdekoodin korjaustoimenpiteitä
- käyttöliittymän kehitystöitä
- palaveri

Päivän haasteita:

- Kehitystöiden aikana jäin hetkeksi jumiin validointiongelman takia. Tästä jouduin lopulta pyytämään apua mentoriltani ja ratkaisu ongelmaan olikin jälleen erittäin yksinkertainen.

Palaverissa tarkastelimme myös kehitystäni ja keskustelimme puhtaan koodin merkityksestä. Palautteessa kävi ilmi, että tuottamani koodi on laadukasta ja selkeää, mutta nopeus on hieman keskivertoa alempana.

Torstai 12.5.2022

Työtehtävät keskittyivät tänään käyttöliittymän ja käyttöliittymälogiikan kehitykseen. Edellisen päivän palautteessa tuli esille nopeus, joten ajattelinkin tänään kokeilla lisätä nopeutta ohjelmakoodin ja ratkaisujen toteuttamiseen.

Työpäivän aikana sain kyllä paljon aikaiseksi, mutta koin olevani epävarma omista toteutuksistani ja työn laadusta.

Perjantai 13.5.2022

Työpäivän aluksi käytin tunnin koodin läpikäymiseen ja suunnittelin parempia ratkaisuja. Tämän jälkeen asetin päivän tavoitteeksi saada refaktoroitua noin 1000 riviä koodia siistimmäksi ja hallittavammaksi.

Lopulta päivän päätteeksi tavoite saavutettiin ja koodista tuli mielestäni huomattavasti luettavampi ja hallittavampi. Tämä auttoi löytämään koodista myös muutamia virheitä, jotka tuli samalla korjattua.

3.5.2 Ketterä ohjelmistokehitys

Ketterä ohjelmistokehitys (engl. agile software development) on syntynyt useiden eri ketterien menetelmien pohjalta ja se määriteltiin vuoden 2001 pidetyssä kokouksessa, josta syntyi ketterä manifesti. (Highsmith 2001.)

Manifestissa tuodaan esille ketterän ohjelmistokehityksen ideologiaa sekä ketteriä periaatteita, jotka antavat ohjeistuksia menetelmien noudattamiseksi.

Kyseessä on siis filosofia ja joukko erilaisia toimintatapoja ja käytänteitä, joita voidaan soveltaa ohjelmistokehityksessä.

Ketterä ohjelmistokehitys on myös aikoinaan ottanut paljon vaikutteita Lean-ajattelusta. Kyseessä on Toyotan määrittelemä johtamisfilosofia ja joukko periaatteita, joka on saanut alkuunsa autoteollisuuden tuotannon tuotekehityksen eri menetelmistä 1900-luvun alkupuolella. (Luukkainen 2022d)

Kanban on yksi Lean-periaatteen mukainen visuaalinen projektinhallintamenetelmä ja -työkalu, jota on myös laajasti otettu käyttöön ketterässä ohjelmistokehityksessä. (Koskinen 2021.)

Ketterien menetelmien yhteisiä piirteitä ovat nopeat kehityssyklit, läpinäkyvyys, nopea reagointi muutoksiin, suora viestintä ja toimivan ohjelmiston ensisijaisuus.

Olemme myös aikaisemmissa teoriaosuuksissa tuonut esille tunnettuja ketterien menetelmien parhaita käytänteitä, joita olivat mm. refaktorointi ja pariohjelmointi.

Koska kyseessä on erittäin laaja aihe, niin tässä teoriaosuudessa otetaan esille yksi yleisesti käytetty menetelmä, joka esitellään lyhyesti.

Scrum

Scrum on yksi suosituimmista ja käytetyimmistä ketterän projektinhallinnan menetelmäkehys/prosessimalli. Kyseessä on iteratiivinen ja inkrementaalinen menetelmä, jossa kehitystyö tapahtuu 1–4 viikon jaksoissa (iteraatioissa). Scrum terminologian mukaan näitä kehitysjaksoja kutsutaan sprinteiksi. (Luukkainen 2022e)

Jokaisen sprintin alussa kehityksestä vastaava tiimi valitsee priorisoidusta listasta (engl. backlog) ne vaatimukset, joita sprintin aikana tulisi toteuttaa itseorganisoidusti. Edellä mainittua priorisoitua listaa eli backlogia hallinnoi ja priorisoi vain tuotteen omistaja (engl. product owner). (Luukkainen 2022e)

Jokaiseen tiimiin kuuluu myös Scrum master, joka vastaa mm. Scrum sääntöjen noudattamisesta, sidosryhmien kanssa kommunikoimisen, työskentelytapojen parantamisesta sekä esteiden poistamisesta. (Luukkainen 2022e)

Sprintin aikana jokainen päivä aloitetaan lyhyellä palaverilla, jota kutsutaan daily scrum. Palaverissa jokainen kehitystiimin jäsen vastaa seuraaviin kysymyksiin:

- Mitä olet tehnyt edellisen kokouksen jälkeen?
- Mitä aiot tehdä seuraavaksi?
- Onko tiedossasi esteitä, jotka vaikuttavat työhösi?

