Älykkään työvaatteen prototyyppi

Case: Iittalan lasitehdas

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinnan korkeakoulukeskus, tietojenkäsittely

syksy 2019

Jonna Lahti

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely
Hämeenlinnan korkeakoulukeskus

Tekijä Jonna Lahti **Vuosi** 2019

Työn nimi Älykkään työvaatteen prototyyppi
Case: Iittalan lasitehdas

Työn ohjaaja Tommi Lahti

TIIVISTELMÄ

Tässä opinnäytetyössä käydään läpi älykkään työvaatteen prototyypin sekä sen kanssa toimivan mobiilisovelluksen projektipolku suunnittelusta toteutukseen. Opinnäytetyön aihe rajautui toimeksiantajan toiveiden, toimeksiantajan asiakasyrityksen tarpeen, aiheen ajankohtaisuuden sekä tekijän kiinnostuksen mukaan. Opinnäytetyön toimeksiantajana toimii Hämeen ammattikorkeakoulun Design Factory.

Opinnäytetyön tietoperusta käsittelee projektin toteutuksen olennaisimpia osia. Tietoperustassa tutustutaan työssä käytettyihin ohjelmointialustoihin ja -kieliin, tarkastellaan muutamaa eri tietokantaratkaisua sekä käydään läpi hyvän käyttöliittymäsuunnittelun periaatteita. Opinnäytetyön tavoitteena oli sekä rakentaa toimiva prototyyppi älykkäästä työvaatteesta että mobiilisovellus, joka ottaa vastaan prototyypin lähettämää dataa.

Työn lopputuotoksena syntyi tavoitteen mukainen prototyyppi älykkäästä työvaatteesta, joka mittaa työntekijän sykettä sekä lämpötilaa työntekijän välittömässä läheisyydessä. Myös mobiilisovelluksen toteutus oli onnistunut sekä tavoitteen mukainen. Opinnäytetyön lopputuotos menee tämän opinnäytetyön toimeksiantajan asiakasyritykselle Fiskars Oyj:n Iittalan lasitehtaalle.

Opinnäytetyön viimeisessä kappaleessa pohditaan syntyneen lopputuotoksen jatkokehitysmahdollisuuksia. Työntekijöiltä kerättyä dataa voisi esimerkiksi hyödyntää työhyvinvoinnin kehittämiseen sekä lopulta myös asiakasyrityksen tuloksen parantamiseen.

Avainsanat Arduino, Android, käyttöliittymäsuunnittelu, älyvaate, LilyPad

Sivut 41 sivua, joista liitteitä 3 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Program in Business Information Technology
University Centre, Hämeenlinna

Author Jonna Lahti **Year** 2019

Subject A Prototype of Smart Workwear
Case: Iittala Glass Factory

Supervisor Tommi Lahti

ABSTRACT

The purpose of this thesis was to go through the project path, from designing to execution, of making a functioning prototype of a smart work clothing and an Android native mobile app. The topic was chosen based on the wishes and needs of the client company and because the topic is a current issue and the author had an interest in it.

The theoretical basis covers the topics which are relevant for the development part of the thesis. The theoretical basis is based on existing literature and documentation of the topics discussed. The theoretical basis consists of the introduction of the integrated development environments and the programming languages used in the development part as well as the examination of few database options. The principles of good user interface design are also discussed in the theoretical basis of this thesis.

As a result, a working prototype and a mobile app were successfully engineered. The prototype measures the heartbeat of the employee as beats per minute and the temperature in close-proximity of the employee. The final product of this thesis is designed for the customer, Fiskars Oyj Iittala Glass Factory, of the client company.

The further development opportunities of the prototype and the mobile app are discussed in the final chapter of this thesis. One of the development possibilities could be the improvement of the well-being at work and thus the improvement of end customer company's profit.

Keywords Arduino, Android, user interface design, smart clothes, LilyPad

Pages 41 pages including appendices 3 pages

SISÄLLYS

1 JOHDANTO

Älyvaatteista on keskusteltu jo muutamia vuosia. Aluksi niiden tarkoitus oli palvella kuntosalilla kävijöitä ja fitness-henkisiä ihmisiä. Ajatus tästä lähti kuitenkin muuttumaan ja älyvaatteissa alettiin nähdä muitakin mahdollisuuksia kuin liikkuvien ihmisten askeleiden ja sykkeen seuranta. Älyvaate voisi olla ihmisen terveyden edistäjä ja sairauksien ennaltaehkäisijä. Älyvaate voisi olla turvavaruste työpaikalla tai jopa työkalu esimerkiksi sairaaloissa. Mahdollisuuksia on monia, kun vain osataan ajatella laatikon ulkopuolelle ja nähdä nykyisen teknologian yli.

Tässä opinnäytetyössä tarkastellaan älykkään työvaatekokonaisuuden prototyyppiä. Mitä kyseisen kokonaisuuden suunnittelussa tulee ottaa huomioon? Miten toteutus käytännössä tapahtuu? Miten rakennetaan toimiva prototyyppi hyödyntäen Arduinoa sekä Androidia? Tässä opinnäytetyössä älykäs työvaatekokonaisuus tarkoittaa älykkään työvaatteen prototyyppiä sekä mobiilisovellusta, johon kerätty data lähetetään tarkastelua varten. Aihe rajautui tekijän kiinnostuksen sekä toimeksiantajayrityksen asiakkaan tarpeiden mukaan. Aihe on ajankohtainen, sillä älyvaatteiden mahdollisuudet kasvavat teknologian mukana ja kysymys kuuluukin: ”Kuka ehtii ensin?”

Opinnäytetyö toteutetaan toiminnallisena opinnäytetyönä, jonka lopputuotoksena syntyy toimiva prototyyppi älykkäästä työvaatteesta sekä Android-pohjainen mobiilisovellus. Opinnäytetyön tilaajana toimii Hämeen ammattikorkeakoulun Design Factory. Hämeen Ammattikoreakoulussa Design Factory -toiminta käynnistyi juuri ja on tarkoitus, että toiminta saadaan kunnolla käyntiin vuoteen 2021 mennessä (HAMK, 2019). Design Factoryn tarkoituksena on tuoda opiskelijat ja työelämä yhteen ilman hierarkkisia esteitä. Siellä elinkeinoelämän kanssa yhdistyvät korkeakoulun vahvuudet, kuten kehittämisosaaminen. (Jussila, Silpola, Laurikainen & Salminen, 2019, s.13-14) Design Factoryn ajatuksena on, että opiskelijat pääsevät tekemään työelämälähtöisiä projekteja, joiden toimeksiantajina on oikeita yrityksiä. Projekteissa opiskelijat, opettajat sekä yrityksien edustajat työskentelevät yhdessä. (Kunnari, Jussila, Tuomela & Raitanen, 2019) Tämän opinnäytetyön lopputuotos menee Design Factoryn asiakasyritykselle Fiskars Oyj Iittalan lasitehtaalle.

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa toimiva prototyyppi älykkäästä työvaatteesta sekä sen kanssa toimiva mobiilisovellus. Työvaatteen on tarkoitus toimia mittalaitteena, joka mittaa työntekijän sykettä ja lämpötilaa työntekijän välittömässä läheisyydessä. Asiakasyritys tarvitsee lopputuotoksena syntyvän prototyypin kaltaisen älykkään työvaatteen siksi, että lämpötilaa työntekijän välittömässä läheisyydessä pystyttäisiin seuraamaan aiempaa tarkemmin. Asiakasyrityksellä on tällä hetkellä käytössä kiinteät mittalaitteet lämpötilan seurantaa varten.

Näissä kiinteissä laitteissa ongelmana on se, että ne mittaavat lämmön vain laitteen ympäriltä. Näin työpaikalle jää sellaisia pisteitä, joissa lämmön mittausta ei ole ja lämpötila saattaa nousta turvallisuusrajan yläpuolelle aiheuttaen mahdollisen työturvallisuusriskin. Kun lämpötila ylittää 28 celsiusastetta, tulee työntekijöiden pitää 10 minuutin tauko kerran tunnissa. Jos lämpötila nousee yli 33:n celsiusasteen tulee tauon olla 15 minuuttia. (Aluehallintovirasto, 2018) Tässä opinnäytetyössä turvallisuusrajana käytetään jäljempänä mainittua 33:a celsiusastetta.

2 ÄLYVAATTEET

Älyvaatteen käsite ei ole yksinkertainen. Työ, terveys ja turvallisuus -lehdessä Haavisto (2018) kirjoittaa, että jotkut ihmiset ajattelevat elektroniikan tekevän vaatteesta älykkään ja toiset taas saattavat ajatella, että pelkkä tekninen kangas tekee vaatteesta älyvaatteen. Kun puhutaan älyvaatetuksesta, on hyvä erottaa kaksi käsitettä toisistaan. Älyvaatteesta puhuttaessa kysymyksessä on vaate, johon on lisätty elektroniikkaa ja älytekstiili taas käsittää Gore-Texin kaltaisen teknisen materiaalin. (Haavisto, 2018) Älyvaatteiden lisäksi voidaan puhua myös puettavasta teknologiasta, jolloin käsite on laajempi ja sisältää muun muassa älykellot.

