Aku Korhonen

Mobiilisovellus painehaavojen ennaltaehkäisyyn pyörätuolin käyttäjillä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologia

Insinöörityö

8.11.2023

Tiivistelmä

Tekijä:	Aku Korhonen
Otsikko:	Mobiilisovellus painehaavojen ennaltaehkäisyyn pyörätuolin käyttäjillä
Sivumäärä:	39 sivua
Aika:	8.11.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintäteknologia
Ammatillinen pääaine:	Hyvinvointi- ja terveysteknologia
Ohjaajat:	Yliopettaja Mikael Soini Projektin CTO Marko Mattila

---

Insinöörityössä oli tarkoituksena käydä läpi mobiilisovelluksen kehityksen vaiheet toimintatutkimuksen metodeja apuna käyttäen. Mobiilisovelluksen kehityksen tavoitteena oli saavuttaa projektissa ennalta määritellyt ominaisuudet. Työ tehtiin osana Aalto-yliopiston research-to-business-projektia. Sovelluksen käyttökohde oli painehaavojen ennaltaehkäisy pyörätuolin käyttäjillä.

Sovelluksen kehitystyö tehtiin ketterän kehityksen sprintteinä. Sprinttien läpikäynnissä käytettiin apuna toimintatutkimuksen spiraalimallia. Jokainen sprintti aloitettiin sen suunnittelulla, jossa päätettiin, mitä kehitysvaiheessa tultiin tekemään. Tämän jälkeen kehitystyötä tehtiin kaksi viikkoa, jonka jälkeen sprinttikatselmuksen avulla havainnoitiin ja reflektoitiin tehtyä työtä. Näiden vaiheiden jälkeen suunnitelmaa uusittiin ja uusi sprintti aloitettiin.

Mobiilisovellus koostui yhdestä sivusta, jonka kautta sovellusta käytettiin. Sovellukseen lisättiin lämpökartta painematosta, jonka päällä sovelluksen käyttäjä istui. Sovellus ja painematto yhdistettiin Bluetooth Low Energyn avulla. Sovellukseen lisättiin neljä algoritmia, jotka käyttivät toimintaansa lämpökartan painearvoja. Algoritmit mittasivat käyttäjän painetta kumulatiivisesti, laskivat painekeskipisteen, sekä korkeimmat painekohdat kartasta. Tämän lisäksi arvioitiin käyttäjän istuma-asento painekeskipisteen avulla. Nämä kaikki tiedot näytettiin sovelluksessa käyttäjälle.

Insinöörityön lopputulos koostuu tästä raportista, jota voidaan tulevaisuudessa käyttää ensimmäisenä iteraationa toisessa toimintatutkimuksessa, sekä ensimmäisestä sivusta laajempaan mobiilisovellukseen, jonka kehitystyö jatkui tämän insinöörityön jälkeen.

Avainsanat:	painehaava, toimintatutkimus, mobiilisovellus, Bluetooth Low Energy, painematto

---

Abstract

Author:	Aku Korhonen
Title:	Mobile Application for Pressure Ulcer Prevention within Wheelchair Users
Number of Pages:	39 pages
Date:	8 November 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Health Technology
Supervisors:	Mikael Soini, Principal Lecturer Marko Mattila, Project CTO

In this engineering study, the aim was to go through the stages of mobile application development using action research methods. The goal of the mobile application development was to achieve certain predefined features in the project. The work was carried out as part of Aalto University's research-to-business project. The application was intended for the prevention of pressure sores in wheelchair users.

Development of the application was done in agile development sprints. The spiral model of action research was used in going through each sprint. For each sprint, a plan was first outlined and then developed. After two weeks of development, the work was observed and reflected upon through a sprint review. After these phases, the plan was revised, and a new sprint was started.

The mobile application consisted of a single page through which the application was used. A heatmap of values from the pressure mat was added to the application. The application and the pressure mat were connected via Bluetooth Low Energy. Four algorithms were added to the application, which used the pressure values from the heatmap in their operations. The algorithms measured the user's pressure over an extended period, calculated the center of pressure, and identified the highest-pressure points on the map. Additionally, the user's sitting position was assessed using the pressure center. All this information was displayed in the application for the user.

The result of the study is a report on the development of the mobile application, which can be used in the future as the first iteration in another action research project, as well as the first page of a more extensive mobile application, the development of which continued after the current study was concluded.

Keywords: pressure ulcer, action research, mobile application, Bluetooth Low Energy, pressure mat

Sisällys

Lyhenteet

API: *Application programming interface.* Ohjelmointirajapinta on tapa, jolla ohjelmistot tarjoavat tietoa muille järjestelmille.

BLE: *Bluetooth Low Energy*. Lyhyenmatkan langaton yhteysteknologia, jota käytetään useissa nykyajan laitteissa.

DB: *DataBase.* Tietokanta, jossa tietoa voidaan säilyttää tai josta sitä voidaan hakea.

GAP: *Generic Access Profile*. Kontrolloi BLE-laitteiden yhteyksiä ja näkyvyyttä muille laitteille.

GATT: *Generic Attribute Profile*. Määrittää tavan, jolla kaksi BLE-laitetta lähettää ja vastaanottaa informaatiota.

HCI: *Host Controller Interface*. Osa BLE-pinoa, joka kuljettaa komentoja pinon eri elementtien välillä.

ID: *Identifier*. Tietojenkäsittelyssä annettava yksilöllinen tunniste.

IDE: *Integrated Development Environment*. Ohjelmointiympäristö, jossa ohjelmointi on tehty helpommaksi erilaisten ominaisuuksien avulla.

IIR: *Infinite Impulse Response.* Digitaalisen signaalinsuodatuksen muoto.

MAC: *Media Access Control Address*. MAC-osoite on tietokoneen tietoverkkoon liittävän verkkosovittimen yksilöitävä osoite.

Mbit/s: Tiedonsiirtonopeus megabittiä sekunnissa (1 000 000 bit/s).

SCI: *Spinal Cord Injury.* Selkäydinvamma.

1 Johdanto

Painehaavat aiheuttavat vuosittain suurta haittaa monella eri tavalla. Ne voivat olla pitkälle edenneinä todella kivuliaita potilaille, jotka usein jo sairastavat muitakin sairauksia. Ne aiheuttavat suuria lisäkustannuksia terveydenhoidolle ympäri maailmaa. Niiden ennaltaehkäisy vaatii jatkuvaa valppautta hoitohenkilökunnalta, jolla ei välttämättä ole nykymaailmassa aikaa tehdä kaikkia varotoimenpiteitä.

Suurin osa olemassa olevista teknologisista ratkaisuista painehaavoihin on suunniteltu makuupotilaiden käyttöön. Tästä syystä lähdettiin kehittämään ennaltaehkäisevää tuotetta, joka olisi nimenomaan suunnattu pyörätuolia käyttäville potilaille.

Insinöörityö tehtiin osana *Research-to-Business* projektia Aalto-yliopistossa, jonka tavoitteena on kehittää keinoja painehaavojen ennaltaehkäisyyn. Tähän projektiin liittyen kehitetään mobiilisovellus, joka toimii käyttöliittymänä mobiililaitteen ja painetta mittaavan istuinmaton välillä. Sovelluksen tavoitteena on yhdistyä *Bluetoothin* kautta painemattoon, josta käyttäjän painearvot kerätään. Nämä arvot esitetään sovelluksessa lämpökartan muodossa ja niiden pohjalta tehdään laskelmia sovellukseen lisättyjen algoritmien avulla.

Tässä työssä sovelluskehitystä katsotaan toimintatutkimuksen viitekehyksessä. Toimintatutkimusta käytetään yleensä sosiaalialoilla toimintatapojen parantamiseen, mutta tässä työssä sitä on sovellettu teknologisesta näkökulmasta. Sovelluskehitys koostuu kahden viikon mittaisista sprinteistä, joiden päätteeksi kaikki tehdyt asiat käydään läpi. Sitten katsotaan, mitä tehdään seuraavaksi, ja pohditaan myös, ovatko jotkin muutokset sittenkään hyödyllisiä tai toimivia nykyisessä muodossaan. Tämä sprintin lopussa oleva sovellus on aina yksi iteraatio, jonka pohjalta sitten katsotaan, miten sovellusta voidaan muuttaa seuraavien kahden viikon aikana.

2 Tausta

Tässä luvussa taustoitetaan projektin aihetta. Aluksi kerrotaan painehaavoista ja niiden syntyyn vaikuttavista tekijöistä. Sen jälkeen avataan niitä keinoja, joilla painehaavoja yleensä pyritään ehkäisemään. Lopuksi kerrotaan hieman mobiilisovelluksen konseptista.

2.1 Painehaavat

Painehaavat aiheuttavat suurta haittaa niin potilaille kuin terveydenhuollolle ympäri maailman. Ne ovat monissa tapauksissa vältettävissä. Isossa-Britanniassa tutkittiin painehaavojen kustannuksia terveydenhuollolle. Tutkimuksen arvion mukaan varhaisen vaiheen painehaavan hoito maksaa noin 1150 €, kun taas pitkälle kehittynyt haava maksaa jopa 11500 €. Vuosittaisen kokonaislaskun tutkijat arvioivat Isossa-Britanniassa olevan 1,6–2,4 miljardia euroa. Tästä laskusta yli 90 % aiheutuu hoitajien ylimääräisistä töistä. Tutkijat mainitsevat myös, että tämän laskun odotetaan lisääntyvän tulevaisuudessa väestön vanhenemisen seurauksena. Tavoitteena tulevaisuudessa tulisi heidän mukaansa olla keskittyminen haavojen ehkäisyyn. Tarkemmin sanoen tähän kuuluisi akuutin paineen, syntyneiden haavojen etenemisen sekä infektioiden ehkäisy. (1.)

Painehaavojen syntyyn vaikuttavia tekijöitä ovat kudoksiin kohdistuva paine, kitka kehon kerrosten välillä, venytys sekä mikroilmasto. Lisäksi pinnallisten ja syvien painehaavojen kehittymisellä on eroja. Paine syntyy siitä, kun ulkoinen voima kohdistuu kohtisuoraan ihon pintaan. Painetta esiintyy sekä iholla, että ihon alaisissa kudoksissa. Paine puristaa kudoksia ja voi aiheuttaa ihon ja kudoksen vääntymistä ja muodonmuutosta. Pehmytkudos vääntyy ja muuttaa muotoaan enemmän, jos paine kohdistuu luun kohdalle. (2.)

Kitka on voima, joka syntyy, kun kaksi toisiaan koskettavaa kohdetta liikkuvat toistensa suhteen. Kitkaa esiintyy esimerkiksi ihon ja kontaktipinnan välissä, kun painovoima pyrkii vetämään potilasta pois sängyltä tai tuolilta. Kitka vaatii jonkinlaisen paineen ilmetäkseen. (2.)

Venytys voi johtua tangentiaalisesta voimasta, eli voimasta, joka on saman suuntainen ihon pinnan kanssa. Kun tangentiaalisen voiman alla esiintyy paljon kitkaa ihon ja kontaktipinnan välissä, iho pysyy kontaktipintaa vasten, mutta ihonalaiskudoksen kerrokset muuttavat muotoaan liikkuessaan potilaan mukana. Venytystä voi tapahtua myös syvemmissä kudoskerroksissa tai niiden välissä luisiin ulokkeisiin kohdistuvan paineen aiheuttaman kudosten muodonmuutoksen vaikutuksesta. Lihakset ovat erityisen alttiita venytyksen aiheuttamille vaurioille. (2.)

Mikroilmasto tarkoittaa lämmön ja kosteuden muodostamia olosuhteita ihon ja kontaktipinnan välissä. Ihon lämpötilan nousun on havaittu lisäävän painehaavojen kehittymisen riskiä. Ihon ja kontaktipinnan välinen korkea kosteuspitoisuus voi edesauttaa painehaavojen kehittymistä heikentämällä ihoa ja lisäämällä ihon ja kontaktipinnan välistä kitkaa sekä venytystä, mikä lisää kudosvaurion todennäköisyyttä. (2.)

Pinnalliset painehaavat, eli kategoria I ja II, ja syvät painehaavat, eli kategoria III ja IV, sekä syvien kudosten vaurio, voivat kehittyä eri mekanismeilla. Ihoon kohdistuva kitka ja leikkaavat voimat sekä jokin pinnallinen ihovaurio, kuten esimerkiksi ärsytysihottuma, ovat merkittäviä pinnallisia painehaavoja edistäviä tekijöitä. Pinnallinen haava syntyy, kun nämä voimat muuttavat ihon fyysisiä ominaisuuksia, lisäävät venytystä ja painetta syvemmissä kerroksissa ja johtavat näiden seurauksena vaurion etenemiseen syvemmälle. (2.)

Syvien painehaavojen ja syväkudosvaurioiden ajatellaan syntyvän pääasiassa paineen ja venytyksen aiheuttamasta syvien kudosten muodonmuutoksesta. Vahinko tapahtuu aluksi lihas-/luurajapinnassa, ja ihon rikkoutumista esiintyy myöhemmässä vaiheessa, kun vaurio lähtee ulottumaan kohti ihon pintaa. (2.)