Jokaisen sprintin lopuksi järjestetään sprinttikatselmus, jossa kehitystiimi esittelee saavutuksensa.

Myös kehitystiimin kesken järjestetään retrospektiivi, jossa pyritään vastaamaan seuraaviin kysymyksiin:

- Mikä meni hyvin?
- Mitä pitäisi parantaa seuraavassa sprintissä?

3.5.3 Analyysiraportti

Keskiviikon palautteen perusteella tähän mennessä tuottamani koodi on ollut laadukasta ja selkeää, mutta nopeus on ollut hieman keskivertoa alempana.

Tähän mennessä tyylini toteuttaa koodia on ollut sellainen, että monimutkaisemmissa toteutuksissa saatan ensiksi suunnitella ratkaisua mm. visuaalisilla esimerkeillä, jonka pohjalta lähden ratkaisemaan ongelmaa. Yksinkertaisten toteutusten kohdalla lähden kokeilemaan suoraan koodaamalla.

Tarkastellaan asiaa vielä kalenterissa olevien merkintöjen pohjalta, että miten olen suoriutunut ohjelmakoodin tuottamisessa.

Kuvasta 7 voi havaita, että sovelluksen toiseksi laajimman kokonaisuuden ohjelmointiin käytetystä ajasta alle 40 % on kulunut parannusehdotusten korjauksiin ja refaktorointiin.

Viikon aikana ohjelmistokoodin tuottamiseen tuli käytettyä 26,5 h, joka keskittyi projektin toiseksi laajimman kokonaisuuden toteuttamiseen. Tästä ajasta noin 55 % kului koodin parannusehdotusten korjauksiin ja refaktorointiin. Kaiken kaikkiaan kyseisen kokonaisuuden toteuttamiseen on käytetty tähän mennessä yhteensä noin 41 % arvioidusta ajasta. Ja tästä kokonaisajasta taas noin 38 % on kulunut koodin parannusehdotusten korjauksiin ja refaktorointeihin.

Kuva 7. Lähdekoodin korjaustoimenpiteisiin ja refaktorointiin kulutettu aika.

3.6 Viikko 6

3.6.1 Päiväkirjamerkinnät

Maanantai 16.5.2022

Viikko alkoi tuttuun tapaan etätyön merkeissä.

Tämän päivän työtehtäviä:

- katselmoinnissa esiin tulleet lähdekoodin korjaustoimenpiteet
- uusien toiminnallisuuksien käyttöliittymän ja logiikan kehitystyöt
- palaveri
- API-rajapintadokumentaation päivittämistä
- testauksessa olevan integraatiohaaran ongelmanselvittelyä.

Tiistai 17.5.2022

Toimistolle saavuttuani aloin valmistella molemmat tietokoneeni ja testilaitteeni valmiuteen, sillä minulla oli sovittu aikainen palaveri heti aamulle.

Palaverissa testailimme mobiilin ja taustapalvelimen toimivuutta. Tämän lisäksi suunnittelimme uutta ominaisuutta, jota pitäisi alkaa kehittämään.

Lounaan jälkeen aloin korjaamaan testauksessa olevan integraatiohaaran ongelmaa, jonka syyn sain selville edellisenä työpäivänä. Korjauksen jälkeen tein lähdekoodista uuden version testattavaksi.

Loppupäivä kului uusimman kehityshaaran käyttöliittymän ja logiikan lähdekoodin korjaustoimenpiteillä ja kehitystöillä. Päivän aikana ei esiintynyt suurempia ongelmia ja selvisin itsenäisesti.

Keskiviikko 18.5.2022

Päivä kului seuraavien tehtävien parissa:

- uusimman kehityshaaran käyttöliittymän ja logiikan korjaustoimenpiteet ja kehitystyöt
- toisen kehityshaaran käännöslogiikan kehitystyöt
- kaksi palaveria

Torstai 19.5.2022

Tämän päivän aikana testasimme jälleen mobiilin ja taustapalvelimen toimivuutta, sillä sain edellisenä päivänä käännöslogiikan valmiiksi testattavaksi. Ratkaisu toimi hyvin Androidilla, mutta jostain syystä iPhonella esiintyi ongelmia.

Loppupäivän työtehtäviin kuului tuttuun tapaan uusimman kehityshaaran käyttöliittymän ja logiikan korjaustoimenpiteet sekä kehitystyöt, joiden toimivuutta testailin myös Androidilla ja iOS:llä.

Päivän päätteeksi meillä oli vielä henkilöstöpalaveri.