Älyvaatteita alettiin alun perin suunnittelemaan urheilu- ja fitness-käyttöön. Kohderyhmänä oli jo valmiiksi omasta terveydestään kiinnostuneet ihmiset, jotka voisivat seurata omia fysiologisia toimintojaan treenin aikana ja sen jälkeen. Nykyään tämä alkuperäinen ajatus on jäänyt taka-alalle ja älyvaatteiden tulevaisuus on normaaleissa arkivaatteissa ja kohderyhmänä on kaikki vaatteita käyttävät ihmiset. (Gokey, 2016) Gokey (2016) kirjoittaa, että alusvaatteet älyvaatteina toimisivat hyvin, koska sensorit olisivat siten mahdollisimman lähellä ihoa ja data olisi näin luotettavampaa. Esimerkiksi sensori, joka on hihansuun sijaan rintaliiveissä, mittaisi sykkeen luotettavammin. Samoin askeleita mittaava sensori olisi järkevämpi sijoittaa sukkaan kuin hihaan. Alusvaatteet älyvaatteina olisi myös helppo tapa saada ihmiset käyttämään niitä, koska useimmat ihmiset käyttävät alusvaatteita joka päivä.

Puettava teknologia, kuten älyvaatteet ja -kellot, ovat parhaimmillaan silloin kun ne eivät ole silmiinpistäviä. Gokey (2016) kertookin artikkelissaan, että älyvaatteiden tulee olla muodikkaita, näyttää tavallisilta vaatteilta ja kertoa enemmän kuin otettujen askelten määrän päästäkseen ihmisten suosioon.

2.1 Älyvaatteiden tulevaisuus

Yksi älyvaatteiden tulevaisuuden kuvista on ihmisen terveyden hoidossa. Esimerkiksi Skin Care, diabeetikoiden terveyden seurantaan erikoistunut yritys, on kehittänyt sukan, joka tarkkailee diabeetikon jalan lämpötilaa. Sukassa on kuusi sensoria, jotka on aseteltu niihin kohtiin, joihin kohdistuu eniten painetta ja missä loukkaantumisriski on suurentunut ihmisillä, joilla on diabetes. (Rucker, 2019)

Kiinassa, Dalian teknologisessa yliopistossa, on suunniteltu paita, joka tarkkailee ihmisen sydämen toimintoja. Paita on suunniteltu niin, että se havaitsee sydämen epänormaalin käyttäytymisen. Paita siis toimii samalla tavalla kuin ECG-laite (Electrocardiogram eli sydänsähkökäyrä, tunnettu myös EKG-laitteena). Tällaisen paidan avulla ihminen pystyisi tarkkailemaan oman sydämen terveyttä kotoa käsin. (Rucker, 2019)

Myös seuraavia ideoita on väläytelty älyvaatteiden saralla: paita, joka auttaa krooniseen selkäkipuun, paita, joka tarkkailee hengitystä ihmisillä, joilla on krooninen keuhkosairaus, sekä pehmeä, koko päivän pidettävä belly-band raskaana oleville, mikä tarkkailisi kohdun supistuksia sekä sikiön sydämen sykettä. (Brown, n.d.) Tulevaisuudessa älyvaatteiden potentiaalia tullaan hyödyntämään kuntosalien ulkopuolella. Fokus tulee olemaan tavallisissa ihmisissä ja siinä, mitä he tarvitsevat. Enää ei vain mitata eri asioita mittaamisen ilosta.

2.2 **Älyvaatteet työturvallisuudessa**

Älyvaatteiden kaavaillaan myös olevan yksi työturvallisuuden avaintekijöistä. Kysyntää näille on jo tänä päivänä, mutta monet älyvaateratkaisut ovat vielä liian kalliita tai niihin tarvittavia teknologiaratkaisuja ei ole vielä olemassa. Haasteina teknologian lisäämiselle työvaatteisiin eivät ole ainoastaan kustannukset vaan myös se, miten teknologia saadaan kestämään teollisuuspesut. Prototyyppejä, kuten tässäkin opinnäytetyössä, on kuitenkin jo kehitelty. (Kuivalahti, 2018)

Suomalainen startup yritys Myontec on kehittänyt älyvaatteita, jotka mittaavat työntekijöiden lihastoimintaa. Kyseisiä älyvaatteita on muun muassa käytetty Atrialla selvittämään lihanleikkaajien jännetuppitulehduksien syytä. Työntekijöiltä kerättyä dataa analysoitiin ja selvisi, että muutamilla muutoksilla, kuten lihanleikkuuveitsien vaihtamisella pidempiteräisiin veitsiin, voisi olla positiivinen vaikutus työntekijöiden työhyvinvointiin sekä lihasperäisiin sairauksiin liittyviin sairauslomiin. Vuoden kuluttua tarkasteltiin tehtyjen muutosten vaikutuksia ja selvisi, että sen jälkeen ei ollut ollut yhtäkään uutta jännetupentulehduksen diagnoosia ja kyseiseen sairauteen liittyvien sairaslomien määrä oli tippunut kaksi prosenttia. (Admin, 2019)

Älyvaatteiden käyttö työturvallisuuden parantamisessa nostaa myös huolenaiheita. Työntekijöillä on huoli omasta yksityisyydestään. Työntekijöiden keskuudessa herää muun muassa kysymyksiä siitä, mitä kaikkea tarkkaillaan ja mihin tietoa käytetään? Myös älyvaatteiden aiheuttamat keskeytykset työssä huolettavat. Jos esimerkiksi laite tärisee varoituksen merkiksi, voi työ keskeytyä sen seurauksena tai se voi jopa aiheuttaa vaaratilanteen. Yrityksiä huolettaa älyvaatteiden kustannukset sekä ROI (Return on Investment) eli sijoitetun pääoman tuotto. Älyvaatteiden tarkoitus on tietysti parantaa työturvallisuutta ja näin vähentää esimerkiksi tapaturmista johtuvia sairauslomia. Sijoitetun pääoman tuotto siis näkyy muun muassa vähentyneessä sairauslomien määrissä sekä vakuutusmaksuissa. (Ferguson, 2019) Eri toimialoilla sairaustapauksista johtuvat tulojen menetykset voivat olla hyvinkin eri suuruisia sekä tarkoittaa monen eri saatavan menetystä. Esimerkiksi tavallisessa vähittäistavarakaupassa, jossa osaamisen voidaan olettaa olevan samalla tasolla kaikilla työntekijöillä, sairaustapaus ei välttämättä aiheita muita kuluja kuin

sairasajanpalkan maksamista. Spesifimmillä aloilla osaaminen taas saattaa nojautua yhden työntekijän varaan, jolloin hänen poissaolonsa saattaa aiheuttaa sairasajanpalkan maksun lisäksi myös myyntisaatavien menetyksen.

3 OHJELMOINTIYMPÄRISTÖT JA LAITTEISTO

Tässä opinnäytetyössä hyödynnettiin kahta ohjelmointiympäristöä: Arduinoa sekä Android Studiota. Arduino-alustalla kirjoitettiin ohjelma, joka lukee ihmisen fysiologisia toimintoja LilyPad-mikrokontrollerin avulla. Android Studiolla taas toteutettiin mobiilisovellus, joka tallentaa LilyPad-mikrokontrollerilta saadut tiedot Firestore-tietokantaan ja näyttää kyseiset tiedot käyttäjälle mobiililaitteessa. Kummankin ohjelmointialustan käyttö ja ohjelmointikielet, Arduino sekä Java, tuli opetella itsenäisesti opinnäytetyön toteutusta varten.

Tutustuminen ohjelmistoalustoihin aloitettiin ilman pohjatietoja. Molemmat, niin Arduino kuin Android Studiokin, osoittautuivat helpoiksi käyttää sekä omaksua. Molempien käyttöön on myös olemassa erilaisia tutoriaaleja, joten apua löytyy tarvitsevalle.