2.2 Olemassa olevat ratkaisut painehaavojen ennaltaehkäisyyn

Tärkeä osa painehaavojen ehkäisyä on kudoksiin kohdistuvan paineen vähentäminen. Monet tukea antavat alustat on suunniteltu tähän tarkoitukseen. Näitä

voivat olla esimerkiksi sängyillä käytettävät erikoispatjat ja vaahdosta muodostetut tyynyt. Ne toimivat muutamalla eri tavalla. Näitä voivat olla esimerkiksi paineen levittäminen mahdollisimman suurelle kehon alueelle, mekaanisesti paineen muuttaminen tietyissä sykleissä, jotta kehon eri alueet eivät ole jatkuvasti samassa paineessa, sekä edellä mainittujen yhdistelmä, joka mahdollistaa sen, että terveydenhuollon työntekijät voivat muokata käytettävää metodia potilaan tarpeiden mukaan. (3.)

Teknisellä puolella on tarjolla paljon ratkaisuja kehoon kohdistuvan paineen seurantaan. Näistä tunnetuimmat on yleensä suunniteltu makuupotilaille, mutta sovelluksia löytyy myös istuville käyttäjille. Tutkimuksissa on huomattu, että suurin osa painehaavan ehkäisyyn käytettävien sovellusten kanssa yhtä aikaa tehdyistä tutkimuksista keskittyvät sovelluskehityksen alkuvaiheisiin. Tämän seurauksena olemassa olevat sovellukset vaativat tutkimuksia myös myöhemmästä vaiheesta, kun niitä kenties jo käytetään potilaiden kanssa. Potilaiden osallistuminen sovellusten kehitystyöhön olisi tärkeää. Kehitettävien sovellusten tulisi myös arvioida niiden toiminnallisuutta vaihtelevissa käyttöympäristöissä ja tilanteissa. (4.)

Varsinaisia teknisiä kehityskohteita ovat muun muassa valokuvien avulla tehtävän painehaavojen tunnistuksen sekä algoritmien parantaminen, tieteellisten tutkimusten hyödyntäminen kehitystyössä, käyttäjädatan ja eettisten huolien käsittely sekä kuvatietokantojen laajuuden parantaminen. (4.)

2.3 Kehitettävän sovelluksen konsepti

Sovelluksen haluttiin sisältävän tiettyjä toiminnallisuuksia ja elementtejä, kun taas niiden varsinainen toteutus on vapaamuotoista. Sovelluksen algoritmit olivat tärkein osa kehitystyötä. Näitä algoritmeja ovat asennontunnistus, paineen seuranta pidemmältä aikaväliltä, painekeskipisteen mittaus sekä korkeimpien painepisteiden tunnistus. Algoritmit toimivat sovelluksen taustalla, kun taas käyttäjällä oli näkyvissä painematon lähettämästä datasta muodostettu

lämpökartta. Algoritmit käyttävät tätä samaa dataa laskutehtävissään. Lämpökartta on tärkein käyttöliittymän elementti sovelluksessa.

Algoritmeista tärkein on paineen kumulatiivinen seuranta. Sen perusteella halutaan lähettää käyttäjälle ilmoituksia, mikäli paine nousee jossain osassa mattoa liian korkeaksi, jolloin käyttäjä toteuttaa helpotusliikkeen. Nämä liikkeet ovat yksinkertaisia, esimerkiksi toisen pakaran kohotus ilmaan pois istuimelta.

Tämä konsepti on realistisesti mahdollista toteuttaa ainoastaan *Bluetooth Low Energyn* avulla. BLE on uudempi versio perinteisestä *Bluetoothista*. Se on tässä tapauksessa hyödyllisempi vanhaan *Bluetoothiin* verrattuna, koska dataa ei ole tarpeellista siirtää pitkiä matkoja, tai suurta määrää kerrallaan. Painematossa on USB-liitin, mutta se ei ole ideaalinen kaikkiin käyttötilanteisiin. BLE puolestaan sopii langattomuutensa ansiosta hyvin tähän käyttötarkoitukseen. BLE:tä käytetään siis yhteyden muodostukseen painematon ja mobiililaitteen välille.

3 Suunnittelu ja menetelmät

Tässä insinöörityössä oli tärkeää suunnitella etukäteen, mitkä asiat sovelluksen kehityksestä tullaan käsittelemään. Sovelluksen kehitys jatkui pidemmälle kuin mitä tässä työssä käsitellään. Lisäksi insinöörityön tekemiseen käytettiin kahta menetelmää, joista kerrotaan tässä luvussa.

3.1 Työn suunnittelu ja aiheen rajaus

Tässä työssä käsitellään enimmäkseen sovelluksen kehitystä prosessina, siinä apuna käytettyjä teknologioita, sekä pohditaan sitä, mitä olisi voitu tehdä paremmin ja tuliko sovelluksesta halutunlainen. Toimintatutkimus luo rungon sprinttien läpikäyntiin. Sprinteissä kerrotaan aluksi, mitä on milläkin aikavälillä tehty. Sen jälkeen katsotaan, mitä havaintoja sprintin aikana tehtiin. Sprinttien edistymistä havainnoidaan vähintään siten, että seurataan, kuinka paljon sprinttiin valituista tehtävistä saatiin valmiiksi sen aikana. Havainnoinnin jälkeen reflektoidaan mennyttä sprinttiä. Tässä saatetaan katsoa, miten toimintaa muutettiin sprintin

aikana, miten muutokset toimivat tai eivät toimineet, sekä verrata myös, miten muutokset vaikuttivat sovelluksen edistymiseen verrattuna menneisiin sprintteihin.

Sovelluksen kehityksen osalta tämä työ sisältää vain osan varsinaisesta tehdystä työstä. Kaikki kehitystyö, joka tapahtui vuoden 2023 helmikuun ja heinäkuun välillä, sisällytetään mukaan. Sen jälkeen tapahtuneita muutoksia tai lisäyksiä ei huomioida. Taulukossa 1 on esitetty ennen aikarajaa tehdyt lisäykset ja muutokset sekä ne asiat joita ei ehditty tehdä ennen aikarajaa.

Taulukko 1. Tässä työssä käsiteltävät sekä rajauksen ulkopuolelle jääneet muutokset.

Sovelluksen osa-alue	**Käsitelty tässä työssä**	**Ei käsitelty**
Käyttöliittymä	Lämpökarttasivu, ohjeet mittauksen suorittamiseen, yksinkertaiset hälytykset käyttäjälle	Tilastosivu, etusivu, profiilisivu, kirjautuminen, tilin luominen, hälytysten ja paineen esittäminen tietyillä aikaväleillä graafisessa muodossa, mahdollisuus valokuvan ottoon painehaavakohdasta.
Algoritmit	Painekeskipiste, korkeat painepisteet, kumulatiivinen paine, asennontunnistus	Kumulatiivinen paineseuranta korkean paineen pisteille, parempi korkeiden pisteiden tunnistus
Yhteydet	BLE-toiminnallisuus	Sovelluksen sisäinen kommunikaatio sivujen välillä

Tiedon tallennus	Tietokanta mittauksille, sekä eri asennoille	Tietokanta käyttäjätiedoille, sekä hälytyksille

Käytettyjä teknologioita käsitellään vaihtelevalla laajuudella. Esimerkiksi BLE on sovelluksen toiminnan kannalta kriittinen teknologia, joten se selitetään huomattavasti syvällisemmin verrattuna esimerkiksi Figmaan, jota ei käyttöliittymätyökaluna päästy kesäkuun loppuun mennessä vielä kunnolla käyttämään. Nämä erilliset teknologiat ovat siis omassa luvussaan, mutta myös kehitystä käsittelevässä luvussa käydään läpi joitain teknologioita. Näitä ovat esimerkiksi erilaiset API:t, joita sovelluksessa päätettiin käyttää. API tarkoittaa ohjelmointirajapintaa, jonka avulla eri ohjelmistot voivat kommunikoida keskenään. Ne selitetään tarkemmin läpi sitä mukaan, kun ne sovelluksen kehityskaaressa tulivat vastaan.

Varsinaisen kehitysprosessin osalta tässä työssä keskitytään tekniseen puoleen, eli mitä piti tehdä, kuinka hyvin tehdyt asiat saatiin toteutettua, mitä muutoksia toteutettuihin ominaisuuksiin tarvittiin seuraavissa sprinteissä ja kuinka hyvin kunkin sprintin tavoitteisiin päästiin. Pois jätetään työntekijän omat huomiot siitä, kuinka vaikeita asiat olivat tai mitä uutta opittiin työtä tehdessä.

3.2 Toimintatutkimus

Toimintatutkimuksella yleensä tarkoitetaan jonkin ongelman selvittämistä tai ratkaisemista tavalla, joka toiminnallistaa tutkimukseen osallistujia. Tällä tarkoitetaan sitä, että esimerkiksi työpaikalla jotain ongelmaa tutkittaessa työntekijät, jotka kohtaavat kyseisen ongelman, ovat tiiviisti mukana myös ratkaisemassa sitä. Toisella tavalla katsottuna sen voisi kiteyttää niin, että sen ytimessä on tiedon tuottaminen ja toimintojen kehittäminen aidoissa ympäristöissä. Toimintatutkimuksissa käytetään yleensä ongelmanratkaisun eri keinoja. (5.)

Yleisesti toimintatutkimuksen hyötyjen katsotaan liittyvän tähän osallistujien aktivointiin. Pelkästään työn lomassa tapahtuva ajattelu ei välttämättä riitä syvällisten ongelmien ratkaisuun. Se ei myöskään takaa, että ongelmaa ratkaiseva taho ajattelee asiaa laaja-alaisesti. Toimintatutkimuksesta on hyötyä siinä mielessä, että se auttaa osallistujia teoretisoimaan toimintatapojaan sekä tarkastelemaan toimintaansa kriittisesti. (5.)

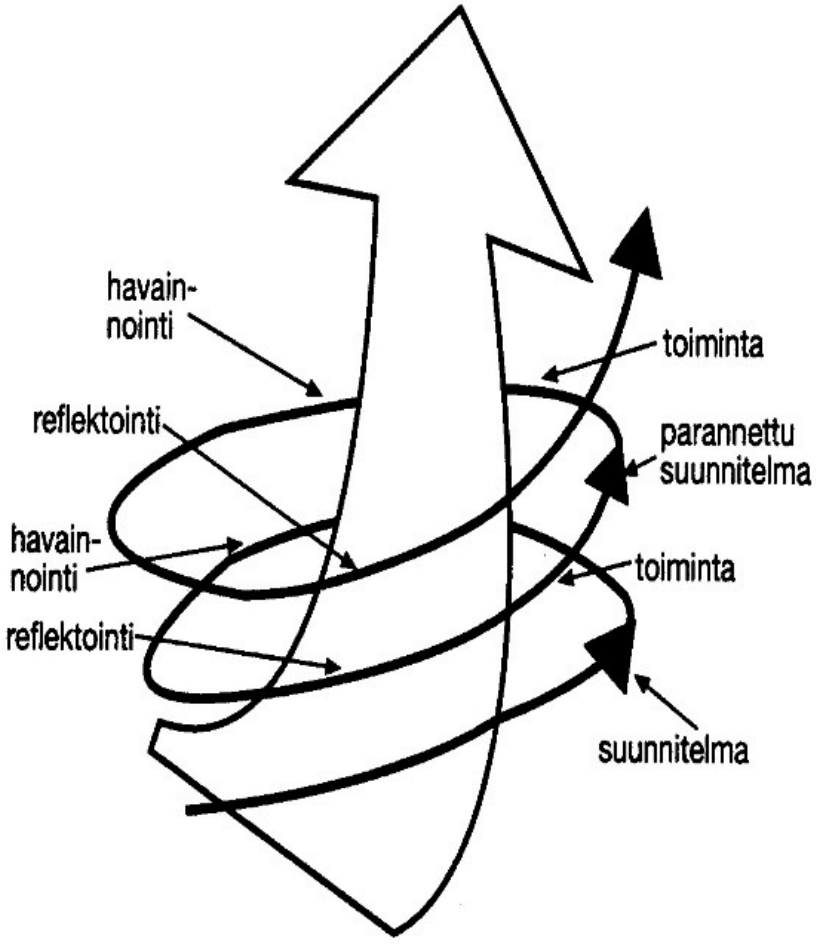
Toimintatutkimusta voidaan verrata perinteisempään tutkimusmalliin, jotta sen erot ja hyödyt tulevat selviksi. Suurin ero näiden kahden välillä on, että toimintatutkimuksessa keskitytään yleensä käytännön ongelmiin ja tutkittavat henkilöt ovat tiiviisti mukana koko tutkimusprosessissa. Perinteisemmässä tutkimuksessa keskitytään yleensä kirjallisuudesta löydettyihin ongelmiin ja niitä yritetään ratkaista tutkittavilta kerättävän tiedon avulla. Taulukko 2 kuvaa joitain eroja näiden metodien välillä. (5.)

Taulukko 2. Toimintatutkimus verrattuna perinteisempään tutkimukseen.