Perjantai 20.5.2022

Tämän päivän tavoitteena oli korjata katselmoinnissa esiintyneet virheet ja luoda uusia tikettejä projektinhallinnan järjestelmään. Tämä tavoite saavutettiin ennen lounasta ja loppupäivä kului opintoihin.

3.6.2 Ketterä oppiminen

Ketterä oppiminen on ongelmanratkaisua ja arvoa tuottavaa itseohjautuvaa jatkuvaa oppimista, jota tehdään työn ohella. Ketterässä oppimisessa on kyse siis itsensä johtamisesta, uteliaisuudesta ja eri toimintatapojen soveltamista oikea-aikaisesti tekemisen ohella, jolla pyritään synnyttämään uutta tietoa ja osaamista. (Otala & Meklin 2021.)

Ketterä oppiminen on syntynyt työelämän muutosten seurauksena, jolloin itse työ asettaa jo tekijälleen oppimistavoitteen ja tarpeen. (Jarenko 2019.) Kuten tämän työn alussa mainittiinkin, että nopeasti muuttuvilla aloilla eilen opittu työ tai taito saattaa olla jo käyttökelvotonta huomisen tuotteessa tai palvelussa.

Teknologia-alan yrityksissä osaaminen on keskeisessä roolissa yrityksen menestymisen kannalta. Koska ketterä oppiminen mahdollistaa uuden tiedon ja osaamisen synnyttämisen, voidaan tätä tietoa soveltaa ja käyttää yrityksessä, jolloin syntyy myös kilpailuetua. Ketterän oppimisen kirjassa mainitaan, että tällöin kyse olisi strategisen osaamisen muotoilusta. (Otala & Meklin 2021.)

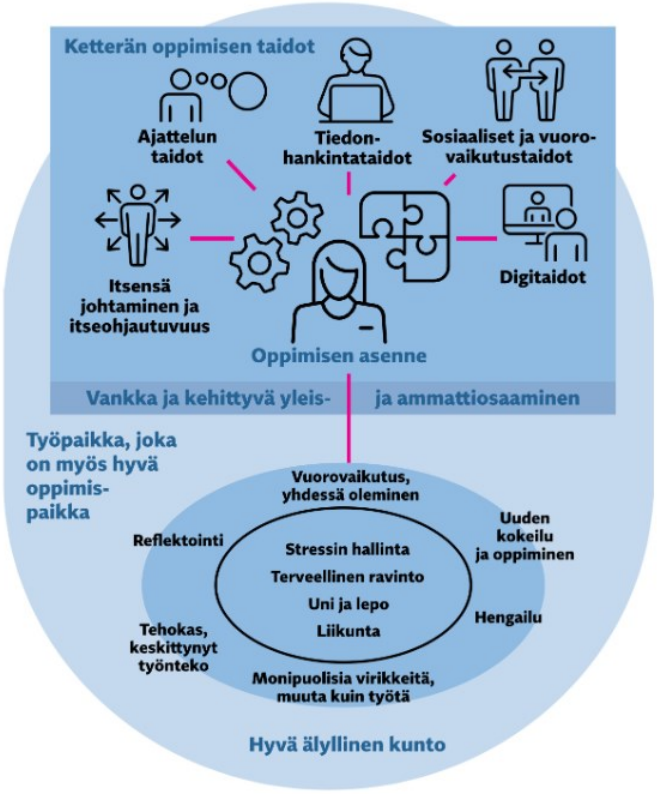
> *Ei ole järkeä palkata fiksuja ihmisiä ja kertoa heille, mitä pitää tehdä; me palkkaamme fiksuja ihmisiä, jotta he voivat kertoa meille, mitä tehdään. Steve Jobs, Applen perustaja ja toimitusjohtaja.*
>
> (Otala & Meklin 2021.)

Myös teknologiajätti Google on tukenut ketterää oppimista jo vuodesta 2004, kun perustajat Sergey Brin ja Larry Page mainitsivat listautumisannin kirjeessään, että työntekijät saavat käyttää 20 % työajastaan uuden oppimiseen. (Page & Sergey 2004.)

Aikaisemmin mainittiin, että ketterässä oppimisessa on kyse itsensä johtamisesta. Tällä tarkoitetaan kykyjä tunnistaa ja kehittää omaa osaamistaan jatkuvasti epävarmassa ja muuttuvassa ympäristössä sekä tietenkin huolehtia omasta jaksamisesta. Jokainen on siis itse vastuussa omasta osaamisestaan ja oppimisestaan. (Kallonen & Kuhmonen 2021.)