3.1 Arduino ja LilyPad

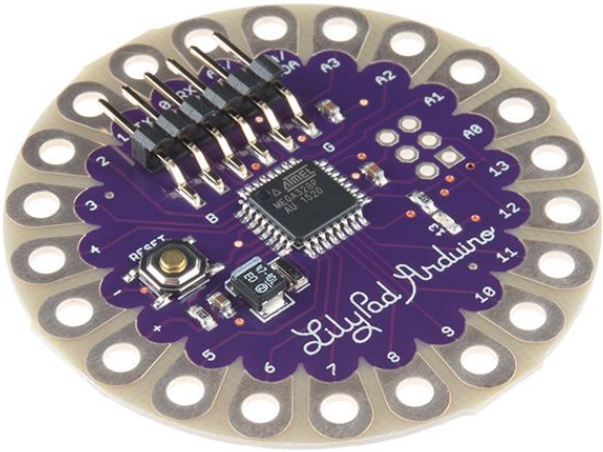
Arduino-ympäristö perustuu avoimeen ohjelmistoon ja laitteistoon. Se kehitettiin Ivrea Interaction Design instituutissa opiskelijoille, joilla ei ole aikaisempaa tietämystä teknologiasta tai ohjelmoinnista. Sen jälkeen se on kehittynyt monimutkaisemmaksi ja useampaan käyttötarkoitukseen sopivaksi prototyyppialustaksi. Sen kehittäjinä ovat toimineet niin opiskelijat, harrastajat, ohjelmoijat kuin ammattilaisetkin. Arduinoa on hyödynnetty tuhansissa eri tason projekteissa sen joustavuuden vuoksi. Se soveltuu helppokäyttöisyytensä vuoksi aloittelijalle, mutta taipuu myös kehittyneemmän ohjelmoijan tarpeisiin. (Arduino, n.d.a)

Arduino IDE:ssä (integrated development environment) käytettävä ohjelmointikieli pohjautuu C ja C++ -ohjelmointikieliin (Arduino, n.d.b). Arduino-ohjelmointikieli voidaan jakaa kolmeen pääosaan; funktiot, muuttujat ja rakenteet (Arduino, n.d.c). Ohjelmointikielessä on kaksi pääfunktiota void setup() ja void loop(). Setup -funktio suoritetaan vain kerran, kun laite käynnistetään. Sen tarkoitus on alustaa kaikki tarvittavat sensorit sekä lisäosat. Void loop -funktion sisään kirjoitetaan koko koodi, joka halutaan suoritettavan. (Ilmarinen, 2019)

Arduino-mikrokontrollerialustojen valmistajia on monia. Tässä työssä käytettiin DFRduinon valmistamaa alustaa (DFRduino UNO R3). Kyseinen alusta on vakain ja käytetyin Arduino-alusta, joka sopii yhteen Arduino IDE:n kanssa. (Robomaa, n.d.)

LilyPad Arduino on avoimen lähdekoodin laitteisto, jonka on suunnitellut ja kehittänyt Leah Buechley yhdessä SparkFun Electronicsin kanssa. LilyPad Arduino saa virran joko USB-yhteyden tai ulkoisen virtalähteen, kuten patterin, kautta. (Arduino, n.d.d) LilyPad kehitettiin älyvaatteisiin ja

muihin puettaviin projekteihin. Se on helppo ommella tekstiileihin ja se on myös pesunkestävä (Sparkfun, n.d.). Tähän projektiin valittiin LilyPad Arduino 328 Main Board (kuva 1), koska se on kevyt ja helppo kiinnittää tekstiileihin. Valinnan perusteena oli myös Bluetooth-integrointi. Opinnäytetyössä hyödynnettiin kolmea eri tyyppistä sensoria. Valosensoria, joka ommeltaisiin älyvaatteeseen indikoimaan työntekijälle lämpötilan olevan liian kuuma. Pulssisensori, joka mittaisi työntekijän pulssia työpäivän ajan sekä lämpötilasensoria, joka mittaisi lämpötilaa työntekijän välittömässä läheisyydessä. Sensorit itsessään ovat pieniä, esimerkiksi pulssisensori on halkaisijaltaan noin 1,5 senttimetriä, joten ne eivät haittaisi työntekoa.



Kuva 1. LilyPad Arduino 328 Main Board (Sparkfun, n.d.)

3.2 **Android-ohjelmointi**

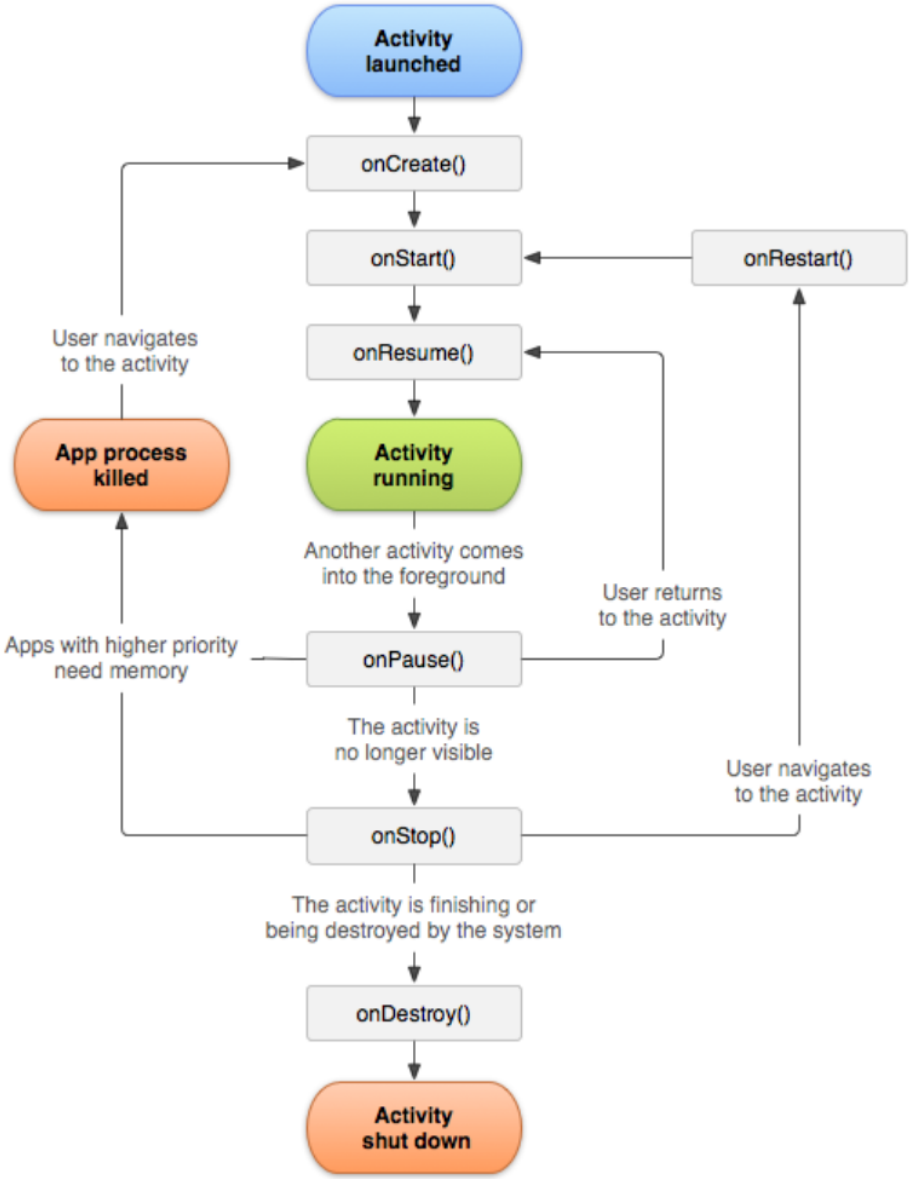
Virallisia Android-ohjelmointikieliä ovat Java, C++ sekä uusimpana tulokkaana Kotlin. Toukokuussa 2019 Google nosti Kotlinin Android-ohjelmointikielien kärkeen. Kotlinin suosimista perusteltiin ohjelmistotestauksen tarpeen sekä ylläpidettävän koodin määrän pienenemisellä. Kotlinia käytettäessä koodia ei tarvitse kirjoittaa yhtä paljon kuin Javassa. Google on kuitenkin ilmoittanut jatkavansa Java- sekä C++- kielten tukemista. (Kolehmainen, 2019).Tässä opinnäytetyössä tehty mobiilisovellus toteutettiin kuitenkin Javalla. Java valittiin siksi, että opinnäytetyön tekijällä oli siitä jo aikaisempaa kokemusta.

Android-sovellukset toteuttavat pienimmän oikeuden periaatetta (principle of least privilege). Tämä tarkoittaa sitä, että oletuksena sovelluksella on pääsy vain niihin komponentteihin, jotka se tarvitsee toimiakseen odotetulla tavalla. Tämä luo erittäin turvallisen ympäristön, jossa sovellus ei pääse käsiksi niihin osiin laitteessa, mihin sen ei tule päästä. On

olemassa kuitenkin tapoja, joilla sovellukselle voidaan antaa pääsy järjestelmään. Yksi tavoista on muun muassa luvan antaminen (request permission). (Google Developers, n.d.d) Tähän palataan hieman myöhemmin luvussa 6.

3.2.1 Aktiviteetit

Android-ohjelmoinnin keskiössä ovat aktiviteetit. Aktiviteetti on yksittäinen asia, jonka käyttäjä voi tehdä. Melkein kaikki aktiviteetit ovat vuorovaikutuksessa käyttäjän kanssa. Aktiviteetti-luokka Android-ohjelmoinnissa luo ikkunan, johon käyttöliittymän voi asettaa. Aktiviteetilla on elämänkaari, johon kuuluu erilaisia metodeja. Metodi on eräänlainen aliohjelma, joka suorittaa sille määrättyjä toimintoja. (Google Developers, n.d.a)



Kuva 2. Aktiviteetin elämänkaari (Google Developers, n.d.a)

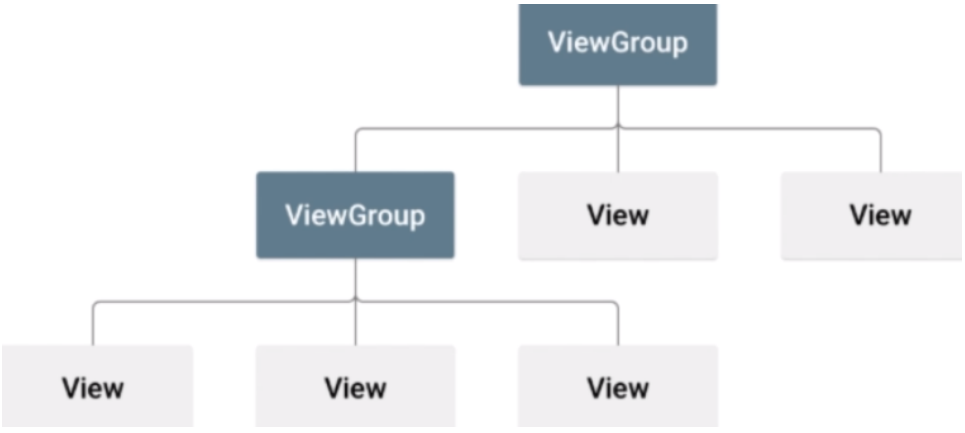
Kuten ylläolevasta kuvasta nähdään, aktiviteetin elämänkaaressa voi olla seitsemän erilaista metodia ja jokaisella niistä on oma tehtävänsä. Alla olevaan taulukkoon on eritelty jokaisen metodin tehtävä.