Arvioinnin kohde	**Toimintatutkimus**	**Perinteinen tutkimus**
Ongelma	Käytännön ongelma (suoranaisesti tai epäsuorasti tutkimukseen osallistujien määrittelemä)	Kirjallisuuteen perustuva ongelma (löydetty perehtymällä kirjallisuuteen)
Tavoite	Kohdistuu käytännön ongelmaan, tavoitteena ammattikäytännön parantaminen	Täyttää kirjallisuudesta esille noussutta aukkoa ja lisää tietämystä tutkimuskohteesta
Osallistuminen	Tutkittava aktiivisia toimijoita tutkimuksen alusta loppuun saakka	Tutkittavat osallistuvat tiedon tuottamiseen

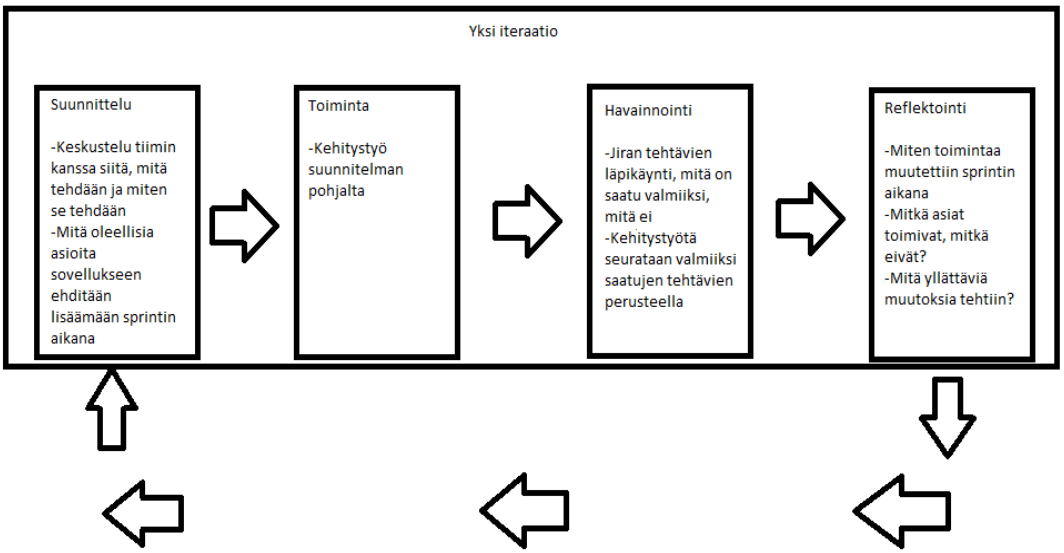
Tämän insinöörityön osalta toimintatutkimusta sovelletaan hieman. Toimintatutkimusta käytetään useimmiten sosiaalialoilla, sillä sen tavoitteena on yleensä muuttaa toimintatapaa jossain yhteisössä. Tämä on suurelta osin ihmiskeskeistä. Teknologia saattaa olla mukana osana prosessia, tai kenties ratkaisu ongelmaan on jokin teknologiaväline, mutta toimintatutkimusta harvoin käytetään itsessään jonkin teknologian kehittämiseen. Tässä työssä on kuitenkin päätetty, että toimintatutkimuksen yleisiä toimintatapoja käytetään mobiilisovelluksen kehittämisessä.

Toimintatutkimukset yleensä tehdään kolmen eri mallin mukaan. Kuvassa 1 on esitetty eräs näistä, eli spiraalimalli.



Kuva 1. Toimintatutkimuksen spiraalimalli. (6)

Spiraalimallinen tutkimus on ns. tekninen toimintatutkimus, joka koostuu mahdollisesti toistuvista tutkimusvaiheista. Nämä voivat esimerkiksi olla suunnittelu -> toiminta -> havainnointi -> reflektointi. (7.) Tämä insinöörityö käyttää spiraalimallia pohjana sprinteissä. Insinöörityön sovellus on kehitetty suurelta osin samalla tavalla, kuin miten tekninen toimintatutkimus tehdään. Se kehitettiin sprinteissä, joissa on myös vaiheina esimerkiksi suunnittelu, toteutus ja reflektointi, jonka jälkeen palataan taas suunnitteluvaiheeseen, josta kierros käynnistyy uudelleen. Sprinteistä kerrotaan myöhemmin lisää. Kuvassa 2 on näkyvillä prosessikaavio siitä, miten toimintatutkimusta on käytetty tässä työssä.



Kuva 2. Prosessikaavio sovelluksen iteraatioista.

Toimintatutkimusta käytetään tässä työssä seuraavalla tavalla. Ongelma, jota tutkitaan, on pyörätuolia käyttävien henkilöiden painehaavojen ehkäisy. Ratkaisuja tähän on jo olemassa, mutta yksikään niistä ei ole täydellinen, ja tässä työssä on aiemmin tarkasteltu hieman sitä, mitä ongelmia niissä on. Näiden tietojen pohjalta verrataan, miten tässä työssä läpikäytävä sovellus vastaa näihin ongelmiin. Näitä eroja on jo sovelluksen kehitysvaiheessa käytetty pohjana sille, mitä ominaisuuksia sovelluksessa täytyy olla. Spiraalimuotoinen toimintatutkimus tulee esiin siinä vaiheessa, kun myöhemmin työssä käydään läpi sprinttejä ja niissä tehtyjä asioita. Tässä tekstissä sprintit jakautuvat neljään osaan. Aluksi

katsotaan, mitä tehtäviä suunnitelman perusteella sprinttiin halutaan sisällyttää. Sen jälkeen katsotaan, mitä sprintin aikana tehtiin. Sen jälkeen havainnoidaan tehtyä työtä, seuraten lähinnä sitä, miten hyvin asetettuihin tavoitteisiin päästiin. Tärkein seurantametodi ovat suunnitelmassa päätetyt tehtävät. Lopuksi reflektoidaan, eli katsotaan mitä opittiin, mitkä asiat toimivat, mitkä eivät sekä miten kehitystyötä muutettiin havaintojen perusteella.

3.3 Ketterä kehitys

Ketterä kehitys on ohjelmistokehityksen menetelmä, joka perustuu nopeuteen, joustavuuteen ja iteraatioihin. Se eroaa filosofiana toisesta ohjelmistokehityksen menetelmästä, eli vesiputousmallista, merkittävästi. Tämä vanha malli perustuu paljolti ennakoivaan suunnitteluun, josta edetään testaukseen, tuotantoon ja ylläpitoon. Tämän mallin heikkous on se, että tehty ennakkosuunnitelma ei todennäköisesti pysy sopivana projektin tai tuotteen kehityksen edetessä. Yleensä kehitettävät ohjelmistot selkeytyvät kokonaisuutena vasta kehitystyön edetessä, johon tämä vesiputousmalli ei välttämättä pysty vastaamaan. (8.)

Ketterässä ohjelmistokehityksessä tähän pyritään vastaamaan luomalla ainoastaan laaja kuva ennakoidusti ja tämän kuvan sisällä yksityiskohdat selvitetään prosessin edetessä tarkemmin. Tässä siis silti suunnitellaan ja määritellään etukäteen raameja, mutta kaikkia yksityiskohtia ei päätetä heti ennen varsinaisen työn alkua. (8.)

Ketterässä kehityksessä on omat haasteensa sen positiivisten ominaisuuksien lisäksi. Se vaatii sitoutumista sen jatkuvaluonteisuuden takia. Sitä ei kannata testata vain kuukauden ajaksi, koska sen kaikkia hyötyjä ei näin saada tuotua esiin. Toinen haaste on sen soveltaminen suurempiin projekteihin. Suuremmat projektit on yleensä jaettu pienempiin, erillisiin osiin, joiden keskinäinen kommunikaatio voi vaihdella laadultaan. Ketterä kehitys perustuu osittain läheiseen ja jatkuvaan kommunikointiin, joten tämä voi aiheuttaa ongelmia. Ketterä kehittäminen ei lisäksi välttämättä sovellu tiukasti säädeltyihin toimialoihin. Yksi näistä aloista on terveydenhuolto. Näiden säädösten takia muutokset ovat usein hitaita

ja vaativat laajempaa validointia verrattuna muihin aloihin. Ketterässä kehityksessä muutokset tehdään yleensä lyhyillä aikaväleillä, jonka takia se ei välttämättä sovellu näille aloille. (8.)

Ketterässä kehityksessä käytetään siis lyhyitä kehitysjaksoja eli sprinttejä. Niiden kesto vaihtelee, mutta useimmiten ne kestävät 1–4 viikkoa. Sprinteissä pyritään yleensä kehittämään joku ohjelmiston osa valmiiksi ja julkaistavaksi. Sprinttejä katsotaan usein jatkuvan kehittämisen näkökulmasta. Tähän kuuluu, että sprinttien jälkeen käydään läpi, mitä on tehty, miten se tehtiin ja mitä olisi voitu tehdä paremmin, jotta seuraavassa sprintissä nämä ideat voidaan ottaa käyttöön. Kehitystiimi on siis jatkuvasti kiinnostunut omasta toiminnastaan ja sen parantamisesta. (8.)

Sprinteissä on yleisesti 3 roolia. Näitä ovat kehittäjä, *Scrum Master*, sekä tuoteomistaja. Kehittäjä on itseselitteinen, eli esimerkiksi sovelluksen kehittäjä. *Scrum Master* auttaa tuoteomistajaa tuottamaan hyödyllistä tekemistä sprinttejä varten, ja muun muassa motivoi kehittäjiä. Tuoteomistajan tehtävä on ymmärtää asiakkaiden tarpeita ja vaatimuksia sekä pitää huolta tulevien sprinttien tehtävistä, jotka vastaavat näihin tarpeisiin. (9.)

Sprintit otettiin käyttöön sovelluksen kehityksessä siten, että yksi projektin esimiehistä toimi *Scrum Masterin* roolissa, jonka lisäksi kaksi muuta esimiestä toimivat tuoteomistajina, vaikka *Scrum Master* teki myös tuoteomistajan tehtäviä projektin aikana. Varsinaisia kehittäjiä oli yksi. Virallisia päivittäispalavereja, jotka yleensä kuuluvat sprinttityöskentelyyn, ei järjestetty. Niiden arvo korvattiin sillä, että projektiryhmä työskenteli samassa tilassa.

Sprinttejä ei heti aluksi ollut, vaan kehitys oli pääosin vapaamuotoista. Tosin kehitystyössä oli jo löyhät tavoitteet sille, mitä haluttiin saavuttaa ennen sprinttien käyttöönottoa. Niitä alettiin käyttämään ensimmäisen kuukauden jälkeen. Sprinttien pituudeksi valittiin 2 viikkoa, joiden jälkeen sprintin arviointi järjestettiin keskiviikkoisin. Keskimäärin sprintit koostuivat 3–4 tehtävästä, jotka olivat työmäärältään pieniä tai keskikokoisia palasia.

4 Käytetyt teknologiat

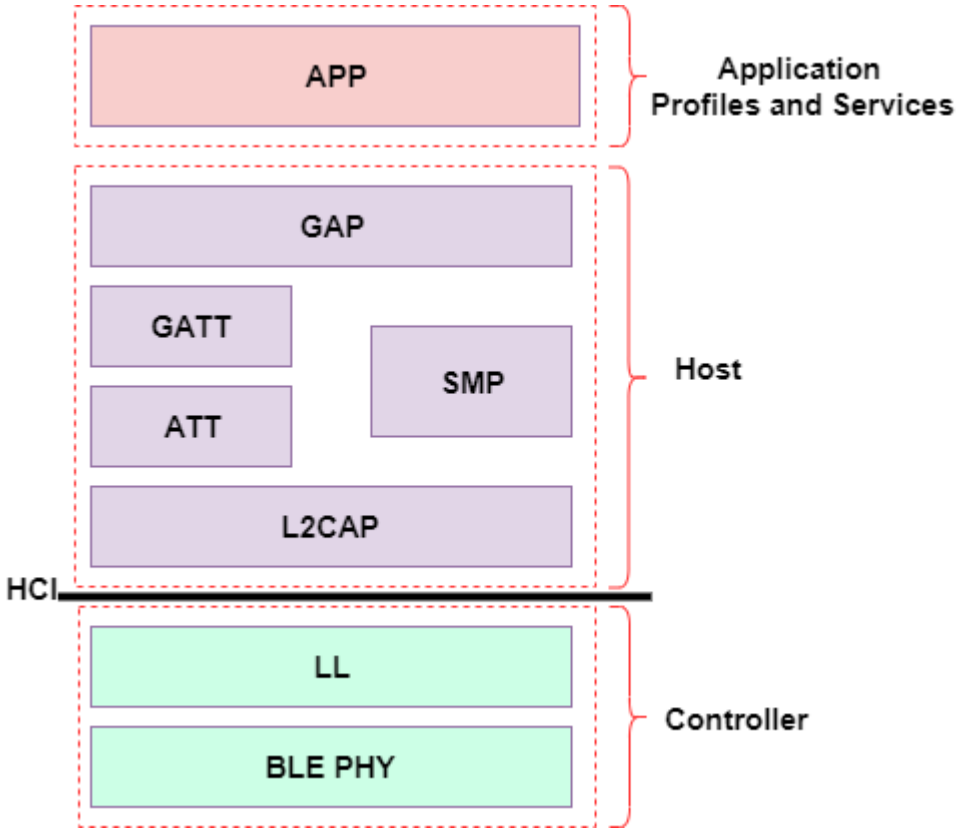
Tässä luvussa käsitellään sovelluksen kehityksessä käytettyjä teknologioita. Se, miten laajasti kutakin teknologiaa on käsitelty, riippuu siitä, miten laajasti ne ovat olleet käytössä tai miten tärkeitä ne ovat sovelluksen kehityksessä ja toiminnallisuudessa. Tässä luvussa ei käsitellä sovelluksessa käytettyjä API:ta tai muuta sovelluksen varsinaiseen lähdekoodiin liittyvää. Lähdekoodiin liittyen kerrotaan kuitenkin jonkin verran Javasta sekä syistä sille, miksi sitä on käytetty tämän sovelluksen koodikielenä. BLE:stä kerrotaan varsin syvällisesti, muista teknologioista ei yhtä pitkästi.