Itsensä johtamiseen liittyy myös motivaatio, joka voi olla ulkoista tai sisäistä. Ulkoisia motivaation lähteitä ovat mm. suurempi palkka, edut, arvostus tai julkisuus. Sisäinen motivaatio tarkoittaa taas sitä, että työ itsessään on kiinnostavaa ja mielekästä, jossa yksilö voi itse vaikuttaa oman työn tekemiseen. Tällöin sisäinen motivaatio ohjaa tekemään ja oppimaan asioita, joista on kiinnostunut ja antaa työlle merkityksen. (Otala & Meklin 2021.)

Kuva 8 havainnollistaa vaatimuksia ja taitoja, jotka edesauttavat ketterää oppimista.



Kuva 8. Ketterän oppijan vaatimukset ja taidot. (Otala & Meklin 2021.)

3.6.3 Analyysiraportti

Tässä analyysissa pohditaan tämä viikon suoriutumista sekä ketterän oppimisen taitoja.

Tämän viikon työtehtävät koostuivatkin monipuolisista tehtävistä. Työtehtävät olivat sopivan haastavia ja mielekkäitä, joten selvisin niistä myös hyvin itsenäisesti. Viikon aikana ei myöskään esiintynyt ylitsepääsemättömiä haasteita, vaikka joitain ongelmia tietenkin oli. Tuntuukin siltä, että tämä viikko kului poikkeuksellisen leppoisasti. Ehkä tämä on signaali siitä, että seuraavalla viikolla saan tämänkin viikon edestä sammutella tulipaloja.

Pohditaanpa sitten aloittelevan ohjelmistokehittäjän näkökulmasta teoriaosuudessa mainittuja ketterän oppijan taitoja.

Ehkä suurimmaksi ja tärkeimmäksi taidoksi aloittelevana ohjelmistokehittäjänä näen tiedon hankintataidot. Tämän tärkeys on vain kasvanut viikkojen myötä, sillä kehitystehtävien aikana ilmestyy jatkuvasti virheitä ja uusia ongelmia, joita pitää selvitellä. Sanoisin, että itsellä olisi tässäkin vielä parannettavaa.

Toiseksi tärkeämmäksi taidoksi näkisin itsensä johtamisen ja itseohjautuvuuden taitoja, varsinkin matalan hierarkian kulttuurissa. Työpaikallani on melko vapaat kädet tehdä asioita, vaikka olenkin vielä harjoittelija ja saan tarpeen mukaan ohjausta. Kehitystyö on kuitenkin melko itsenäistä työtä, vaikka työskentelemmekin tiimissä.

Kolmanneksi tärkeäksi taidoksi näen kyvyn ylläpitää omaa jaksamista. Näiden päiväkirjamerkintöjen aikana olen kokenut ylä -ja alamäkeä oman jaksamisen kanssa. Varsinkin töiden ja harjoittelun alussa uutta tietoa tuli runsaasti ja piti omaksua uudenlainen työskentelytapa. Näiden lisäksi opinnäytetyön kirjoittamistakin piti opetella samalla. Myös riittävällä unella ja liikkumisella on ollut positiivisia vaikutuksia työn tehokkuuteen ja motivaation löytämiseen.

Lopuksi nostaisin esille sosiaaliset ja vuorovaikutustaidot. Nämä ovat ehdottomasti tärkeitä taitoja aloittelevan ohjelmistokehittäjän kannalta. Asioiden

leikittely ja rohkea kysyminen yksinkertaisimmistakin asioista ovat olleet erittäin opettavaisia itselleni. Myös eri sidosryhmien kanssa vuorovaikuttaminen on osa työtä.

3.7 Viikko 7

3.7.1 Päiväkirjamerkinnät

Maanantai 23.5.2022

Maanantai lähti käyntiin tuttuun tapaan etätyön merkeissä.

Tämän päivän tekemisessä oli taas sitä tuttua kaoottista viikonlopun jälkeistä “kun mikään ei taas toimi” -meininkiä. Eli työpäivän ensimmäiset kuusi tuntia keskittyivät eri kehityshaarojen ongelmienselvittelyyn ja ihmettelyyn.

Lopulta päivän päätteeksi pääsin hetkeksi työstämään uusimman kehityshaaran toiminnallisuuksia.

Tiistai 24.5.2022

Tämän päivän tavoitteena oli keskittyä mobiilisovelluksen uusimman kehityshaaran toiminnallisuuksien kehittämiseen.

Viime päivät ovatkin koostuneet muista tehtävistä, joten en ole ehtinyt viemään tätä projektia hirveästi eteenpäin. Kehitystyöt keskittyivät tänään käyttöliittymän API-pyyntöjen logiikan koodaamiseen ja refaktoroimiseen.

Päivän päätteeksi pidin vielä vajaan tunnin mittaisen tietoiskun ja vapaamuotoisen keskustelun puhtaan koodin ja koodikatselmuksen teemoista. Pienen alkujännityksen jälkeen tietoisku ja keskustelu kehittyikin erittäin mielenkiintoiseksi, eikä varattu aikakaan tahtonut riittää tähän.