Taulukko 1. Aktiviteetin eri metodit (Google Developers, n.d.a)

Metodi	Tehtävä
onCreate()	Kutsutaan, kun aktiviteetti luodaan ensimmäisen kerran.
onRestart()	Tämä kutsutaan, kun aktiviteetti on pysäytetty. Kutsutaan, ennen kuin aktiviteetti aloitetaan uudelleen.
onStart()	Kutsutaan, kun aktiviteetti on näkyvä käyttäjälle.
onResume()	Kutsutaan, kun aktiviteetti on vuorovaikutuksessa käyttäjän kanssa.
onPause()	Kutsutaan, kun aktiviteetti ei ole enää näkyvä käyttäjälle tai kun aktiviteetti on taustalla.
onStop()	Kutsutaan, kun aktiviteetti ei ole enää näkyvä käyttäjälle, esimerkiksi, kun jokin muu aktiviteetti käynnistetään tai aktiviteetti ollaan lopettamassa kokonaan.
onDestroy()	Viimeinen metodi, joka kutsutaan ennen aktiviteetin lopullista tuhoamista.

3.2.2 Android View ja ViewGroup

Androidissa View-näkymä tarkoittaa mitä tahansa elementtiä, jonka kanssa voi olla vuorovaikutuksessa. View-luokka on yksi tärkeimmistä komponenteista Androidissa, koska melkein kaikki (näkymät, aktiviteetit ja niin edelleen) ellei jopa kaikki liittyvät jollakin tapaa kyseiseen luokkaan. ViewGroup taas voi sisältää muita näkymiä (view). Näitä näkymiä kutsutaan usein lapsiksi (children). Alla olevasta kuvasta nähdään paremmin näkymien hierarkia. (Dichone, 2019)



Kuva 3. View ja ViewGroup hierarkia (Dichone, 2019)

Kun käyttöliittymiä koodataan, tehdään usein näkymien hierarkioita. Kuvassa 3 nähdään näkymien hierarkia koodiesimerkissä. Koodissa ensimmäinen ViewGroup pitää sisällään aktiviteetin muut näkymät. Kuten esimerkistä nähdään, ViewGroup voi pitää sisällään myös muita ViewGroupeja.

Kuva 4. View ja ViewGroup koodiesimerkissä

4 TIETOKANTA

Tietokanta on systemaattinen kokoelma tiettyyn aihepiiriin liittyvää tietoa. Tietokannat tekevät tiedon käsittelemisestä helpompaa ja nopeampaa. (Guru99, n.d.) Tietokantoja on ollut olemassa siitä lähtien, kun ihmiset ovat alkaneet säilöä tietoa. Tietokoneellistetut tietokannat syntyivät 1960-luvulla, kun tietokoneista tuli halvempi vaihtoehto yksityiselle sektorille. Relaatiotietokannan kehitti E.F.Codd 1970-luvulla. (Quickbase, n.d.) Relaatiomallissa tietokannat koostuvat tietoalkioista ja näiden yhteyksistä eli relaatioista. Perusajatus tässä mallissa on se, että tallennettava tieto jaetaan käsitteisiin ja käsitteiden välisiin yhteyksiin. (Salminen, 2018)

SQL-kieli standardoitui kyselykielenä 1980-luvulla. Samaan aikaan relaatiotietokannasta tuli kaupallinen menestys, kun tietokoneiden myynti moninkertaistui. 1990-luvulla tietokantojen kasvu räjähti internetin mukana. Samoihin aikoihin avoimen lähdekoodin ratkaisuja jaettiin internetin välityksellä. Tietokannat ovat kasvaneet 1960-luvun jälkeen ja nykyään niitä onkin kaikkialla ja niitä käytetään parantamaan ihmisten jokapäiväistä elämää henkilökohtaisista pilvipalveluista aina yrityselämään saakka. Monet nykypäivän palvelut eivät olisi mahdollisia ilman tietokantoja. (Quickbase, n.d.)

4.1 SQLite

SQLite on avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä. SQLite ei tarvitse erillistä palvelinta, kuten muut SQL tietokannat, vaan se tallentaa tiedot suoraan tekstitiedostoon. SQLite on saavuttanut maineen erittäin luotettavana tietokantana, koska se testataan huolellisesti ennen jokaista julkaisua. SQLitellä on myös pitkä tuki, sillä kehittäjät ovat kertoneet tukevansa sitä vuoteen 2050 asti. (SQLite, n.d.a.) SQLite löytyy kaikista mobiilikäyttöjärjestelmistä sekä useimmista tietokoneista valmiina, joten erillisiä asennuksia sitä varten harvoin tarvitaan (SQLite, n.d.b.). SQLiteä suositaan mobiilisovelluksissa siksi, että se ei vaadi paljon muistia suorittamiseen. Lite-sana SQLitessä viittaakin keveyteen asennuksen, tietokannan hallinnan sekä tarvittavien resurssien suhteen. (SQLite tutoria, n.d.)

SQLitessä on erityisiä piirteitä, mitkä erottavat sen muista SQL-tietokannoista. SQLite käyttää dynaamisia tyyppejä tauluissa. Tämä tarkoittaa sitä, että sarakkeeseen voi säilöä minkä tahansa arvon huolimatta sen datatyypistä. SQLite pystyy myös luomaan tietokantoja muistiin, mikä tekee työskentelystä nopeaa. (SQLite tutorial, n.d.)

Androidissa on android.database.sqlite-niminen paketti, joka sisältää luokat, joiden avulla omia tietokantoja voi hallita. Luokka, jonka avulla voi

luoda sekä hallita tietokannan versioita on SQLiteOpenHelper. (Google Developers, n.d.e)

4.2 **Android ROOM**

Android ROOM on kirjasto, joka tarjoaa erillisen kerroksen (abstraction layer) SQLiten ylle ja sen ansiosta SQLiten käyttö on vakaampaa sekä helpompaa. Tämä kerros on yleistys joistakin tarkoista toiminnoista. (Google Developers, n.d.b) Android ROOM on siis järjestelmällisempi ja parempi tapa tallentaa tietoa hyödyntäen SQLite-tietokantaa.

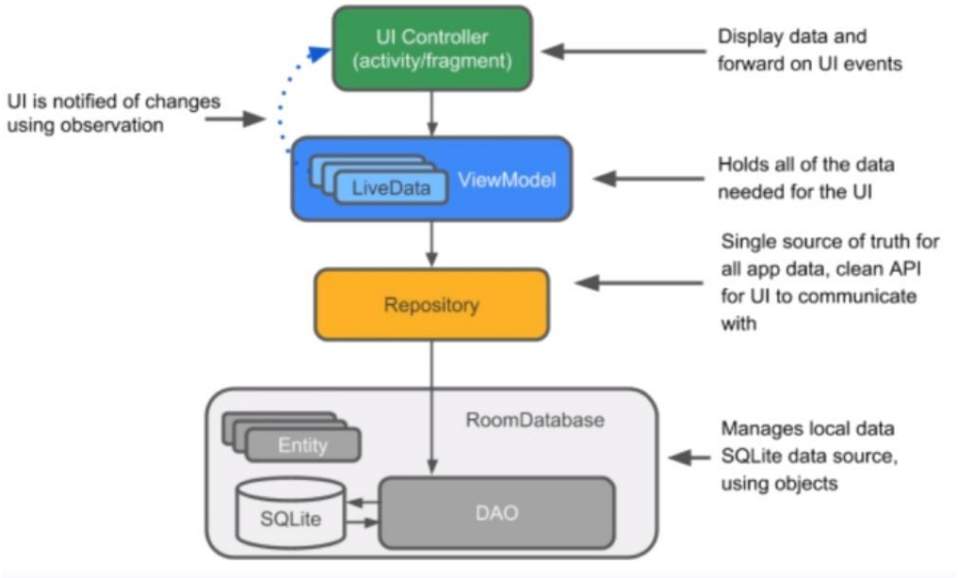
ROOM-tietokanta, jossa kaikki tapahtuu (tietokannan luonti, tiedonhaku eli kyselyt ja niin edelleen), koostuu kolmesta palasta. Entity-luokka edustaa tietokannan taulun riviä. Sen avulla tieto saadaan haettua tietokannasta riveittäin. SQLite on Androidin sisäänrakennettu tietokannan hallintajärjestelmä ja DAO eli Data Access Object, tekee kaikki CRUD-toiminnot. CRUD on lyhenne sanoista create (luo), read (lue), update (päivitä) ja delete (poista). (Dichone, 2019)