4.1 Bluetooth Low Energy

BLE oli alun perin osa *Bluetooth* 4.0 spesifikaatiota. Se eroaa monilta osin alkuperäisestä *Bluetoothista* suunnittelufilosofian sekä käyttötarkoituksen, mutta myös muiden osa-alueiden takia. BLE:n filosofia oli sen kehittäjien mukaan alusta lähtien seuraavanlainen: mahdollisimman matala virrankulutus, matala hinta, matala kaistanleveys sekä helppokäyttöisyys. (10.)

Kirjan *Getting started with Bluetooth Low Energy* kirjoittajat sanovat, että BLE on otettu laajamittaiseen käyttöön yllättävän nopeasti. Se tuli alun perin käytettäväksi vuonna 2010, ja se on huomattavasti muita langattomia samaan aikaan käyttöön tulleita teknologioita edellä. Se otettiin laajasti käyttöön Applen ja Samsungin toimesta juuri samaan aikaan, kun mobiililaitteet nousivat nopeasti suosiossa. (8.) Osaltaan tämä tekee siitä järkevän teknologian langattomaan yhteydenmuodostukseen mobiilisovelluksen sekä ulkoisen laitteen välillä.

Tärkeä osa *Bluetoothin* toimintaa on pino (*stack*). Kuvassa 3 on esitetty Androidin BLE-pino.



Kuva 3. Androidin BLE-pinon rakenne.

Pinon tarkoitus on tarjota kehittäjille työkaluja, teknologioita sekä muita kehityksessä käytettäviä elementtejä työn helpottamiseksi. *Bluetoothilla* on monia eri protokollia, joiden käyttö olisi vaikeampaa ilman kehittäjille tehtyjä pinoja. Androidin BLE-pinoon kuuluvat muun muassa sovelluskerros, jonka kautta skannaus, yhteyden muodostus ja muut vastaavat toiminnot tehdään. Seuraava kerros koostuu profiileista, kuten GATTista ja GAPista. Näiden tehtävä on muun muassa määrittää, missä muodossa BLE:n välityksellä kulkeva data on. HCI-kerros vastaa kommunikaatiosta Bluetooth-radion ja laiteajureiden kanssa. (10.)

Apple on käyttänyt paljon aikaa oman BLE-pinonsa kehitykseen sekä käyttöohjeiden suunnitteluun. (11.) Joidenkin sovelluskehittäjien mielestä Android ei sen sijaan ole yhtä pitkällä oman pinonsa osalta. iOS on edellä muutamilla tärkeillä osa-alueilla. Esimerkiksi eri operaatioiden jonotus tapahtuu Applen pinolla automaattisesti, kun taas Androidilla kehittäjät joutuvat toteuttamaan jonon manuaalisesti. Käytännössä operaatioiden jonotus tarkoittaa sitä, että ne tapahtuvat

yksi toisensa jälkeen. Androidilla ilman manuaalista jonotusta eri BLE-operaatiot saattavat tapahtua samanaikaisesti, mikä voi johtaa sovelluksen virheelliseen toimintaan. Muitakin eroja näillä eri valmistajilla on, joista suurin osa liittyy Applen API-paketteihin. (12.) Nämä tekijät vaikuttavat BLE:tä käyttävän sovelluksen kehitykseen Androidilla.

BLE on hyödyllinen alhaisten käyttövaatimustensa ja helpon langattoman yhdistämisen lisäksi myös tiedonsiirron osalta. BLE:n alhaiset vaatimukset muodostuvat muun muassa alhaisemmasta virrankulutuksesta, matalammasta tiedonsiirtonopeudesta sekä tehokkaammasta kommunikaatiosta laitteiden välillä. (11.) Tämän insinöörityön sovellus vaatii tiedonsiirron käytetyn ulkoisen laitteen ja mobiilisovelluksen välillä langattoman yhteyden lisäksi. BLE on näistä syistä sopiva teknologia kehitetylle sovellukselle.

BLE:n kantama on lyhyempi verrattuna normaaliin *Bluetoothiin*. (13.) Tämä ei ole merkittävää kehitetyssä sovelluksessa, koska 50 metrin käyttöetäisyys riittää lähes jokaiseen käyttötilanteeseen. Tämä etäisyys on kuitenkin vain ilmoitettu maksimi, joka vaihtelee esimerkiksi erilaisten ympäristön esteiden takia. Realistinen tehokkaan datan siirron mahdollistava ja varmasti toimiva yhteys on alle 10 metriä. (11) Useimmiten sovellusta käyttävä kännykkä on painematon lähellä, esimerkiksi käyttäjän taskussa. Myös 1 Mbit/s maksimitiedonsiirto on riittävä sovelluksen hyödylliseen toimintaan.

BLE:tä käyttävä laite kykenee kommunikoimaan muiden laitteiden kanssa kahdella tavalla: lähettäminen (*broadcasting*) ja yhteyden muodostus (*connection*). Lähetys on pienten tietomäärien siirtoa ilman varsinaisen yhteyden muodostusta yhdestä laitteesta toiseen. Lähetyksessä siirtyvä data on usein jonkinlainen tunniste, jolla vastaanottava laite kykenee erottamaan sen muista läheisistä BLE-laitteista. Tällä mekanismilla tiedon lähetys on yksisuuntaista. Rooleja on kaksi: lähettäjä sekä tarkkailija. Lähettäjä nimensä mukaisesti lähettää ns. mainospaketteja kaikille tarkkaileville laitteille. Tarkkailija sen sijaan kuuntelee ja vastaanottaa näitä paketteja. Tämä mekanismi on hyödyllinen pieneen tiedonsiirtoon. (11.)

BLE:tä käyttävä sovellus (*central*) muodostaa yhteyden ulkoisen laitteen (*peripheral*) kanssa kuuntelemalla ulkoisten laitteiden lähettämiä ns. mainospaketteja. Näitä mainospaketteja on mahdollista rajata erilaisten suodattimien avulla. Kun sovellus poimii sopivan paketin, se voi lähettää pyynnön ulkoiselle laitteelle yhteyden muodostamiseen. Tämä yhteys on eksklusiivinen, joka tässä tapauksessa tarkoittaa suojattua tiedonsiirtoa laitteiden välillä. Sovellus on tässä tapauksessa se osapuoli, joka ilmoittaa toiselle laitteelle sen, milloin tiedonsiirto tapahtuu, ja myös aloittaa sen. (11.) Kehitetyn sovelluksen osalta roolijako on selkeä: sovellus on hallinnoivassa roolissa, ja matto toteuttaa tiedonsiirron sovelluksen ohjeiden mukaisesti.

BLE:n yhteyksien suurin etu on *Generic Attribute Profile* (GATT). Tieto on organisoitu palveluihin (*services*) sekä ominaisuuksiin (*characteristics*). Avaintekijä GATT-profiilissa on se, että palveluilla voi olla monia eri ominaisuuksia, ja näillä ominaisuuksilla on esimerkiksi omat pääsynrajoituksensa. (11.) Kehitetyn sovelluksen kannalta tämä on hyödyllistä, koska eri palveluilla on omat ID-arvonsa, mukaan lukien painematoilla. Näin on mahdollista erotella sellaiset BLE-laitteet, jotka mainostavat tätä haluttua palvelua muista laitteista. Käytännössä tämä varmistaa sen, että sovellus löytää ja pystyy muodostamaan yhteyden ainoastaan haluttuihin painemattoihin.

4.2 Android Studio

Android Studiota käytetään tämän projektin mobiilisovelluksen kehitykseen. Se on Androidin virallinen sovelluskehitysympäristö tai *Integrated Development Environment* (IDE). Se käyttää pohjana IntelliJ IDEA-koodinkirjoitusalustaa. (14.) Tätä päätettiin käyttää tässä projektissa, koska kehittäjällä oli jo aiempaa kokemusta sen käytöstä.

Android Studiolla on monia hyödyllisiä ominaisuuksia Android-sovellusta kehitettäessä. Muutamia näistä ovat:

- *Gradle*-pohjainen rakennusjärjestelmä

- suhteellisen nopea emulaattori
- testausjärjestelmät, kuten *debugger*
- mahdollisuus testata sovellusta fyysisellä laitteella USB:n kautta
- mahdollisuus selata helposti Android-sovelluksen normaalisti piilotettuja kirjastoja
- *Lint*-työkalut, joiden avulla on helpompaa tunnistaa virheet ja varoitukset koodissa
- tuki Java- ja Kotlin-koodikieliin.

Gradle on järjestelmä tietokonekoodin automaattiseen rakentamiseen. Tietokonekoodin muuntaminen toimivaksi sovellukseksi vaatii koodin pakkaamisen ja muuntamisen lähdekoodista binäärikoodiksi. *Gradle* tekee tämän nopeasti ja tehokkaasti käyttämällä pohjana virtuaalikonetta. (15.)

4.3 Versiohallinta

Versiohallintaan käytetään järjestelmää nimeltä Git. Se on yksi suosituimmista versiohallintajärjestelmistä, joita käytetään muutoksien seuraamiseen koodissa sekä samanaikaisten muutosten hallinnointiin, mikäli kehittäjiä on enemmän. Sen avulla on mahdollista palata edelliseen lähdekoodiversioon, mikäli jokin toiminto koodin uudessa versiossa rikkoutui. Git:llä on muutamia tässä projektissa tärkeitä toimintoja, jotka auttavat muun muassa päivitettävien tiedostojen valinnassa, sekä muutosten tallennuksessa. Git itsessään käyttää terminaalia komentojen käyttöön, joten versiohallinnan käytön helpottamiseksi päätettiin käyttää *Sourcetree*-sovellusta. (16.)

Sourcetree on helpompi käyttöliittymä Git-toiminnoille ja komennoille. Se auttaa visualisoimaan Git:n eri toiminnot ja antaa myös selkeän kuvan siitä, mitä muutoksia tietyissä tiedostoissa on tapahtunut. Tämä auttaa muutosten muistamisessa. Lisäksi se luo selkeän aikajanan eri muutoksille, josta oli paljon hyötyä tämän insinöörityön kirjoittamisessa. (17.)

Tässä projektissa ei ollut tarvetta suurimmalle osalle Git:n toiminnoista, koska versiohallintaa käyttäviä kehittäjiä oli vain yksi. Se oli kuitenkin hyödyllinen

vanhoihin sovellusversioihin palaamisessa, jota jouduttiin käyttämään monta kertaa.

4.4 Java

Java-ohjelmointikieli julkaistiin vuonna 1995. Se on kehittynyt alkutilanteestaan huomattavasti, ja nykyään sitä käytetään suuressa osassa maailman teknologioista. Tämä on kyetty saavuttamaan tekemällä Javasta luotettava alusta, jonka avulla on mahdollista kehittää monia sovelluksia ja palveluita. (18.)

Java on ohjelmointikielenä monialustainen, olio-ohjelmointia käyttävä ja tietoverkkopohjainen. Se on lisäksi nopea, turvallinen ja luotettava. (19.)

Java on nykyään suosittu monesta eri syystä. Näistä muutamia ovat korkealaatuiset opintomateriaalit, sisäänrakennetut funktiot ja kirjastot, aktiivinen yhteisön tuki sekä toimivuus monilla eri alustoilla, kuten Windowsilla, Linuxilla, iOS:llä sekä Androidilla, ilman uudelleenkirjoitusta. Tämä avustaa varsinkin nykymaailmassa, jossa sovelluksia halutaan käyttää monilla eri alustoilla. (19.)

Java valittiin sovelluksen ohjelmointikieleksi, koska kehittäjillä oli jo aiempaa kokemusta siitä Metropolian aiempien projektien kautta. Lisäksi aiemmin mainitut laajat ja hyvälaatuiset oppimateriaalit auttoivat sen valinnassa.

4.5 Jira

Jira on tarkoitettu tehtävien seurantaan, projektien hallinnointiin ja työn suoraviivaistamiseen. Sen on kehittänyt Atlassian, joka on myös kehittänyt *Sourcetreen*. Jira perustuu neljään konseptiin, joita ovat asiat, projektit, taulut ja työvirrat. (20.)

Asia on Jiran yhteydessä yksittäinen tehtävä, jota seurataan sen luonnista sen viimeistelyyn asti. Se voi olla esimerkiksi bugi, testi tai tarina. Projektit Jirassa ovat käytännössä keino monen asian yhdistämiseen ns. yhden nimen alle.