Keskiviikko 25.5.2022

Tämän päivän agendana oli selvittää mobiilisovelluksen iOS puolen ongelmat. Tehtäviin kuului paljon googlettelua ja selvittelyä vanhemman ohjelmistokehittäjän kanssa.

Päivän päätteeksi löysimme potentiaalisen ratkaisun ongelmaan, mutta pääsemme testailemaan sen toimivuutta vasta seuraavalla viikolla back end-kehittäjien kanssa.

Tämän päivän aikana jouduin myös pyytämään apua kokeneemmalta kehittäjältä.

Torstai 26.5.2022

Helatorstai.

Perjantai 27.5.2022

Viikon viimeinen päivä ja myös tämän oppimispäiväkirjan viimeinen päiväkirjamerkintä.

Päivän tavoitteena oli päivittää ja valmistella projektinhallintajärjestelmässä olevia tikettejä tuleville viikoille, sillä ensi viikosta lähtien saan kesätyöntekijän parikseni projektiin.

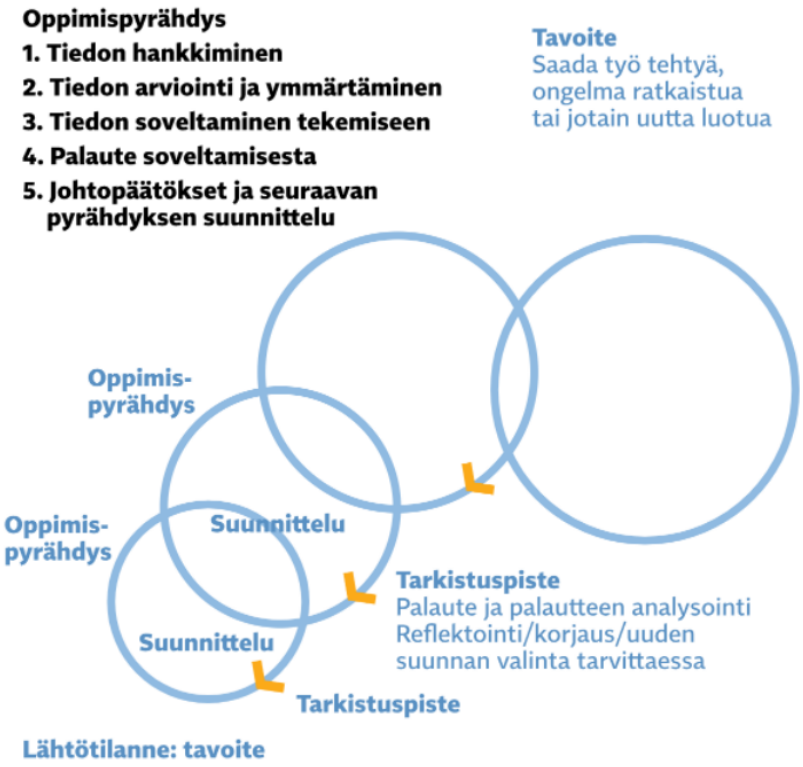
Tavoite saavutettiin ja loppupäivä keskittyi opintoihin.

3.7.2 Ketterän oppimisen menetelmiä

Tässä teoriaosuudessa tuodaan lyhyesti esille keskeisiä ketterän oppimisen menetelmiä, joita Otala ja Meklin käsittelevät kirjassaan Ketterä oppiminen 2, Strategiasta käytäntöön.

Oppimispyrähdys

Oppimispyrähdys on tiedonhankintaan soveltuva iteratiivinen prosessimalli. Kuvassa 9 on kuvattu Otalan ja Meklinin esittämä oppimispyrähdys.



Kuva 9. Oppimispyrähdys. (Otala & Meklin 2021.)

Kysyminen

Kysyminen ja kyseenalaistaminen on yksi tehokkaimpia tapoja oppia, kehittyä ja löytää vastauksia. Myös Googlea johtanut Eric Schmidt on aikoinaan todennut, että Googlea johdetaan kysymyksillä, eikä niinkään vastauksilla. (Otala & Meklin 2021.)

> *One of the really tough things is figuring out what questions to ask. Once you figure out the question, then the answer is relatively easy.* -Elon Musk (Quote by Elon Musk: “One of the really tough things is figuring out ...” s.a.)

Tiedon jakaminen

Hyvien ja toimivien käytäntöjen jakaminen koko organisaatiolle on tärkeä osa ketterää työssä tapahtuvaa oppimista sekä yrityskulttuuria.

Tiedon, kokemuksen ja oppien jakaminen auttaa myös tuomaan hiljaista tietoa, jota voidaan hyödyntää organisaatiossa.

Tiedon jakaminen auttaa tuomaan myös erilaisia näkökulmia esille.