Repository on luokka, joka luodaan itse. Se ei ole pakollinen, mutta se on hyvä olla. Repository toimii lisäpuskurina ROOM:n ja käyttöliittymän välillä. Repository-luokkaa kutsutaan silloin, kun halutaan tehdä tiettyjä toimintoja ja luokka tekee halutut toiminnot ROOM-tietokannassa ja sen jälkeen välittää saadun tiedon eteenpäin. Repository toimii siis eräänlaisena viestintävälineenä. (Dichone, 2019)

ViewModel on vastuussa kaiken sen tiedon hausta, mitä tarvitaan käyttöliittymään, jotta tiedot saadaan esille varsinaiseen aktiviteettiin tai mihin tahansa muuhun näkymään, jossa tieto halutaan näyttää käyttäjälle. ViewModel tarkistaa, että kaikki varmasti toimii, kun tietoa aletaan siirtämään Repository-luokasta käyttöliittymään. Tässä tarkistuksessa on apuna eräänlainen entity-luokka, LiveData, joka tarkistaa, että käyttöliittymässä on kaikki kunnossa, kun tieto saapuu Repository-luokasta. Esimerkiksi tarvitseeko näkymä päivittää vai ei. LiveData siis tarkkailee koko ajan tapahtuvia muutoksia ja, kun jokin tieto muuttuu, se välittää sen eteenpäin ja kertoo, että näkymä käyttöliittymässä tulee päivittää. (Dichone, 2019)

Esimerkki ViewModelin toiminnasta voisi olla seuraavanlainen: Ajatellaan, että on olemassa tietovisa-sovellus, johon tallentuu korkein saatu pistemäärä. Kun sovellus käynnistyy, näkyy tuo korkein pistemäärä pelaajalle. Ajatellaan, että pelaaja pelaa peliä ja saakin korkeamman pistemäärän, mitä pelissä aiemmin on saatu ja sitten poistuu sovelluksesta. Seuraavan kerran, kun sovellus avataan, astuu ViewModel LiveDatan kanssa peliin. Kun sovellus avataan, taustalla tapahtuu toimintoja ViewModel-luokassa ennen kuin mitään näkyy käyttäjälle. ViewModel pitää aiempaa korkeinta saatua pistemäärä muistissa samalla, kun sitä verrataan viimeksi saatuun pistemäärään. Sen perusteella käyttöliittymä joko

päivitetään tai ei. Tässä tapauksessa käyttöliittymä tulee päivittää, koska viimeksi saatu pistemäärä onkin korkeampi kuin edellisellä kerralla, kun sovellus avattiin. Koska ViewModel toimii, saadaan käyttäjälle näytettyä oikea korkein pistemäärä. Kuvassa 4 nähdään paremmin Android ROOM -kokonaisuuden järjestys.



Kuva 5. Android ROOM (Dichone, 2019)

4.3 **Firestore**

Firestore on NoSQL, real-time tietokanta, joka skaalautuvuutensa ansiosta sopii tarkoitukseen kuin tarkoitukseen. Firestore toimii niin IOS, Android kuin web-sovelluksissakin. Firestore tarjoaa myös offline-tuen, joten sovellus, joka hyödyntää Firestorea toimii myös ilman internet yhteyttä. (Google Developers, n.d.c) Firestore on osa Googlen Firebasea. Jotta Firestore saadaan projektissa käyttöön, tulee Firebase yhdistää myös sovellusprojektiin.

Tapa, jolla tietoa tallennetaan Firestore-tietokantaan, on erilainen kuin tavallisessa SQL-tietokannassa. Firestore-tietokannan voidaan ajatella olevan kuvan 6 mukainen kansio. Firestoressa, kuten muissakin NoSQL-tietokannoissa, puhutaan taulun sijaan collectionista ja rivin sijaan dokumentista, johon tietoa tallennetaan.

Kuva 6. Firestoren rakenne (Google Developers, n.d.c)

Firestorea hyödyntämällä sovellukseen saadaan paljon muutakin kuin tietokantaratkaisu. Firebasen avulla muun muassa käyttäjien autentikointi kirjautumisvaiheessa on helppoa ja Firebasen avulla saadaan myös seurattua käyttäjämäärää sekä tarkasteltua sovelluksen muuta analytiikkaa. Firebase firestoressa on myös erillinen varastotila (storage), jota voi hyödyntää esimerkiksi kuvien tallennukseen.

5 KÄYTTÖLIITTYMÄ

Käyttöliittymällä tarkoitetaan laitteen, palvelun tai minkä tahansa tuotteen sitä osaa, jonka kanssa käyttäjä on vuorovaikutuksessa. Esimerkiksi jossakin ohjelmistossa käyttöliittymä on se osa, joka näkyy käyttäjälle kuvaruudulla. Yksinkertaisesti käyttöliittymä on linkki käyttäjän ja tuotteen välillä. (Pasanen, 2018)

Graafisella käyttöliittymällä tarkoitetaan ohjelmistojen käyttöliittymiä. Ne sisältävät esimerkiksi tekstilaatikoita, painikkeita sekä kuvia, jotka yhdessä muodostavat käyttökokemuksen. Kun puhutaan jonkin ohjelmiston käyttöliittymästä, tarkoitetaankin usein graafista käyttöliittymää. (Every interaction, 2018)

5.1 Käyttöliittymäsuunnittelu

Käyttöliittymäsuunnittelu tarkoittaa käyttöliittymän suunnittelua laitteille ja ohjelmistoille, missä keskitytään erityisesti käytettävyyteen ja tehokkuuteen. Usein käyttöliittymät suunnitellaankin tehokkaiksi ja käytettävyydeltään hyväksi. Joskus myös käyttöliittymät suunnitellaan tahallisesti epäeettisiksi. Tällä tarkoitetaan esimerkiksi sitä, että sovelluksen käyttäjätilin poistaminen tehdään tahallaan hankalaksi. (Pasanen, 2018) Hyvä käyttäjäkokemus on tiiviisti sidoksissa hyvin tehtyyn käyttöliittymäsuunnitteluun.

Käytettävyys on yksinkertaisesti ihmisen ja laitteen, esimerkiksi mobiilisovelluksen, vuorovaikutusta. Käytettävyydellä tarkoitetaan myös sitä, miten kyseinen vuorovaikutus saataisiin tehokkaammaksi. Käytettävyyttä voidaan mitata kuuden eri määreen avulla. Kolme näistä on ISO 9241-11 standardin mukaisia (vaikuttavuus, tehokkuus ja tyytyväisyys). Jakob Nielsen laajensi ISO-standardin määritelmää kolmella muulla kriteerillä (opittavuus, muistettavuus ja virheiden vähyys). Vaikuttavuutta tarkasteltaessa tarkastellaan sitä, miten täydellisesti käyttäjä saavuttaa sen, mitä halusi tehdä. Tehokkuudella tarkoitetaan esimerkiksi sitä, miten nopeasti käyttäjä saa tehtyä haluamansa toimen. Tyytyväisyydellä tarkoitetaan lyhyesti sitä, kuinka tyytyväinen käyttäjä on käyttämänsä laitteen, palvelun tai tuotteen käyttöön. Nielsenin kriteereillä opittavuudella ja muistettavuudella tarkoitetaan sitä, kuinka nopeasti ja helposti käyttäjä oppii laitteen/järjestelmän käytön ja logiikan sekä sitä, kuinka helppo käyttäjän on muistaa, miten laitetta käytetään sen jälkeen, kun on jo aiemmin oppinut sen. Virheiden vähyydellä tarkoitetaan sitä, kuinka virheettömästi käyttäjä saavuttaa tavoitteensa käyttäessään laitetta/järjestelmää. (Saksa, 2018)

5.2 **Hyvä käyttöliittymä**

Hyvän käyttöliittymän tulisi olla intuitiivinen, tehokas sekä käyttäjäystävällinen. Tällä tarkoitetaan sitä, ettei sen tulisi vaatia käyttäjältä käytön opiskelua, sen ei tulisi luoda turhaa työtä käyttäjälle ja sen käytön tulisi olla helppoa ja mukavaa. (Every interaction, 2018) Blogipostauksessaan Pasanen (2018) kirjoittaa 10 perussääntöä hyvälle käyttöliittymälle, jotka ovat tunne käyttäjäsi, kiinnitä huomiota malleihin, pidä homma yhtenäisenä, hyödynnä visuaalista hierarkiaa, anna palautetta, ole anteeksiantava, voimaannuta käyttäjäsi, puhu käyttäjien kieltä, pidä homma simppelinä sekä siirry eteenpäin.