Yleensä yhdistettävät asiat omaavat jonkin yhteisen elementin tai kontekstin. Taulut taas näyttävät visuaalisesti tiimin jäsenille heidän työvirtansa projektissa. Taulut liittyvät läheisesti sprinttityöskentelyyn, josta kerrotaan myöhemmin enemmän. Viimeinen asia ovat työvirrat. Työvirta kuvaa polkua, jota pitkin projektin asiat kulkevat alusta loppuun. Työvirtaan kuuluvat esimerkiksi sellaiset vaiheet, kuten ”tulossa tehtäväksi” ja ”työ kesken”. Näitä voidaan eri asioiden välillä muokata, esimerkiksi jotkin asiat voidaan määritellä siten, että niitä ei voi laittaa ”työ kesken” -vaiheeseen ollenkaan. (20.)

Tässä projektissa Jiraa käytettiin lähinnä sprinttityökaluna. Sinne lisättiin työnantajan toimesta kaikki ne asiat, joita seuraavien kahden viikon aikana haluttiin lisätä sovellukseen. Yleisesti tästä olisi enemmän hyötyä suuremmalla työntekijäkokoonpanolla, mutta siitä oli hyötyä myös yksittäiselle sovelluskehittäjälle tässä projektissa.

5 Sovelluksen kehitys ja iteraatiot

Sovelluksen kehitystä käsitellään tässä luvussa laajasti. Ensimmäisessä aliluvussa käydään läpi kehitystyötä ennen sprinttien käyttöönottoa, joten se on varsin irrallinen toimintatutkimuksesta. Toisessa aliluvussa käydään läpi kuusi sprinttiä. Näissä kerrotaan aluksi siitä, mitä tehtiin ja miksi. Tämän jälkeen katsotaan toimintatutkimuksen mallin mukaisesti, mitä havaintoja tehtiin sprintin aikana. Lopuksi reflektoidaan eli katsotaan, mitkä asiat toimivat, mitkä eivät ja miten kehitystyötä muutettiin havaintojen perusteella. Kaikista muutoksista ei kerrota, vaan sprinteissä keskitytään olennaisimpiin muutoksiin.

5.1 Kehitys ennen sprinttien käyttöä

Tässä aliluvussa käydään läpi sovelluksen kehityksen ensimmäinen vaihe ennen sprinttien käyttöönottoa.

5.1.1 Alkuvaiheet ja BLE:n implementointi

Sovellusta kehitettiin ennen sprinttien alkua noin 1,5 kuukautta. Tämä alkuvaihe käytettiin BLE:n ominaisuuksien käyttöönottoon. BLE:tä tarvittiin sovelluksessa tiedonsiirtoon ulkoisen painematon ja sovelluksen välillä. Tämän mahdollistamiseksi kaksi asiaa tuli tehdä ensimmäiseksi. Näitä olivat lupien kysyminen ja laitteiden skannaus.

Aluksi sovelluksen luettelotiedostoon lisättiin BLE:n vaatimat luvat. Toimiakseen Androidilla BLE vaatii luvat *bluetooth*-skannaukseen, *bluetooth*-mainostamiseen, *bluetooth*-yhdistämiseen sekä tietyissä tapauksissa myös mobiililaitteen sijaintiin. Mainitut kolme *bluetooth*-lupaa ovat ns. *runtime* lupia, eli ne täytyy pyytää sovelluksen käyttäjältä suoraan jossain vaiheessa sovelluksen elinkaarta. Androidilla on tätä varten valmiiksi rakennetut viestit, joten lupien ohjelmointi oli yksinkertaista. (21.)

BLE:n toiminta vaatii mobiililaitteessa *Bluetooth*-adapterin. Tämä on helppo hakea ohjelmoinnin kautta Androidilla. Kun tämä nopea haku on suoritettu, ohjelma tietää, että bluetooth-toiminnot ovat mahdollisia. Skannaus itsessään on yksinkertaista aloittaa, mutta sen tulokset tulevat ohjelmaan ns. takaisinkutsun kautta. Tässä takaisinkutsussa on sitten mahdollista hakea haluttu laite ja myöhemmin yhdistää siihen. Skannausta on mahdollista rajata suodattimen avulla. Yksinkertaisin suodatin tämän sovelluksen käyttötarkoitukseen tässä alkuvaiheessa oli käyttää painematon MAC-osoitetta. Kun tämä saatiin selvitettyä käyttämällä erillistä BLE-skannaus sovellusta, haku pystyttiin rajaamaan vain haluamaamme laitteeseen. (22.)

Näiden ominaisuuksien lisäyksen jälkeen otettiin *Sourcetree* käyttöön, ja tämä oli ensimmäinen ns. *commit*. *Commit* on tallenne sovelluksen koodista tallennushetkellä. Näihin versioihin on mahdollista palata myöhemmin takaisin. Tässä vaiheessa sovelluksen käyttöliittymä koostui kahdesta napista, jotka aloittivat ja lopettivat skannauksen, sekä listasta, jossa löydetyt ulkoiset laitteet näkyivät.

Seuraava vaihe oli varsinaisen yhteyden luominen laitteiden välillä. Tähän tarvitaan aiemmin tekstissä mainittua GATT-profiilia. Sovellukseen lisättiin yhdistäminen painematon GATT-palvelimeen. Tämä oli yhden rivin operaatio, jonka avulla saatiin viittaus tämän painematon palvelimeen. Tämä toiminnallisuus vaatii oman takaisinkutsunsa, eli GATT-takaisinkutsun. Sen kautta on mahdollista käsitellä yhteyden tilaa sekä laitteen palveluja ja ominaisuuksia.

Tärkeä osa BLE:tä Androidilla on toimintojen synkronointi. Normaalisti nämä toiminnot ovat asynkronoituja, eli ne voivat tapahtua samanaikaisesti. BLE vaatii virheettömään toimintaan kuitenkin monia peräkkäisiä toimintoja. Esimerkiksi aluksi yhdistetään GATT-palvelimeen. Sitten haetaan GATT-ominaisuuksia, joiden kautta päästään käsiksi laitteen palveluihin. Näiden palveluiden lähettämä data saadaan takaisin GATT-takaisinkutsussa, jossa sitä muokataan ja siirretään muun sovelluksen käyttöön. Mikäli yksikin näistä vaiheista ei ole synkronoitu, on mahdollista, että koko toiminto katkeaa. Synkronointia varten lisättiin joitain toimintoja, joiden ansiosta koko toimintojono saatiin synkronoitua.

Sovelluksen ohjelma oli tässä vaiheessa vielä samassa Android-aktiviteetissa. Minkäänlaisia ulkoisia luokkia ei vielä ollut. Seuraava vaihe oli varsinaisen lämpökartta-aktiviteetin luominen. Androidilla aktiviteetti on aina yksi sivu, jolla on käyttöliittymän elementtejä. Erilliset luokat taas ovat samantapaisia, mutta niillä ei ole käyttöliittymää, joten ne yleensä kommunikoivat jonkin aktiviteetin kanssa. Ennen lämpökartta-aktiviteettia sovelluksessa oli vain etusivu, jossa tässä luvussa mainittu ohjelma sijaitsi. Lämpökartta-aktiviteetin luomisen jälkeen kaikki tuleva ohjelmointityö keskittyi lämpökarttaan. Painematto ei ollut käyttökunnossa vielä tässä vaiheessa, eikä lämpökartan toimintoja ollut vielä ohjelmoitu.

5.1.2 Lämpökartta, käyttöliittymä ja BLE:n viimeistely

Painematon toiminta aiheutti ongelmia, sillä sen käyttöohjeet olivat harhaanjohtavia. Matto toimi ohjeiden mukaan siten, että yhdistämisen jälkeen yhteen sen BLE-ominaisuuksista kirjoitetaan kirjain R. Tämä aktivoisi maton lähettämään matriisin sen painearvoista. Tätä toimintoa tutkittaessa huomattiin, että GATT-

takaisinkutsun metodi *onCharacteristicChanged*, joka on Androidilla BLE:n vakiometodi, vastaanotti riveittäin dataa matolta. Tämä oli siis se paikka, jossa painematriisi voitiin vastaanottaa.

Kun tämä ongelma ratkaistiin, datan siirto avaussivulta lämpökarttasivulle tuli mahdolliseksi. Lämpökarttaa varten otettiin käyttöön ulkoinen kirjasto, nimeltä *HeartlandsSoftware Android HeatMap*. Se on käytännössä muokattu versio Androidin näkymästä, jolla on joitain omia metodeja lämpökartan värien suhteen. Tämä helpotti ohjelmointiprosessia. Tätä kirjastoa käytettiin koko projektin ajan. Data saatiin näkymään sovelluksessa, mutta se ei ollut oikeassa muodossa, eikä se ollut jatkuvaa, koska sitä siirrettiin yksi matriisi kerrallaan aina, kun siirryttiin aktiviteetista toiseen.

Seuraava vaihe oli BLE-ohjelmoinnin siirto toiseen luokkaan. Tämä mahdollisti sen helpomman käytön useammassa aktiviteetissa. Toinen lisäys oli säädin, jonka avulla käyttäjä pystyi vaikuttamaan siihen, kuinka suuret painearvot näkyivät lämpökartalla. Lämpökartan näkymä saatiin korjattua. Painematon lähettämä matriisi oli oudossa muodossa, eli matriisissa, joka ei ollut leveydeltään ja korkeudeltaan samankokoinen. Tämä ongelma oli kuitenkin ilmentynyt jo aiemmin projektin aikana, joten korjaus siihen oli tiedossa. Tässä vaiheessa aiemmin mainitut skannausnapit lisättiin myös tähän lämpökartta-aktiviteettiin, jotta kaikki toiminnallisuus olisi samalla sivulla.

Seuraavaksi tehtiin edellä mainittu datan muokkaus sopivampaan muotoon. Tähän tarkoitukseen otettiin uusi API käyttöön, eli *Apache Commons:n RealMatrix*. Tämä on natiivi-Javan tyylinen matriisi, mutta se mahdollistaa suuremman muokkauksen jo olemassa olevaan matriisiin, kun Javan omissa matriiseissa muokkausta ei voi tehdä ilman uuden matriisin luomista. Matolta saatu data muokattiin siten, että siitä muodostui 16x16-matriisi, jolla painearvot sitten jakautuivat oikeaan muotoon.

Kun samalla sivulla oli niin paljon erilaista toiminnallisuutta sekä jatkuvaa datan käsittelyä ja esittämistä, siirrettiin seuraavaksi suuri osa tästä ohjelmoinnista toimimaan eri ”langassa”, eli toisin sanoen taustalla. Aluksi implementoitiin natiivi-

Javan *Runnable*-metodeja eri vaiheisiin. Tämä mahdollisti ohjelman toiminnan taustalla, mutta aiheutti synkronointiongelmia. Tässä vaiheessa implementoitiin *switch*-nappi sovellukseen, josta käyttäjä voi aloittaa datan keräyksen. Yksi ongelma tämän kanssa oli, että koko sovelluksen käyttöliittymä pysähtyi matriisin vastaanottoon asti.

Datan keräystä varten sovellukseen otettiin mukaan RXJava-kirjasto. Lyhyesti selitettynä se on Javalla toimiva versio ReactiveX-nimisestä kirjastosta. Tämä kirjasto perustuu *observable*-objekteihin ja tapahtumaperäiseen ohjelmointiin. Se mahdollistaa suoraviivaisen synkronoinnin tapahtumien välillä sekä helpon käyttöliittymän päivityksen missä tahansa vaiheessa Javan *Runnable*-luokasta poiketen. ReactiveX hoitaa asiat, kuten synkronoinnin ja lankojen turvallisuuden itsenäisesti. Natiivi-Javassa monet eri samanaikaiset toisiinsa liittyvät *runnable*-metodit voivat häiritä toistensa toimintaa, kun taas ReactiveX:n avulla kaikki tapahtuu samassa synkronoidussa putkessa. (22.)

Painematto lähetti uuden kartan melkein kerran sekunnissa, joten RXJavaa käyttämällä tämä synkronointi saatiin suoraviivaistettua. Lisäksi käyttöliittymän, eli tässä vaiheessa lämpökartan, päivittäminen oli paljon helpompaa RXJavalla. Tästä oli myös myöhemmissä vaiheissa hyötyä, kun käyttöliittymään lisättiin muuta toiminnallisuutta.