Mentoroinnissa tiedon jakaminen on myös keskeisessä roolissa.

Palaute

Palautteen antaminen, vastaanottaminen ja hyödyntäminen on tehokas keino laajentaa tietämystä sekä tunnistaa puutteet sekä vahvuudet. Palaute on siis tärkeä työkalu hiljaisen tiedon vuorovaikuttamiseen ja osa oppimista.

Reflektointi

Reflektointi on pitkään ollut yksi tehokas keino oppia uutta ja menetelmänä tehokas hiljaisen tiedon jakamisessa. Reflektoinnissa on kyse hetkessä pysähtymisestä ja tekemisen analysoimisesta. (Otala & Meklin 2021) Menetelmää hyödynnetään myös ketterän menetelmän Srum:issa nimellä retrospektiivi. Itsereflektointia käsittelimme myös lyhyesti ensimmäisessä analyysiraportissa.

Virheistä oppiminen

Epäonnistumiset ja virheet tarjoavat erittäin tarkeitä mahdollisuuksia menestyä ja oppia uutta. (Rytkönen 2018.)

Epäonnistumisien ja virheiden analysoinneilla voidaan saada esille runsaasti uutta tietoa, jota voidaan hyödyntää seuraavien epäonnistumisien ja virheiden minimoimiseksi.

Myös maailman rikkaimmaksi ihmiseksi tituleerattu moniyrittäjä Elon Musk on lukuisten epäonnistumisien avulla saavuttanut menestyksensä. (101 Elon Musk Quotes That Will Inspire Your Success | Totempool s.a.)

Mentorointi

Mentoroinnissa on kyse avoimesta ja luottamuksellisesta valmentavasta suhteesta, jossa kokeneempi työntekijä (mentori) neuvoo ja opastaa kehityshaluista ja kokemattomampaa henkilöä (aktori/kisälli) työssään.

Kokeilut

Kyseessä on iteratiivinen prosessimalli, jossa hyödynnetään useita eri ketterän oppimisen menetelmiä. Kuvassa 10 on havainnollistettu Otalan ja Meklinin kokeiluprosessia.

Kuva 10. Kokeiluprosessi. (Otala & Meklin 2021.)

3.7.3 Analyysiraportti

Tässä viimeisessä analyysiraportissa pohditaan teoriaosuudessa esille tuotuja menetelmiä, joita on hyödynnetty työn ohella ammatillisen oppimisen tukena.

Kun tarkastellaan viimeisen seitsemän viikon suoritusta, ovat tärkeimmiksi oppimisen tukemisen menetelmiksi mielestäni nousseet järjestetyt tietoiskut, palautteet, mentorointi, kysyminen ja reflektointi.

Tietoiskut olivat mielenkiintoisia ja avartavia kokemuksia, joita työn ohella järjestettiin esimiehen aloitteesta. Tietoiskut koostuivat oppimispäiväkirjan teoriaosuuksien teemoista, joista valmistelin tiivistettyjä presentaatioita ja joita esittelin pienelle yleisölle yrityksen sisällä. Oppimista tapahtui ensin tiedon hankkimisella ja analysoimisella, jonka jälkeen tätä tietoa jaettiin osallistujille. Tämän jälkeen oppimista ja uutta tietoa syntyi osallistujien välisellä vuorovaikutuksella, sillä esitetyt teemat herättivät keskusteluja ja toivat erilaisia näkökulmia esille. Esityksistä sain myös positiivista palautetta, sillä nämä olivat tuoneet uutta tietoa osalle osallistujille. Tietoiskussa hyödynnettiin useita eri ketterän oppimisen menetelmiä.

Palautteiden avulla olen saanut nopeasti tietoon oman kehitykseni tilanteen ja mahdolliset puutteet. Palautetta saa myös nopeasti kysymällä, jota olen hyödyntänyt oppimisen tukena.

Mentorointi on ollut erittäin tärkeässä roolissa uuden oppimisen tukena. Olen onnekseni saanut loistavan mentorin, joka on helpottanut huomattavasti työhön mukaan pääsemisellä. Olen saanut aina apua, kun tarvitsen ja tämä on poistanut pelot myös siitä, etten ole jäänyt jalkoihin pyörimään.

Reflektointia on taas syntynyt viikoittaisten analyysien toimesta, ja menetelmä on osoittautunut erittäin tehokkaaksi oppimisen ja itsetuntemisen työkaluksi.