Hyvä käyttöliittymä on suunniteltu niin, että se on saumaton sekoitus visuaalista, interaktiivista sekä informatiivista suunnittelua. Visuaalisella suunnittelulla tarkoitetaan sitä, että käyttöliittymä on näyttävä ja visuaalisesta kaunis, niin ettei se kuitenkaan häiritse käyttöä tai tarpeellisia toimintoja. Interaktiivisella suunnittelulla tarkoitetaan puolestaan sitä, että tarkastellaan, miten käyttäjä on vuorovaikutuksessa teknologian kanssa. Hyvä käyttöliittymä ei ainoastaan ennakoi käyttäjän toimia, vaan korjaa myös mahdollisia ongelmia hyvissä ajoin. Informatiivinen suunnittelu on sitä, että käyttäjää ohjeistetaan ja autetaan eri toimintojen suorittamisessa. Tällä tarkoitetaan esimerkiksi lomakkeissa olevia vihjeitä siitä, mitä käyttäjän odotetaan täyttävän mihinkin kohtaan. (Berezhnoi, 2019)

6 TYÖVAATTEEN PROTOTYYPIN JA MOBIILISOVELLUKSEN TOTEUTUS

Opinnäytetyön käytännön osuutta ei työstetty lineaarisesti Arduino-ohjelmoinnista varsinaisen sovelluksen suunnitteluun, vaan työskentely tapahtui samanaikaisesti eri osa-alueilla siten kuin se oli mahdollista ja mielekästä.

Tiedonsiirto älyvaatteen ja mobiilisovelluksen välillä oli suuri kysymysmerkki. Vaihtoehtoina pohdittiin Bluetooth-yhteyttä sekä palvelinratkaisua. Bluetoothia hyödynnettäessä data siirtyisi älyvaatteesta suoraan mobiiliin ja sovelluksessa tapahtuisi datan tarkastelu ja datan tallennus tietokantaan. Palvelinratkaisussa älyvaatteen keräämä data lähetettäisiin palvelimelle ja sitä kautta tallennettaisiin tietokantaan, jonka jälkeen mobiilisovelluksessa data vain haettaisiin tietokannasta.

Bluetooth-ratkaisun ajateltiin olevan selkeä, koska ylimääräinen palvelinyhteys saataisiin suljettua pois ja älyvaatteen datan linkittäminen oikeaan käyttäjään olisi helppoa MAC-osoitteen avulla. Huonona puolena Bluetoothissa on se, että kantaman vuoksi puhelin tulisi olla työntekijällä koko ajan mukana ja mobiilisovelluksen tulisi tallentaa dataa koko ajan. Tämä taas veisi puhelimen akun virtaa koko työpäivän ajan enemmän kuin normaalisti, koska sovelluksen tulisi olla päällä tallentaakseen tiedot tietokantaan. Palvelinratkaisussa data tallentuisi tietokantaan, vaikka puhelin ei edes olisi mukana työpäivän aikana. Datan linkitys oikeaan käyttäjään ei olisi yhtä helppoa, kuin Bluetoothin kanssa ja toisaalta, kun data pyöritetään palvelimen kautta se lisää tietoturvariskiä.

6.1 Arduino ja LilyPad

Älykkään työvaatteen prototyyppi koodattiin Arduinolla. Laitteistona hyödynnettiin DFRduino-alustaa, LilyPad Arduino 328 Main Board -mikrokontrollerialustaa, Bluetooth RN-42 Silver -moduulia, Pulse sensor -pulssisensoria ja LilyPad Temperature -sensoria, sekä DFRduinon omaa valosensoria. Näiden lisäksi käytössä oli kahdenlaisia kaapeleita, jotta tarvittavat kytkennät saatiin tehtyä.

Koska erilaisia tutoriaaleja löytyi kyseisten sensorien käyttöön, ei pyörää lähdetty keksimään uudelleen. Näistä olemassa olevista dokumentaatioista etsittiin tähän työhön sopivat komponentit ja niistä yhdistämällä saatiin toimiva kokonaisuus prototyyppiä varten. Hyödynnetyt koodit ovat avoimia ja niitä saa käyttää halutulla tavalla sekä muokata tarpeeseen sopivaksi.

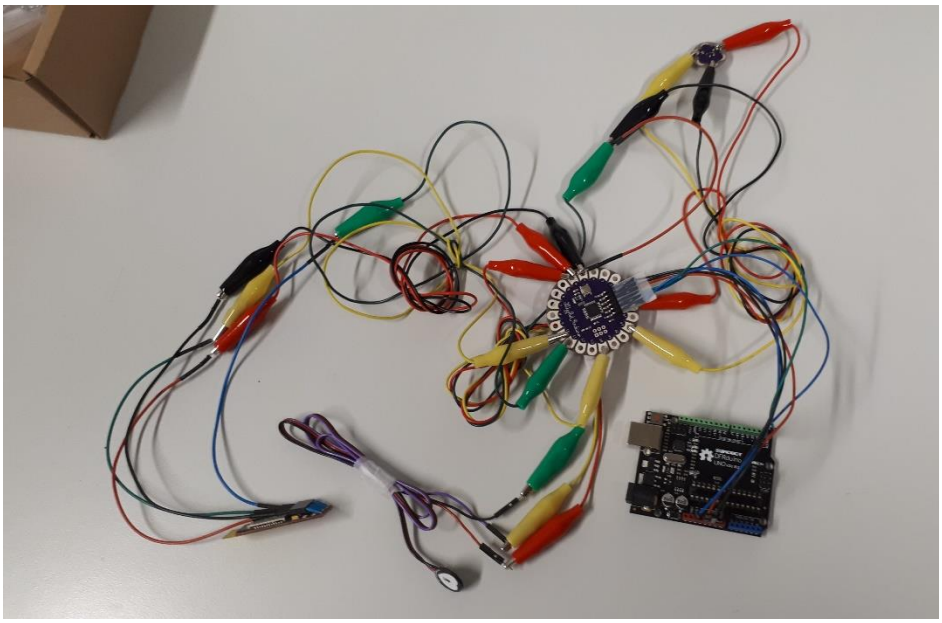
Prototyypin koodi kirjoitettiin toteuttamaan seuraavat toiminnot: mittaamaan lämpötilan ja muuntamaan sen celsiusasteiksi, mittaamaan pulssin lyönteinä minuuteissa (BPM), sytyttämään ledin, kun lämpötila ylittää 33,0 celsius-astetta, lähettämään mitatun datan viiden minuutin välein

Bluetooth-yhteyden avulla. Alla olevasta kuvasta nähdään Arduinon ohjelmakoodin palat, jotka toteuttavat lämpötilan sekä pulssin mittauksen. Osa koodista on kommentoitu ”pois”, koska se ei ole oleellinen lopullisen ohjelman toiminnalle, vaan sitä käytettiin vain ohjelmakoodin testaukseen.

```
// Calls function on our pulseSensor object that returns BPM as an "int".
 int myBPM = pulseSensor.getBeatsPerMinute();

// hold raw tempvalues
 long rawTemp;
// store voltage calculations
 float voltage;
// store celsius
 float celsius;

// read the raw 0-1023 value to variable and calculate voltage based on that value
 rawTemp = analogRead(tempSensorPin);
// Multply by maximum voltage and divide by maximum ADC value
 voltage = rawTemp * (5 / 1023.0);

// Calculate celsius temperature based on voltage
 celsius = (voltage - 0.5) * 100;

// For debugging
 // Serial.print(celsius);
```

Kuva 7. Lämpötilan ja pulssin mittaus

Prototyypissä ei voitu käyttää ulkoista virtalähdettä, koska saatavilla oli vain kolmen voltin virran tarjoava virtalähde ja toimiakseen Bluetooth-moduuli tarvitsisi vähintään 3,3 volttia. Virta otettiin siis suoraan DFRduino-alustasta. Kytkentä toteutettiin niin, että LilyPad Main Board kytkettiin DFRduinoon ja sensorit sekä Bluetooth-moduuli Main Boardiin. Alla olevasta kuvasta nähdään tehdyt kytkennät. Koska ulkoista virtalähdettä ei voitu käyttää, laite ei pysty toimimaan itsenäisenä laitteena.