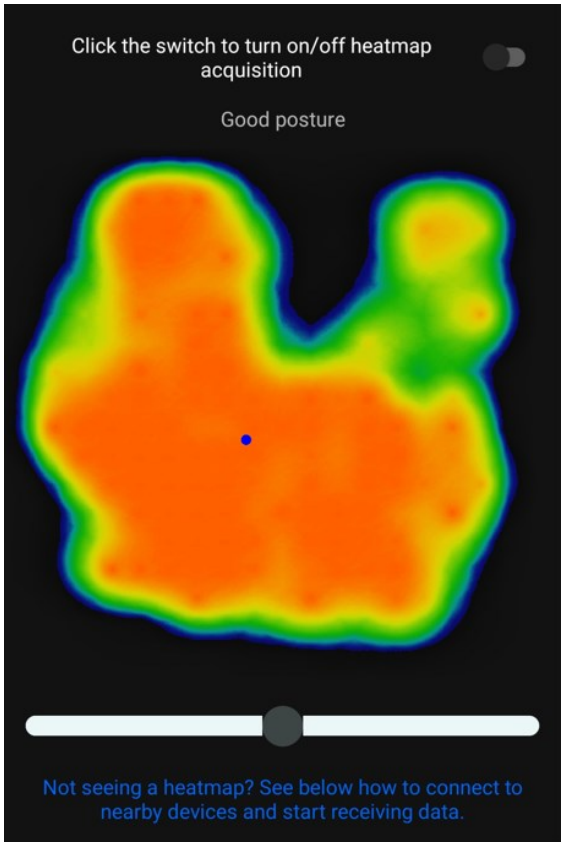
5.2 Ensimmäinen sprintti

Ennen ensimmäisen sprintin alkua sovelluksessa oli siis toimiva lämpökartta, joka käyttää painematolta saatua dataa pienen muuntelun jälkeen, josta lämpökartan värit muodostetaan. Lisäksi maton värigradientti oli varsin yksinkertainen, eli keltaisesta punaiseen. Siirtymäkohtia väreille oli vain kolme tässä vaiheessa, joten gradientti ei ollut vielä valmis. Lämpökarttasivulle lisättiin ohjeita käyttäjälle sivun navigointiin ja mittauksen aloittamiseen.

Tässä ensimmäisessä sprintissä keskityttiin sovelluksen algoritmeihin. Ensimmäisen vaiheen algoritmeja olivat painekeskipisteen mittaus, alkukantainen

asennontunnistus sekä huippupaineiden tunnistaminen. Projektin datatieteilijällä oli jo ennestään Python-kielellä tehdyt algoritmit, jotka käännettiin Javalle. Painepisteen hakeminen oli varsin yksinkertaista, eli otettiin koko painekartan arvot ja niiden sijainnit ja niiden perusteella laskettiin, missä painekeskipiste oli.

Seuraavaksi sovellukseen lisättiin asennontunnistus. Tämä tehtiin siten, että painekartta jaettiin neljään osaan, jonka jälkeen painekeskipistettä seurattiin kartalla. Asentoja oli neljä, eli eteen- ja taaksenojaus, sekä vasemmalle- ja oikealle nojaus. Tämä versio ei ollut kovin toimiva, koska mattoa ei saatu pilkottua halutulla tavalla. Yhdessä vaiheessa esimerkiksi oikea nojaus tapahtui silloin, kun painepiste oli vasemmassa yläkulmassa. Tämä jätettiin tässä vaiheessa tähän, ja siirryttiin muihin algoritmeihin. Kuvassa 4 on näkyvillä sovelluksen lämpökartta sekä painekeskipiste, jonka perusteella on näkyvillä myös arvioitu asento.



Kuva 4. Sovelluksen lämpökartta 16x16-matriisina.

Viimeinen uusi asia tässä sprintissä oli korkeiden painepisteiden algoritmin kehitys. Tämä päätettiin tehdä sillä periaatteella, että jokaista kartan pistettä seurataan ja katsotaan, onko piste korkein kaikista sitä ympäröivistä pisteistä. Jos se ei ole, niin se ei myöskään voi olla korkein piste. Tällä logiikalla tehtiin totta/ei totta-matriisi, joka yleensä löysi 3–5 mahdollista korkeinta pistettä. Toinen tähän liittyvä lisäys oli *OpenCV*, joka on mm. Javan kuvien muokkaamiseen keskittyvä kirjasto. Tämä lisättiin mustaharmaakuvien ja gaussin sumennuksen mahdollisuuden takia. Gaussin sumennus on käytännössä suodatinfunktio kuvalle, jota halutaan sumentaa. Tämä haluttiin tehdä siksi, koska painematon lähettämän matriisin resoluutio oli varsin matala. Suodatin poistaa tietyn kokoisen liikkuvan neliön sisällä kaikista suurimman arvon ja lisää sen paikalle pienemmän arvoisen pikselin. Mitä suurempi tämä neliö on, sitä suodattuneempi kuva on. Tästä on esimerkki kuvassa 5, jossa on näkyvillä alkuperäinen kuva, 3x3-neliö sekä 10x10-neliö.

Kuva 5. Esimerkki Gaussin sumennuksesta eri suuruuksilla. (23)

Tämän suodatuksen avulla virheelliset yksittäiset korkeat painearvot saatiin poistettua, ja korkeimmat pisteet päätyivät todennäköisemmin niihin paikkoihin, missä monet vierekkäiset pikselit olivat korkeita arvoja.

Tässä vaiheessa toimintatutkimuksen elementit tulivat mukaan sovelluksen kehitykseen. Tässä luvussa on jo kerrottu, mitä sprintissä tehtiin. Seuraavaksi kerrotaan havainnoista. Tässä sprintissä tavoitteena oli saada algoritmit toimimaan sovelluksessa, ja tähän tavoitteeseen päästiin. Sprintin lopuksi reflektoitiin sitä, miten kehitystyötä voisi parantaa. Tässä sprintissä ilmeni, että tehtäviä oli liian vähän ja halutut asiat oli jo saatu implementoitua muutama päivä ennen sprintin loppua. Tämän seurauksena seuraavaan sprinttiin päätettiin ottaa enemmän tehtäviä. Sovelluksen kehityksen ei tarvinnut olla niin rajattua kuin normaalisti, koska projektissa ei ole asiakasta, jolle sovelluksen versio pitäisi toimittaa valmiina käyttöä varten. Seuraavaan sprinttiin päätettiin siis lisätä tehtäviä, ja kaikkia niitä ei välttämättä tarvinnut saada valmiiksi sprintin loppuun mennessä.

5.3 Toinen sprintti

Nyt kun ensimmäiset algoritmit oli implementoitu, niiden toiminnan testaus oli seuraavana vuorossa. Tätä varten oli varattu laboratoriokäynti, jossa SCI-potilas kokeili sovellusta. SCI-potilas tarkoittaa selkäydinvamman saanutta potilasta. Selkäydinvammat voivat vakavimmillaan olla täysi halvaantuminen joko lantiosta, tai niskasta alaspäin. Jotta kerätystä mittaustiedosta olisi jotain hyötyä, tarvitsi sovellus jonkin tavan tallentaa dataa, joka mahdollisti sen käytön joko sovelluksen ulkopuolella tai sisällä. Toinen tärkeä asia testiä varten oli sovelluksen tasaisuuden parantaminen. Siinä oli tässä vaiheessa vielä monta pientä ongelmaa, jotka saattoivat aiheuttaa kaatumisen.

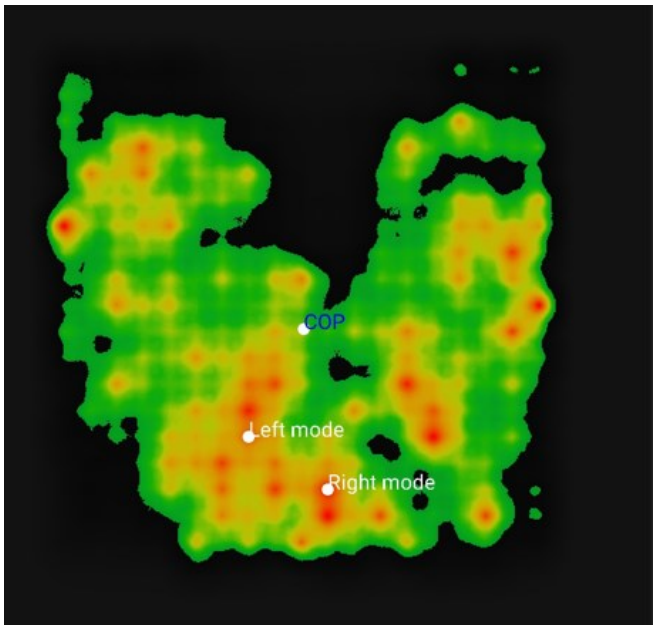
Tiedon tallentamiseen valittiin käytettäväksi tietokanta nimeltään *Android Room Database*. *Room* DB pohjautuu SQLite:n, joka taas on itsenäinen, ja helpompikäyttöinen versio SQL-ohjelmointikielestä. SQL on tehty tiedon tallennukseen ja informaation prosessointiin relationaalisessa tietokannassa. Relationaalinen tietokanta säilyttää tietoa taulukkomuodossa, jossa rivit ja sarakkeet vastaavat

säilötyn tiedon ominaisuuksista ja yhteyksistä toisiinsa. SQL:llä on mahdollisuudet tiedon tallennukseen, päivittämiseen, poistamiseen, etsimiseen, sekä hakuun. *Room* DB on Androidin versio SQLite:stä, jossa sen käyttöä on helpotettu turhan ohjelmoinnin vähentämiseksi. Android itse suosittelee, että sovelluskehittäjät käyttävät *Room* DB:tä itsenäisen SQLite:n sijaan. (24; 25; 26.)

Tässä vaiheessa ainoa tarve kerätyn tiedon tallennukseen oli se, että painematon lähettämät kartat voitaisiin tallentaa johonkin. Lisäksi haluttiin sovelluksen sisäinen keino entisten mittausten uudelleenkatseluun. Tältä pohjalta päätettiin siis tallentaa *Room* DB -tietokantaan jokainen painekartta sekä niiden tallennusaika. Tallennuksen jälkeen koko tietokannan sisältö voitiin sitten näyttää sovelluksessa listana, josta käyttäjä pystyi tässä vaiheessa valitsemaan haluamansa painekartan, ja näkemään sen näytöllä lämpökarttana.

Ainoat lisäykset tässä vaiheessa datan tallennukseen olivat tallenteiden poistaminen sekä se, miten käyttäjältä kysytään, mitkä mittaukset tallennetaan ja mitä ei. Poistamista varten lisättiin ponnahdusikkuna, mikäli tiettyä mittausta listassa painettiin sekunnin ajan. Erillisen poiston sijaan lisättiin toinen ikkuna ennen mittauksen alkua, jossa käyttäjältä kysyttiin, halutaanko kerättävää dataa tallentaa. Lisäksi lisättiin nappi, jota painamalla koko tietokanta tyhjeni.

Seuraava vaihe oli painematon vaihto. Käyttöön otettiin uudempi versio vanhasta matosta, joka aiheutti suurta muokkausta eri puolille sovellusta. Kun edellisen maton matriisikoko oli 16x16, uusi matto oli 20x20. Muuten se oli sovelluksen kannalta identtinen, joten BLE-ohjelmointia ei tarvinnut muuttaa. Hyvä puoli tässä uudessa matossa oli parempi kuvanlaatu. Kuitenkin vanha raakadatan muokkausmetodi ei enää toiminut, ja sitä jouduttiin muuttamaan. Lisäksi kaikki algoritmit jouduttiin myös sovittamaan tähän uuteen matriisikokoon. Viimeinen tästä aiheutunut ongelma oli lämpökartan väreihin liittyvä. Kolmen värin gradientti ei enää uuden maton kanssa riittänyt, joten gradientti päätettiin muodostaa 10 väristä. Tämän muutoksen seurauksena korkeat painepisteet erottuivat entistä paremmin. Kuvassa 6 on näkyvillä uusi lämpökartta sovelluksessa.



Kuva 6. Sovelluksen lämpökartta 20x20-matriisina.

Ennen sprintin loppua lisättiin vielä muutama visuaalinen merkki lämpökarttaan. Paineen keskipiste, sekä korkeimmat painearvot saivat omat pisteet kartalle, ja ne eroteltiin nimiteksteillä.

Tässä sprintissä kaikkia valittuja tehtäviä ei saatu valmiiksi ennen sen loppua. Tärkein tehtävä saatiin kuitenkin valmiiksi, eli painematon vaihto. Sprintti oli suurelta osin onnistunut, koska ennen sen alkua tehtäviä päätettiin lisätä hieman enemmän. Sovellus saatiin käyttökuntoon myös laboratoriomittauksia varten, tosin kaikkia virheitä ei ehditty korjata. Näiden tulosten pohjalta reflektoitiin, että tulevassakin sprintissä tehtäviä lisätään suurempi määrä. Suurin sprintin aikana tapahtunut muutos oli uuden painematon implementointi, joka oli ensimmäinen prioriteetti. Tämän onnistumisesta ei ollut varmuutta, koska painematon saapumispäivä ei ollut varma ennen sprintin alkua.

5.4 Kolmas sprintti

Ennen kolmannen sprintin alkua mietittiin uudelleen BLE:n toimintaa sovelluksessa. Nykyisessä muodossaan se toimi hyvin yhden sivun sovelluksessa, mutta koska sovellusta aiottiin laajentaa tulevaisuudessa, BLE vaati hieman

uudistamista. Ensinnäkin BLE-skannauksen tuloksia ei tullut ruudulle ollenkaan. Tämä oli yksi asia, joka haluttiin muuttaa. Toinen oli nopeampi yhdistäminen, joka tällä hetkellä vaati vielä skannauksen ja sitten yhdistämisen. Androidin BLE:llä on mahdollista yhdistää automaattisesti jo ennestään skannattuun ja tunnistettuun laitteeseen, jota pidettiin hyvänä lisäyksenä.