4 Yhteenveto ja pohdinta

Tämän opinnäytetyön tarkoituksena oli tarkastella ja tukea aloittelevan ohjelmistokehittäjän oppimista ja sen kehitystä mobiilisovelluksen jatkokehitystyön ohella. Oppimisprosessin tukemisen ja tarkastelun lisäksi tavoitteena oli perehtyä ohjelmistotuotannon keskeisempiin käytäntöihin, joita projektityössä esiintyi. Lisäksi tavoitteena oli tuottaa kuva ohjelmistokehittäjän työstä ja työtehtävistä sekä vastata seuraaviin kysymyksiin:

- Onko oppimispäiväkirjan kirjoitusten jälkeen tapahtunut ammatillista kehitystä?
- Minkälaisia menetelmiä tai työkaluja hyödynnettiin ammatillisen kehityksen tukena?
- Minkälaisia ominaisuuksia vaaditaan aloittelevalta ohjelmistokehittäjältä?

Opinnäytetyön oppimisprosessin aikana havaittiin, että päiväkirjamuotoinen tutkielma on antanut opiskelijalle hyvän oppimiskokemuksen työn ohella.

Viikkojen 1–5 välillä perehdyttiin ohjelmistotuotannon käytäntöihin ja aiheisiin, joita projektityössä esiintyi ja koettiin tärkeäksi. Projektin osalta ohjelmistotuotannon vaiheet keskittyivät lähinnä suunnittelun ja toteutuksen osa-alueisiin ketteriä menetelmiä hyödyntäen, joten aiheet rajattiin näiden vaiheiden mukaisesti. Jokaisella viikolla perehdyttiin teorian pohjalta tiettyyn aiheeseen, jota projektityössä esiintyi. Tämän lisäksi aiheista pidettiin tietoiskuja yrityksen sisällä.

Päiväkirjamerkinnät antavat jonkinlaisen käsityksen ohjelmistokehittäjän päivittäisistä työtehtävistä ja arjesta, vaikka merkinnöissä ei avata työtehtäviä sen laajemmin.

Koen kehittyneeni myös ammatillisesti opinnäytetyöprosessini aikana. Työni alussa minulla ei ollut oikeastaan lainkaan työkokemusta ja tarvitsin tukea ja ohjeistusta työtehtävien suorittamiseen. Olen näiden viikkojen aikana todennut, että olen kyennyt toimimaan paljon itsenäisemmin ja saanut itsevarmuutta

työskentelyyni. Tukea ja ohjeistusta tulen tarvitsemaan vielä, sillä matkani asiantuntijaksi on vasta alkumetreillä.

Ehkä tärkeimmiksi menetelmiksi ja työkaluiksi ammatillisen kehityksen tueksi näen viimeisessä analyysissakin esille tuodun mentoroinnin ja tietoiskut sekä itse opinnäytetyön. Päiväkirjamerkintöjen kirjaaminen, analysoiminen ja peilaaminen pieniin teoriaosuuksiin itsereflektoimalla on ollut erittäin tehokas tapa tunnistaa ja kasvattaa omaa ammatillista kehitystä ketterästi työn ohella.

Aloittelevalta ohjelmistokehittäjältä vaaditaan oma-aloitteisuutta, itsenäisyyttä, uteliaisuutta, itsensä johtamisen taitoja, tiedon hankintataitoja ja tietenkin halua ja kykyä oppia jatkuvasti uutta. Viikon 6. analyysissä käytiin myös läpi taitoja, joita itse koin tärkeäksi aloittelevan ohjelmistokehittäjän näkökulmasta.

Mielestäni opinnäytetyön kaikki vaatimukset saavutettiin, ja olen löytänyt työkaluja, joita tulen varmasti hyödyntämään tulevaisuudessakin uuden oppimisessa. Opinnäytetyön kirjoittamisen aikana olen oppinut paljon uutta ja kyennyt tunnistamaan omia heikkouksia sekä vahvuuksia. Lopuksi voin vielä todeta, että päiväkirjamuotoinen opinnäytetyö on osoittautunut loistavaksi työkaluksi ketterään oppimiseen, jota voin lämpimästi suositella jokaiselle alalle pyrkivälle.

Lähteet

101 Elon Musk Quotes That Will Inspire Your Success | Totempool s.a. URL: https://totempool.com/blog/elon-musk-quotes/. Accessed: 5 June 2022.

Aronen, A. 2017. Opinnäytetyö (AMK) Tietojenkäsittelyn koulutusohjelma. Accessed: 24 April 2022.

Böckeler, B. & Siessegger, N. 2020. On Pair Programming. URL: https://martinfowler.com/articles/on-pair-programming.html. Accessed: 4 June 2022.

Campbell, D. 2019. The Real Reason it's Difficult to Write Clean Code | by Drew Campbell | Better Programming. URL: https://betterprogramming.pub/the-real-reason-its-difficult-to-write-clean-code-gamedevunboxed-5ccdb06ca33f. Accessed: 11 May 2022.

Clean, high quality code: a guide on how to become a better programmer s.a. URL: https://www.butterfly.com.au/think-pieces/clean-high-quality-code-a-guide-on-how-to-become-a-better-programmer/. Accessed: 11 May 2022.