Kuva 8. Prototyyppi kasattuna

Bluetooth-yhteyden avaamiseksi, Bluetooth-moduulin asetuksia tuli muuttaa. RN-42 Mate Silver toimii oletuksena 115200:n baudin nopeudella. Nopeus tuli muuttaa 9600:n baudin nopeuteen. Bluetooth-moduulin kytkennät tulee myös asettaa koodissa siten, että laite tietää, mitkä pinnit ovat käytössä. Alla olevassa kuvassa (kuva 9) näkyy ne palat koodista, joissa yllä mainitut alustukset tehdään.

```
const int bluetoothTx = 10;           // TX-0 of bluetooth mate to LilyPad pin 10
const int bluetoothRx = 11;           // RXD-1 of bluetooth mate to LilyPad pin 11

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);  // Creates bluetooth object

bluetooth.begin(115200);        // The Bluetooth Mate defaults to 115200bps
delay(320);                     // IMPORTANT DELAY! (Minimum ~276ms)
bluetooth.print("$");
bluetooth.print("$");
bluetooth.print("$");           // Enter command mode
delay(15);                      // IMPORTANT DELAY! (Minimum ~10ms)
bluetooth.println("U,9600,N");  // Temporarily Change the baudrate to 9600, no parity
bluetooth.begin(9600);          // Start bluetooth serial at 9600
```

Kuva 9. Bluetooth-asetukset

Jotta laite lähettää tiedot, tulee bluetooth-muuttujalle kertoa, mitkä arvot lähetetään ja missä järjestyksessä. Arduinossa kyseinen syntaksi on todella tarkka. Syntaksin oikein saamisen tekee vielä haastavammaksi se, että laite saattaa toimia muuten moitteetta (pois lukien datan lähetyksen), eikä virheilmoitusta tule lainkaan. Kuvassa 10 on se osa koodista, joka tekee datan lähetyksen.

```
bluetooth.print(celsius);
bluetooth.print(",");                    // Separator between temp and pulse
bluetooth.print(myBPM);
bluetooth.print(';');                    // End message to send
```

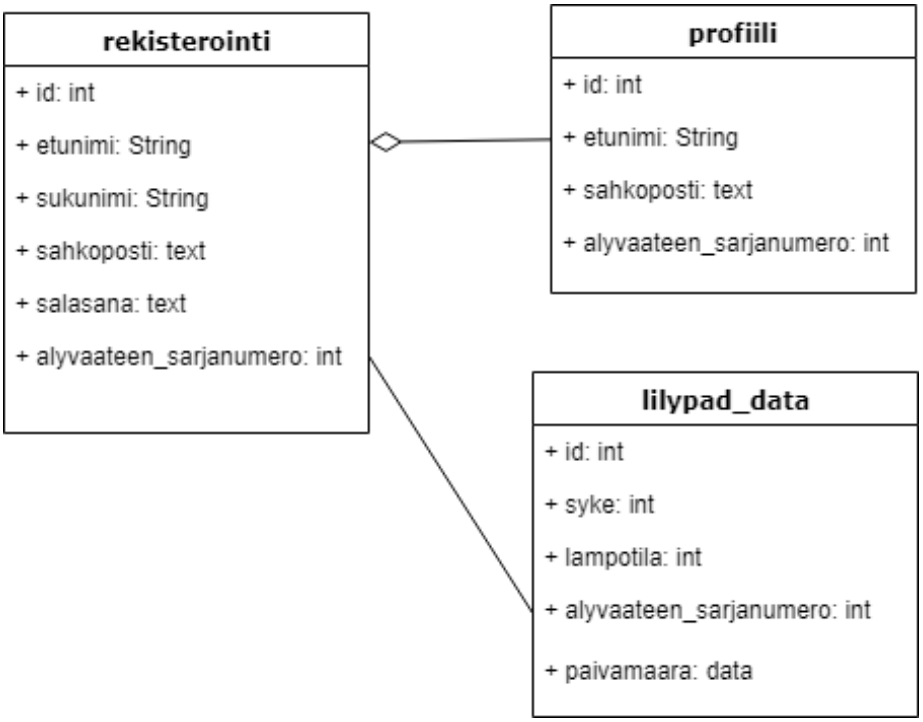
Kuva 10. Koodi datan lähetyksestä

Laitteen testausvaiheessa huomattiin myös se, että laitteen ollessa tarpeeksi kauan poispäältä Bluetooth-moduuli palauttaa asetuksensa ja näin ollen palaa takaisin aiemmin mainittuun 115200:n baudin nopeuteen. Tämä aiheuttaa sen, että ilman koodin uudelleen lähetystä mikrokontrollerille, Bluetooth-yhteyttä ei pysty muodostamaan. Ratkaisuna tähän voisi olla vaihtaa moduulin johonkin toiseen, esimerkiksi HC-05 moduulin, joka toimii oletuksena 9600:n baudin nopeudella. RN-42 moduulin kantama ei myöskään ole kovin pitkä, joten vaihto toiseen moduulin korjaisi myös tämän ongelman. Yllä mainitut ongelmat ovat kuitenkin varsinaisia ongelmia vasta, jos prototyypistä valmistettaisiin varsinainen älyvaate.

6.2 **Tietokantaratkaisu**

Tietokannan rakennetta suunniteltiin ennalta, jotta saataisiin kokonaiskuva siitä, millainen tietokanta olisi työn kannalta paras. Suunnitelma

tehtiin draw.io ohjelmalla. Alla olevassa kuvassa tehty suunnitelma tietokannan mahdollisesta rakenteesta.



Kuva 11. Tietokantasuunnitelma

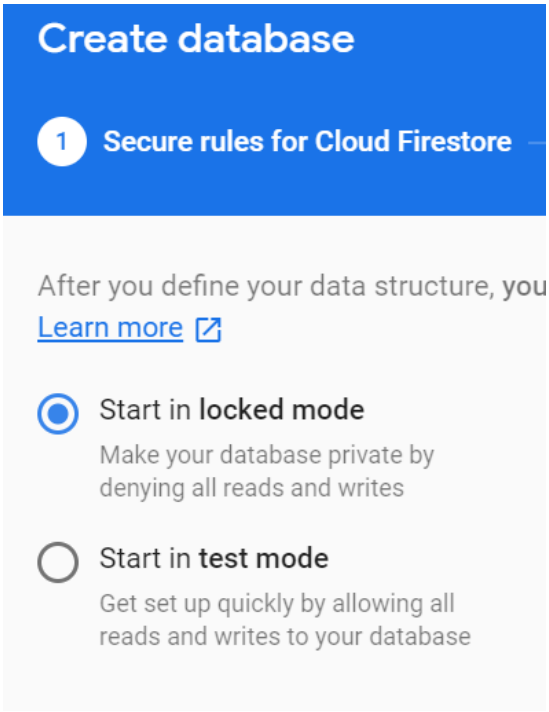
Tietokantaratkaisuksi valittiin NoSQL Firebase Firestore koska Firebase Firestoren avulla sovellukseen saatiin käyttäjien autentikointi sekä tietokantaratkaisu valmiina. Valintaan vaikutti myös kyselyiden suorittamisen tehokkuus ja nopeus sekä Firestoren mahdollistamat realtime-päivitykset. Jotta Firestore saadaan käyttöön, tulee Firebase yhdistetään Android Studiossa työkalujen kautta Tools -> Firebase -> Analytics -> Connect to Firebase. Autentikointi lisätään erikseen Tools -> Firebase -> Authentication. Firestore lisätään sovelluksen build.gradle -tiedostoon. Alla olevasta kuvasta nähdään Firebasen ja Firestoren implementointi kyseiseen tiedostoon.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.firebase:firebase-core:17.2.0'
    implementation 'com.google.firebase:firebase-firestore:21.1.1'
    implementation 'com.google.firebase:firebase-auth:19.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

Kuva 12. Firebase ja Firestore buld.gradle -tiedostossa

Varsinainen tietokanta alustetaan Firebase-konsolissa selaimen kautta. Tietokanta luodaan lukitussa tilassa, jottei kukaan muu pääse siihen käsiksi, vaikka omaisivat siihen suoran linkin. Testitilaa voi käyttää

testiprojekteissa, joihin pääsisi helposti muutkin käsiksi. Kuvassa 13 tietokannan luonnin asetukset.