Aluksi sovellukseen yritettiin lisätä pudotusvalikko, josta kaikki skannatut laitteet voisi nähdä, ja niistä voisi valita halutun laitteen, johon yhdistetään. Ideana tässä oli, että käyttäjällä olisi mahdollista tehdä yhdistäminen vaiheittain, jolloin hän ei esimerkiksi aloittaisi mittausta liian aikaisin. Hetken mietinnän jälkeen todettiin, että tämä ei ole tarpeellista tässä vaiheessa, ja tähän käyttäjälle kommunikointiin oli parempi keino. Tämä keino olivat lähetykset (*Broadcasts*).

Androidilla lähetykset ovat tapa, jolla luokat voivat kommunikoida keskenään. Yksi luokka voi lähettää näitä lähetyksiä, joiden mukana voidaan lähettää joko dataa, tai ne voivat olla ilmoitus jonkin asian tapahtumisesta kyseisessä luokassa. Toinen luokka, jossa on lähetysten vastaanottamista varten oma ohjelmointinsa, voi sitten tehdä mitä tahansa, kun lähetys on vastaanotettu. Tämän sovelluksen näkökulmasta lähetyksiä käytettiin siten, että seurantakoodi lisättiin BLE-luokkaan. Tämä luokka sitten kommunikoi lämpökarttasivun kanssa. BLE-luokan lähetykset olivat käytännössä ilmoituksia BLE-yhteyden tilanteesta. Nämä kaikki liittyivät johonkin yhteyden muodostuksen vaiheeseen. Näitä varten lämpökarttasivulle lisättiin sitten ruudulle ilmoitustekstiä riippuen yhteyden tilasta, jotta käyttäjällä olisi parempi ymmärrys yhteyden tilasta. Lisäksi mittauksen aloittamisen päälle/pois päältä-katkaisija muutettiin sellaiseksi, että sitä ei voi laittaa päälle, jos yhteyttä ei ole.

Toinen asia, joka lisättiin, oli siis automaattinen yhdistys jo tunnettuun BLE-laitteeseen. Tämän seurauksena yhteys laitteeseen tapahtui muutamassa sekunnissa, kun aiemmin siinä kesti yli kymmenen sekuntia. Tämä nopea yhteyden luominen perustuu siihen, että jokaisella BLE-laitteella on oma MAC-osoitteensa, jonka kautta tiedossa olevat laitteet voidaan tunnistaa.

Kolmas lisäys tässä sprintissä oli uusi algoritmi. Tämän algoritmin ideana oli seurata paineen määrää matossa pidemmällä aikavälillä. Sen nimeksi tuli kumulatiivinen paine. Käytännössä se toimi siten, että jokainen matriisi, jonka lämpökartta lähetti lisättiin pienen muokkauksen jälkeen toiseen matriisiin, joka oli siis kumulatiivinen paine. Muokkaus, joka matriiseihin tehtiin, oli käytännössä arvojen pienennys, mutta tärkeänä osana oli myös kynnysarvon käyttö. Tämä kynnys varmisti sen, että erittäin pieniä arvoja ei lisätty ollenkaan kumulatiiviseen stressiin. Lisäksi tärkeää oli myös sen varmistaminen, että kumulatiivinen stressi laski realistisesti, eikä vain mennyt nollaan siinä vaiheessa, kun paine katosi jonkin pisteen kohdalta. Tätä varten lisättiin myös funktio, joka otti huomioon kynnysarvon, sekä esimerkiksi vakioarvon, jonka perusteella paine laski nopeammin kuin se nousi.

Tämä sprintti loppui uusien lisäysten osalta tähän. Tavoitteisiin päästiin varsin hyvin, mutta esimerkiksi kumulatiivisen stressin perusteella annettuja hälytyksiä ei vielä ehditty lisätä sovellukseen. Kumulatiivisen stressin algoritmi ei myöskään ollut vielä viimeisessä muodossaan. Reflektoinnissa päätettiin, että tämän algoritmin viimeistely olisi seuraavan sprintin prioriteetti, koska tämä paineen seuranta oli yksi suurimmista hyödyistä käyttäjälle tässä sovelluksessa. Suunnitelmaa ei jouduttu muuttamaan sprintin aikana, joten kehitystyö sujui sulavasti. Jotkin ideat sprintin aikana eivät toimineet halutusti, joten lopullista ratkaisua jouduttiin muuttamaan. Näin kävi BLE-lähetysten kohdalla, jotka olivat toinen vaihtoehto yhteyden luomisen helpottamiseen. Ne kuitenkin osoittautuivat sopivammaksi ratkaisuksi kuin alkuperäinen idea.

5.5 Neljäs sprintti

Neljännessä sprintissä haluttiin viimeistellä ensimmäisen version kumulatiivisesta stressialgoritmista. Tähän kuului hälytysten lisääminen kertyneen paineen perusteella sekä erillisen tiedoston luominen ja yhdistäminen algoritmin arvoihin, jotta niitä voitiin muokata nopeasti. Toinen lisäys oli toisen lämpökartan implementointi. Tämä kartta tulisi näyttämään kumulatiivisen paineen käyttäjälle. Viimeinen iso asia, joka haluttiin lisätä, oli mahdollisuus vanhan datan

läpikäyntiin automaattisesti. Tähän asti vanhan datan katselu oli tehty vaikeasti, eli jokainen katsottava lämpökartta piti valita erikseen listasta. Sovellukseen lisättiin myös uusi versio kumulatiivisen stressin algoritmista.

Notifikaatioiden lisääminen Androidilla vaatii tietyn luvan käyttäjältä, jotta ne tulevat esiin laitteessa. Tämä ei ollut tässä vaiheessa ongelma, koska testilaitteet olivat tiimin omia. Lisäksi hälytykset vaativat uusissa Android-versioissa kanavan, jonka kautta ne lähetetään. Tämä mahdollistaa sen, että käyttäjä voi valikoida, mitä hälytyksiä hän haluaa vastaanottaa sovellukselta. (27.) Tämän sovelluksen kannalta haluttiin lähettää notifikaatiot siinä tapauksessa, jos kumulatiivinen paine nousi liian korkeaksi. Se raja, joka sovellukseen laitettiin, oli täysin arvailua, mutta tässä vaiheessa kumulatiivisen stressin matriisiarvot olivat välillä 0–256. Rajaksi laitettiin 200, eli mikäli kumulatiivinen stressi jossain kohdassa kehoa kasvoi yli 200:n, sovellus lähetti hälytyksen. Notifikaatioita lähetettiin sen perusteella, millä painekartan alueella paine oli kasvanut yli tämän rajan.

Toisen lämpökarttanäkymän lisäys tapahtui varsin samalla lailla ensimmäisen kanssa. Ainoa ero oli, että tässä näkymässä käytettiin kumulatiivista matriisia. Tätä varten lisättiin myös nappi, jota painamalla käyttäjä voi vaihtaa näiden kahden eri lämpökartan välillä. Tähän uuteen lämpökarttaan ei lisätty korkean paineen pisteitä tai painekeskipistettä.

Vanhojen lämpökarttojen katselua varten lisättiin tässä vaiheessa hieman ohjelmointia, mutta se ei ennen tämän sprintin loppua vielä toiminut kunnolla. Viimeiseksi lisättiin uudempi versio kumulatiivisesta stressialgoritmista. Tämä versio oli suoraviivaistettu versio vanhasta algoritmista. Tämä johti siihen, että notifikaatiot eivät enää toimineet. Tämä oli seurausta siitä, että vanha notifikaatio-ohjelmointi käytti *OpenCV*:n arvoja rajana, mikä toimi vielä siinä vaiheessa, kun kumulatiivinen stressi pohjautui *OpenCV*-koodiin. Uusi algoritmiversio käytti puolestaan aiemmin mainittua *RealMatrix*-matriisia, joka ei ollut yhteensopiva vanhan koodin kanssa. Hyvä puoli tässä muutoksessa oli se, että sovellus ei enää käyttänyt kahta eri matriisityyppiä, mutta notifikaatiot jouduttiin korjaamaan myöhemmin.

Tässä sprintissä oli paljon eri tehtäviä, mutta ne kaikki joko keskittyivät vanhan ohjelmoinnin muuttamiseen, tai sen uudelleenkäyttöön toisen painekartan tapauksessa. Tavoitteisiin päästiin suhteellisen hyvin, mutta vanhan datan uudelleentoistoa ei saatu kovin pitkälle. Yli jääneiden tehtävien osalta tämä sprintti eteni heikoiten kaikista tähänastisista sprinteistä. Tämän seurauksena päätettiin, että seuraavassa sprintissä korjattaisiin ongelmia ja viimeisteltäisiin edellisiä tehtäviä uusien ominaisuuksien lisäämisen sijaan. Sprintti itsessään eteni suunnitelmien mukaan. Ainoa ongelma oli ajan loppuminen kesken.

5.6 Viides sprintti

Tässä sprintissä oli muutama pienemmän skaalan asia, mitkä haluttiin korjata. Notifikaatioiden toiminta, mittauksen kytkimen logiikan muutto, sekä tietokannan logiikan muuttaminen. Lisäksi oli pari pienempää käyttöliittymän asiaa, jotka vaativat muutoksia. Isoina lisäyksinä saatiin vanha mittausdata toistumaan sulavasti, jonka lisäksi *pipeline*-ohjelmointi aloitettiin.

Notifikaatiot saatiin korjattua pienen muokkailun jälkeen. Tässä oli siis ainoana tehtävänä se, että notifikaatioista vastaava metodi käyttää nyt *RealMatrix*-matriisia *OpenCV*:n matriisin sijaan. Tämä vaati uuden raja-arvon miettimisen, mutta muuten notifikaatiot saatiin toimimaan hyvin.

Mittauksen aloittamista varten oleva kytkin toimi tähän asti siten, että kun käyttäjältä kysytään halua mittausdatan tallennukseen, mittaus alkoi heti, vaikka käyttäjä ei olisi vastannut ruudulla näkyvään kysymykseen. Tämä haluttiin korjata järkevämmäksi. Tämä tehtiin siten, että varsinainen datan hakemisesta vastaava koodi eroteltiin kytkimen koodista, johon jätettiin ainoastaan koodi sitä varten, haluaako käyttäjä tallentaa mitattavan datan. Sitten kun käyttäjä klikkaa joko kyllä tai ei, mittaus voi alkaa.

Tietokannan logiikkaa muutettiin siten, että mittauksen aluksi käyttäjä voi tallennuskysymyksen yhteydessä antaa nimen mittaukselle. Tämä nimi tuli näkyviin vanhojen mittausten listassa. Tämä mahdollisti sen, että käyttäjällä olisi

kontekstia siihen vanhaan dataan, mitä hän katselee. Tämä tuli käytännössä vain kehitystiimin käyttöön, jotta pystyttiin erottamaan se, mitä testeissä on milläkin hetkellä tehty.

Vanhojen mittausten toistaminen saatiin toimimaan sulavasti. Tämä tehtiin siten, että sovelluksessa säilytettiin vanha lista, josta käyttäjä voi valita haluamansa mittauksen. Uudessa versiossa tämä toimi siten, että sovellus alkoi automaattisesti näyttämään vanhoja mittauksia yksi toisensa perään, lähtien siitä mittauksesta, jota käyttäjä on klikannut listassa. Tähän lisättiin myös viiden lämpökartan hypyt eteen- ja taaksepäin sekä pysäytysmahdollisuus toistolle.

Tässä sprintissä päätettiin ottaa käyttöön uusi ohjelmointifilosofia, eli ns. putkiohjelmointi (*pipeline*). Putkiohjelmointi tämän sovelluksen näkökulmasta toimii kahdella tavalla. Ensimmäinen putki on sellainen, jossa sisään- ja ulostulo ovat samassa muodossa eli matriisina. Putkiohjelmointi tässä tapauksessa mahdollistaa operaatioiden nopean lisäyksen ja poistamisen, koska ne kaikki käyttävät samaa matriisia metodeissaan. Toinen käyttökohde on se, kun sisääntulo on matriisi, mutta ulostuloja on monta, ja ne kaikki ovat eri muodossa kuin sisääntulo. Tämä oli astetta vaikeampaa, mutta sprintin aikana saatiin toimimaan kaikki paitsi kumulatiivinen stressi.

Tässä sprintissä oli paljon pienempiä muokkauksia, joiden katsottiin olevan tärkeitä. Niistä suurin osa joko suoraviivaisti sovelluksen käyttöä ja kehitystyötä, tai paransi käyttäjäystävällisyyttä. Tavoitteisiin päästiin hyvin. Putkiohjelmoinnin tiedettiin vaativan paljon aikaa, joten se päätettiin jättää alhaisemmalle prioriteetille muihin tehtäviin verrattuna. Sen viimeistelyn päätettiin tapahtuvan tämän sprintin jälkeen. Reflektoinnissa huomattiin, että sovelluksen lämpökarttasivu alkoi olla ominaisuuksien osalta valmis. Tämä tarkoitti sitä, että korkeintaan muutaman sprintin sisällä kehitystyö siirtyisi muuhun sovellukseen. Tämän perusteella päätettiin joitain avainalueita, joihin keskityttäisiin seuraavassa sprintissä.