Cohn, M. 2004. Project Advantages of User Stories as Requirements. URL: https://www.mountaingoatsoftware.com/articles/advantages-of-user-stories-for-requirements. Accessed: 23 April 2022.

Danjou, J. 2021. Pull Request vs. Merge Request: What's the Difference? URL: https://blog.mergify.com/pull-request-vs-merge-request-whats-the-difference/. Accessed: 10 June 2022.

Fowler, M. 1998. 1998, Refactoring: Doing Design After the Program Runs. URL: https://martinfowler.com/tags/1998.html. Accessed: 4 May 2022.

Fowler, M. s.a. Refactoring. URL: https://refactoring.com/. Accessed: 10 June 2022.

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Talentum. Accessed: 23 April 2022.

Hannonen, K. 2012. Henkilökohtainen SWOT-analyysi | Oppimisen osasia. URL: https://karihanse.wordpress.com/2012/11/14/henkilokohtainen-swot-analyysi/. Accessed: 23 April 2022.

Highsmith, J. 2001. History: The Agile Manifesto. URL: https://agilemanifesto.org/history.html. Accessed: 17 June 2022.

Jarenko, K. 2019. Ketterässä oppimisessa tavoitteena ei ole oppiminen - Filosofian Akatemia Oy. URL: https://filosofianakatemia.fi/blogi/ketterassa-oppimisessa-tavoitteena-ei-ole-oppiminen/. Accessed: 19 May 2022.

Kallonen, T. & Kuhmonen, A. 2021. Jatkuva oppiminen : työelämän tärkein taito. Kauppakamari. URL: https://kauppakamaritieto.fi/. Accessed: 23 April 2022.

Koskinen, I. 2021. Mikä on Kanban? Katsaus menetelmään ja sen käyttöön ketterässä projektinhallinnassa - PSA. URL: https://psa.visma.fi/blog/mika-on-kanban-katsaus-menetelmaan-ja-sen-kayttoon-ketterassa-projektinhallinnassa/. Accessed: 17 June 2022.

Luukkainen, M. 2022a. Ohjelmistotuotanto avoin yliopisto 2022, Osa 2. URL: https://ohjelmistotuotanto-hy-avoin.github.io/osa2/. Accessed: 23 April 2022.

Luukkainen, M. 2022b. Ohjelmistotuotanto avoin yliopisto 2022, Osa 4. URL: https://ohjelmistotuotanto-hy-avoin.github.io/osa4/. Accessed: 23 April 2022.

Luukkainen, M. 2022c. Ohjelmistotuotanto avoin yliopisto 2022, Osa 3. URL: https://ohjelmistotuotanto-hy-avoin.github.io/osa3/. Accessed: 4 June 2022.

Luukkainen, M. 2022d. Ohjelmistotuotanto avoin yliopisto 2022, Osa 5. URL: https://ohjelmistotuotanto-hy-avoin.github.io/osa5/. Accessed: 17 June 2022.

Luukkainen, M. 2022e. Ohjelmistotuotanto avoin yliopisto 2022, Osa 1. URL: https://ohjelmistotuotanto-hy-avoin.github.io/osa1/. Accessed: 17 June 2022.

Martin, R.C. 2008. Clean Code: A Handbook of Agile Software Craftsmanship - Robert C. Martin - Google-kirjat. URL: https://books.google.fi/books?id=_i6bDeoCQzsC&printsec=frontcover&dq=Clea

n+Code:+A+Handbook+of+Agile+Software+Craftsmanship&hl=fi&sa=X&redir_esc=y#v=onepage&q&f=false. Accessed: 4 May 2022.

Otala, L. & Meklin, S. 2021. Ketterä oppiminen. 2, Strategiasta käytäntöön. Kauppakamari. Accessed: 23 April 2022.

Page, L. & Sergey, B. 2004. 2004 Founders' IPO Letter - Investor Relations - Alphabet. URL: https://abc.xyz/investor/founders-letters/2004-ipo-letter/. Accessed: 19 May 2022.

Quote by Elon Musk: "One of the really tough things is figuring out ..." s.a. URL: https://www.goodreads.com/quotes/9554987-one-of-the-really-tough-things-is-figuring-out-what. Accessed: 5 June 2022.

Rytkönen, M. 2018. Virheistä oppiminen - organisaation mahtava mahdollisuus menestyä - LMI Finland. URL: https://www.lmi.fi/virheista-oppiminen-organisaation-mahtava-mahdollisuus-menestya/. Accessed: 5 June 2022.

Sironen, L. 2021. Asiakkaan itsereflektio fysioterapiassa. Accessed: 23 April 2022.

Vihlman, R. 2022. Koodipohjan refaktorointi ja ylläpi-dettävyys. Accessed: 20 May 2022.