5.7 Kuudes sprintti

Kuudes sprintti oli viimeinen tässä insinöörityössä käsiteltävistä sprinteistä. Se loppui 28.6.2023. Tässä sprintissä haluttiin viimeistellä edellisessä sprintissä aloitettu putkiohjelmointi sekä implementoida IIR-suodattimen korkeiden pisteiden hakuun. IIR-suodatin tarkoittaa *Infinite Impulse Response*-suodatinta, joka on eräs tapa signaalin suodatukseen digitaalisesti.

Pipeline-koodi saatiin valmiiksi, kun syy kumulatiivisen stressin toimimattomuudelle löydettiin. *Pipeline* loi aina uuden version koko luokasta, kun painekartta ajettiin siitä läpi, jonka takia kumulatiivinen stressi ei lisääntynyt missään vaiheessa. Korjaus tehtiin siten, että *pipelinen* ulkopuolella on erillinen matriisi, joka säilyttää kumulatiiviset painearvot. Tähän asti ne säilytettiin algoritmin luokassa, joka siis aiheutti edellä mainitun ongelman.

IIR-suodatin on yksi versio digitaalisesta signaalin suodatuksesta. Digitaalinen suodatus tehdään laskutoimitusten avulla. (28.) Sovelluksessa tätä suodatinta käytettiin siis korkeiden pisteiden tunnistukseen. Tämä päätettiin lisätä, koska korkeat pisteet liikkuivat välillä epäluontevasti, vaikka itse lämpökuva pysyi samanlaisena. Suodatin implementoitiin siten, että tehtiin luokka, jossa seurataan aiempia korkeiden pisteiden sijainteja ja verrataan tätä pisteiden tämänhetkiseen sijaintiin. Tämä vähensi pisteiden liikkumista huomattavasti.

Havainnoinnin osalta tässä sprintissä saatiin kaikki paitsi yksi tehtävä valmiiksi. Tehtäviä ei ollut montaa, mutta ne olivat suuremman skaalan tehtäviä kuin monissa aiemmissa sprinteissä. Tavoitteisiin päästiin siis hyvin, mutta ei täydellisesti.

Tässä viimeisessä sprintissä reflektoimme, että nykyinen korkeiden pisteiden tunnistus ei suodattimenkaan kanssa anna järkeviä sijainteja huippupisteille. Tämän perusteella päätettiin tulevaisuudessa tehdä uusi algoritmi, joka piirtää lämpökarttaan kolmion kuvaamaan häntäluun, ja molempien istuinluiden paikkoja. Toinen havainto liittyi *pipeline*-koodiin. *Pipeline* oli hyvä idea, koska sovelluksessa käytetään yhtä matriisityyppiä, josta pitäisi saada monta eri ulostuloa,

eli korkeat pisteet, keskipiste sekä kumulatiivinen paine. Kuitenkin sen varsinainen toteutus osoittautui vaikeaksi, ja järkevämpää olisi saattanut olla putkiohjelmoinnin jättäminen pois ja keskittyminen muihin sovelluksen osa-alueisiin.

6 Yhteenveto

Tässä työssä haluttiin käsitellä mobiilisovelluksen kehitystyötä toimintatutkimuksen metodeja apuna käyttäen. Sovelluksen kehityksellä oli vaatimuksia, joihin yritettiin päästä. Yhteenvedossa käydään lyhyesti läpi nämä asiat sekä se, miten toimintatutkimuksen metodeista oli hyötyä kehitystyössä.

Sovelluksen tavoitteena oli tarjota suurempaa hyötyä käyttäjille, kuin mitä muut vastaavanlaiset painehaavojen ennaltaehkäisyyn käytettävät sovellukset ovat tarjonneet. Suurin hyöty muihin sovelluksiin verrattuna ovat kehitetyt algoritmit sekä tieteeseen perustunut kehitystyö. Sovellus ei tämän insinöörityön rajauksen aikana ehtinyt vielä tulla valmiiksi käyttäjille, mutta kehitetyt algoritmit, ja niiden implementointi sovellukseen, saatiin valmiiksi hyvin.

Toimintatutkimus oli vahvasti mukana sprinttien läpikäynnissä. Sovelluksen kehitystyötä seurattiin sprinttien tehtävien avulla. Tehtävät saatiin valmiiksi suhteellisen hyvin jokaisessa sprintissä. Ensimmäisessä sprintissä kaikki tehtävät saatiin valmiiksi, jonka jälkeen päätettiin, että tehtäviä on parempi olla enemmän, vaikka kaikkia niistä ei saataisi valmiiksi sprintin aikana. Reflektoinnin osalta toimintatutkimuksesta oli eniten hyötyä kehitystyön aikana. Toimintaa muutettiin moneen kertaan sprinteissä opittujen asioiden perusteella.

Lisäksi ketterä kehitystyö oli sopiva tapa tämäntyyppisessä sovelluskehityksessä. Muutoksia tehtiin jokaisessa sprintissä, joissain tapauksissa muutettiin sitä sovelluksen osa-aluetta, johon keskitytään, kun taas muulloin saatettiin muuttaa sitä tapaa, jolla jokin sovelluksen osa-alue haluttiin implementoida. Tiimin sisäisen keskustelun ansiosta kehitystyö pysyi jatkuvana, ja olennaisiin asioihin voitiin keskittyä silloin, kun niille oli eniten tarvetta. Muunlaisella kehitystyöllä näiden muutosten tekeminen olisi luultavasti ollut vaikeampaa.

Ketterän kehityksen haasteisiin pystyttiin vastaamaan projektissa suhteellisen hyvin. Ketterän kehityksen jatkuvaluonteisuus ei ollut haitta, koska sitä käytettiin lähes koko sovelluksen kehityksen ajan. Työryhmä oli myös pieni, jonka takia kommunikaatio oli yksinkertaista ja helppoa. Kolmas haaste oli vaikeampi ratkaista. Sovellus on tarkoitettu terveydenhuoltoalalle, joten se vaati paljon määrittelyä. Tätä haastetta pystyttiin lieventämään, koska sovelluksen vaatimuksia ja tarpeita, sekä rajoituksia, oli jo ennen varsinaisen sovelluksen kehityksen alkua määritelty pitkään.

Toimintatutkimuksen metodeista oli paljon hyötyä projektin aikana. Hyvä puoli tässä oli myös se, että ketterän kehitystyön sprintit ja toimintatutkimuksen metodit ovat varsin samanlaisia. Toimintatutkimus tarjosi sprinttien suunnitteluun ja läpikäyntiin enemmän syvyyttä, kuin mitä pelkkä ketterän kehityksen metodien käyttö olisi tarjonnut.

Insinöörityön ansiosta tämän aiheen tai kyseisen sovelluksen jatkokehitystä voidaan jatkaa sen pohjalta. Insinöörityötä voidaan pitää ensimmäisenä iteraationa suuremmassa tutkimuksessa, mikäli toimintatutkimusta halutaan käyttää myös tässä hypoteettisessa tutkimuksessa. Lisäksi insinöörityö antaa pohjan, jolta sovelluksen jatkokehittäjät voivat katsoa, mitä on tehty tai yritetty jo aiemmin.

Lähteet

1 Bennett, Gary. Delaney, Carol. Posnett, John. 2004. The cost of pressure ulcers in the UK. Verkkoaineisto. Age and Ageing. British Geriatrics Society. <https://doi.org/10.1093/ageing/afh086>. Luettu 16.10.2023.

2 Black, Joyce. Santamaria, Nick. Ohura, Norihiko. 2016. Consensus document: Role of dressings in pressure ulcer prevention. Verkkoaineisto. <Consensus document: Role of dressings in pressure ulcer prevention - Wounds International>. Luettu 16.10.2023.

3 Shi, Chunhu. Dumville, Jo C. Cullum Nicky. 2018. Support surfaces for pressure ulcer prevention: A network meta-analysis. Verkkoaineisto. <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0192707>. Luettu 16.10.2023.

4 Koepp, Janine. Ym. 2020. The quality of mobile apps used for the identification of pressure ulcers in adults: systematic survey and review of apps in app stores. Verkkoaineisto. JMIR Publications. <https://mhealth.jmir.org/2020/6/e14266/>. Luettu 16.10.2023.

5 Koski, Pirjo. Kelo, Marjatta. 2019. Toimintatutkimus menetelmänä. Verkkoaineisto. Metropolia. <https://blogit.metropolia.fi/masterminds/2019/09/30/toimintatutkimus-menetelmana/>. Luettu 16.10.2023.

6 Jykämä, Jyrki. Toimintatutkimus. Verkkoaineisto. Tietoarkisto. <https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvali/tutkimusasetelma/toimintatutkimus/>. Luettu 4.11.2023.

7 George, Tegan. 2023. What is Action Research? | Definition and Examples. Verkkoaineisto. Scribbr. <https://www.scribbr.com/methodology/action-research/>. Luettu 16.10.2023.

8 Luotio, Juha. 2023. Ketterä kehittäminen – ohjelmistokehityksen moderni menetelmä. Verkkoaineisto. Innofactor. <https://blog.innofactor.com/fi/kettera-kehittaminen>. Luettu 16.10.2023.

9 West, Dave. Agile scrum roles and responsibilities. Verkkoaineisto. Atlassian. <https://www.atlassian.com/agile/scrum/roles>. Luettu 16.10.2023.

10 Bluetooth. 2023. Verkkoaineisto. Android Developers. <https://source.android.com/docs/core/connect/bluetooth>. Luettu 16.10.2023.

11 Townsend, Kevin. Cufi, Carles. Akiba. Davidson, Robert. 2014. Getting Started with Bluetooth Low Energy. Verkkoaineisto. O'Reilly Media.

<https://books.google.fi/books?hl=fi&lr=&id=24N7AwAAQ-BAJ&oi=fnd&pg=PP1&dq=bluetooth+low+energy+mobile+device+and+peripheral&ots=k_GXIv6euQ&sig=QfnMCUt7bNIEB1B4GNnJ2hMTXLo&redir_esc=y#v=onepage&q=bluetooth%20low%20energy%20mobile%20device%20and%20peripheral&f=false>. Luettu 16.10.2023.

12 Yi Ong, Chee. 2019. 4 Tips to make Android BLE work. Verkkoaineisto. PunchThrough. <https://punchthrough.com/android-ble-development-tips/>. Luettu 16.10.2023.

13 Admin. 2023. Bluetooth Low Energy basics: classic Bluetooth Vs. Bluetooth LE. Verkkoaineisto. HowToElectronics. <https://how2electronics.com/classic-bluetooth-vs-bluetooth-low-energy-comparison/>. Luettu 16.10.2023.

14 Meet Android Studio. 2023. Verkkoaineisto. Android Developers. <https://developer.android.com/studio/intro>. Luettu 16.10.2023.

15 Gradle User Manual. 2023. Verkkoaineisto. Gradle.org. <https://docs.gradle.org/current/userguide/userguide.html>. Luettu 26.10.2023.

16 Milu. 2020. Git explained: the basics. Verkkoaineisto. Dev.to. <https://dev.to/milu_franz/git-explained-the-basics-igc>. Luettu 16.10.2023.

17 Topchyi, Liubov. 2019. What is SourceTree and how to use it. Verkkoaineisto. Alpha Serve. <https://www.alphaservesp.com/blog/what-is-a-sourcetree-and-how-to-use-it/>. Viimeksi päivitetty 21.7.2022. Luettu 16.10.2023.

18 What is Java Technology and why do I need it? 2023. Verkkoaineisto. Java.com. <https://www.java.com/en/download/help/whatis_java.html>. Luettu 16.10.2023.

19 What is Java? Verkkoaineisto. Amazon aws. <https://aws.amazon.com/what-is/java/>. Luettu 16.10.2023.

20 What is Jira? Verkkoaineisto. ProductPlan. <https://www.productplan.com/glossary/jira/>. Luettu 16.10.2023.

21 Blueooth Low Energy. 2023. Verkkoaineisto. Android Developers. <https://developer.android.com/guide/>. Luettu 16.10.2023.

22 ReactiveX. Verkkoaineisto. <https://reactivex.io/intro.html>. Luettu 16.10.2023.

23 Save data in a local database using Room. 2023. Verkkoaineisto. Android Developers. <https://developer.android.com/training/data-storage/room>. Luettu 16.10.2023.

24 About SQLite. Verkkoaineisto. SQLite. <https://www.sqlite.org/about.html>. Luettu 16.10.2023.

25 What is SQL (Structured Query Language)? Verkkoaineisto. Amazon aws. <https://aws.amazon.com/what-is/sql/>. Luettu 16.10.2023.

26 Gaussian Blur. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Gaussian_blur.> Luettu 4.11.2023.

27 Notifications overview. 2023. Verkkoaineisto. Android Developers. <https://developer.android.com/develop/ui/views/notifications>. Luettu 16.10.2023.

28 Smith, Steven W. 2003. Recursive Filters. Verkkoaineisto. ScienceDirect. <https://www.sciencedirect.com/topics/engineering/iir-filter>. Luettu 16.10.2023.