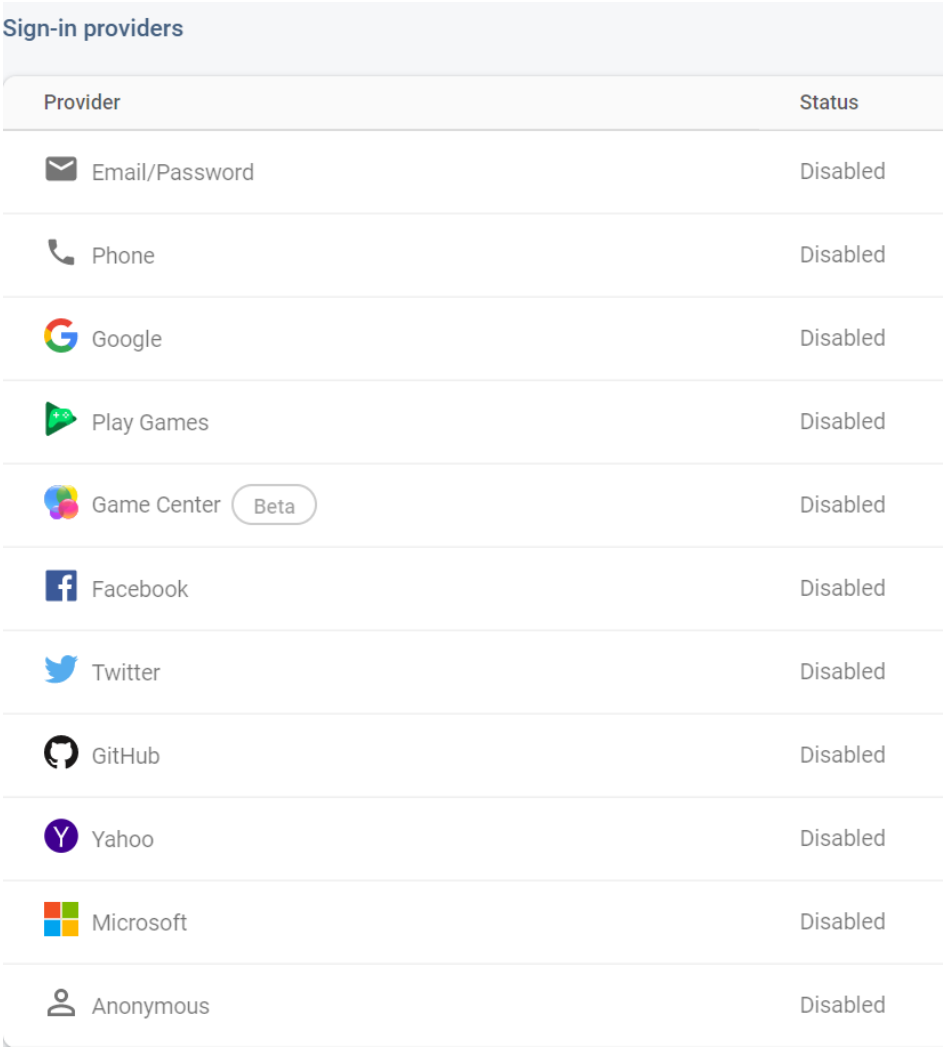
Kuva 13. Tietokannan luonti konsolissa

Kun tietokanta luodaan lukitussa tilassa, tulee tietokannan asetuksia muuttaa niin, että käyttäjät voivat käyttää tietokantaa sovelluksessa. Tietokannan säännöt tulee muuttaa siten, että autentikoidut käyttäjät voivat lukea tietoja tietokannasta sekä kirjoittaa uutta tietoa sinne. Alla olevasta kuvasta nähdään, minkälaiseksi tietokanta asetukset tulee muuttaa. Muutos tehdään selaimessa Firebase-konsolissa.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth.uid != null;
    }
  }
}
```

Kuva 14. Autentikoitujen käyttäjien salliminen tietokannan säännöissä

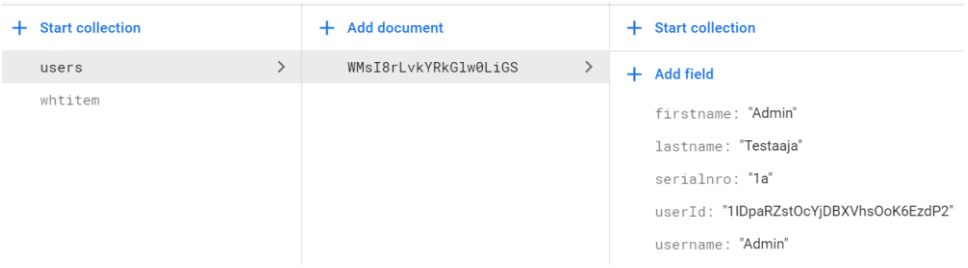
Käyttäjien tunnistamisen asetukset tulee myös määrittää Firebase-konsolissa. Konsolissa määritetään, mitä tunnistustapaa sovellus käyttää. Tässä työssä tehdyn sovelluksen autentikointitavaksi valittiin sähköpostitunnistus. Alla olevasta kuvasta nähdään, mitä eri tunnistustapoja Firebase tarjoaa.



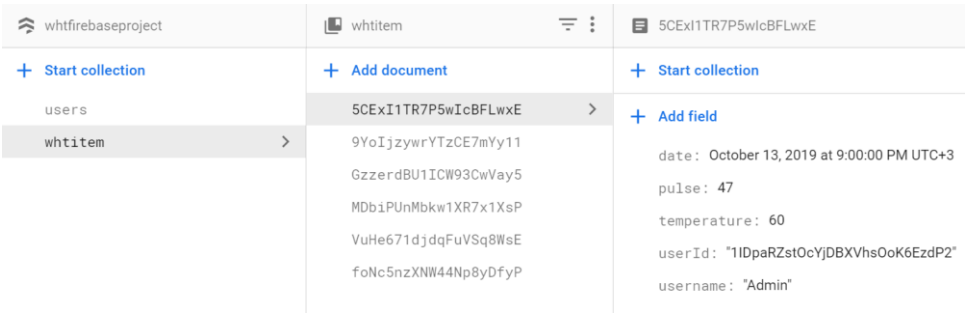
Kuva 15. Autentikointivaihtoehdot

Firebase-konsolissa luotiin yksi testikäyttäjä, Admin, jotta sovellusta voitiin testata sekä esitellä. Käyttäjä luotiin ensin autentikoituihin käyttäjiin, jonka jälkeen se lisättiin myös tietokantaan. Tätä varten tietokantaan luotiin collection ”users” ja sille ensimmäinen dokumentti, joka sisälsi tiedot testikäyttäjästä.

Lopullisen tietokannan rakenne erosi suunnitellusta. Tauluja eli collectioneita luotiin vain kaksi ”users” ja ”whtitem”, jotka sisälsivät sovelluksen vaatimat tiedot. Molempiin tauluihin luotiin testitietoja Firebase-konsolissa, jotta sovelluksen toiminnallisuuksia saatiin testattua ennen varsinaista dataa. Kuvassa 16 nähdään users-collectioniin lisätyt testitiedot ja kuvassa 17 whtite-collectioniin lisätyt testitiedot.



Kuva 16. users-collection



Kuva 17. whtitem-collection

6.3 **Käyttöliittymän suunnittelu**

Varsinaisen sovelluksen suunnittelu aloitettiin käyttöliittymän suunnittelulla. Suunnitelma toteutettiin Figma-ohjelmalla. Figma on käyttöliittymien suunnittelua varten tehty ohjelma, joka toimii selaimessa. Figmalla pystyy suunnittelemaan käyttöliittymän prototyypin, jonka toimintoja pystyy testaamaan suoraan selaimessa. (Bracey, 2018) Sovelluksen prototyyppiä voi kokeilla Figmasta löytyvillä eri puhelimien näytöillä.

Suunnitelmapohjaksi valittiin Android-frame ja prototyyppiä kokeiltiin Figman Google Nexux 5X puhelimella. Varsinaisen sovelluksen jokainen näkymä suunniteltiin pääpiirteissään Figmalla, jotta sovelluksen rakentaminen Android Studiolla olisi mahdollisimman sujuvaa. Suunnitelmaan valittiin yrityksen värit, mutta värimaailma katsotaan kohdalleen lopulliseen sovellukseen asiakkaan toiveiden mukaisesti. Suunnitelma sisälsi kymmenen eri näkymää (LIITE 1), mitä sovelluksessa voitaisiin tarvita.

Käyttöliittymän suunnittelussa pidettiin mielessä aiemmin esitellyt kymmenen perussääntöä. Alla olevaan taulukkoon on kerätty nämä säännöt sekä se, miten ne huomioitiin sovelluksen käyttöliittymän suunnittelussa.

Taulukko 2. Perussääntöjen hyödyntäminen suunnitelmassa

Sääntö	Miten se huomioitiin?
Tunne käyttäjäsi	Tiedossa sovelluksen käyttäjäryhmä
Kiinnitä huomiota malleihin	Selvitettiin millaisia ovat toimivat käyttöliittymät
Pidä homma yhtenäisenä	Painikkeet samoissa paikoissa
Hyödynnä visuaalista hierarkiaa	Käyttäjän tarvitsemat asiat ovat selkeästi esille
Anna palautetta	Toast-viestit, jos esimerkiksi jokin vaadittu kenttä on tyhjä
Ole anteeksiantava	Käyttäjälle kerrotaan, mitä hänen odotetaan tekevän
Voimaannuta käyttäjäsi	Esimerkiksi salasanan syöttöön on @-pikanäppäin
Puhu käyttäjien kieltä	Sovelluksessa esitetään asiat ymmärrettävästi. Ei jargonia.
Pidä homma simppelinä	Sovellus on yritetty suunnitella mahdollisimman helpoksi ja yksinkertaiseksi käyttäjän näkökulmasta
Siirry eteenpäin	Käyttöliittymää päivitetään ja käyttäjäryhmän palautteet otetaan huomioon

Sovelluksen käyttöliittymää suunniteltaessa heräsi myös ajatus varsinaisen älyvaatteen käyttöliittymästä. Tavallaan työvaate on tässä tapauksessa myös käyttöliittymä. Siinä on teknologiaa sekä asioita, jotka tulisi ottaa huomioon varsinaista vaatetta suunniteltaessa. Suunnittelussa tulisi esimerkiksi huomioida, että sensorit ovat kunnolla kiinnitetty, etteivät ne hankaa, paina tai häiritse muulla tavoin työntekoa. Sensoreiden tulisi myös mahdollisesti kestää rankempi teollisuuspesu tai vaihtoehtoisesti olla helposti irrotettavissa sellaisen ajaksi. Vaate itsessään tulisi olla miellyttävää materiaalia sekä kevyt käyttää. Parhaassa tapauksessa älyvaate ei eroaisi nykyisistä työvaatteista mitenkään tai olisi jopa mahdollisesti alusvaatteen tapainen vaatekappale, jotta sen käyttäminen olisi vaivatonta eikä vaatisi työntekijältä mitään ylimääräisiä toimia.

6.4 **Sovelluksen toteutus Android Studiolla**

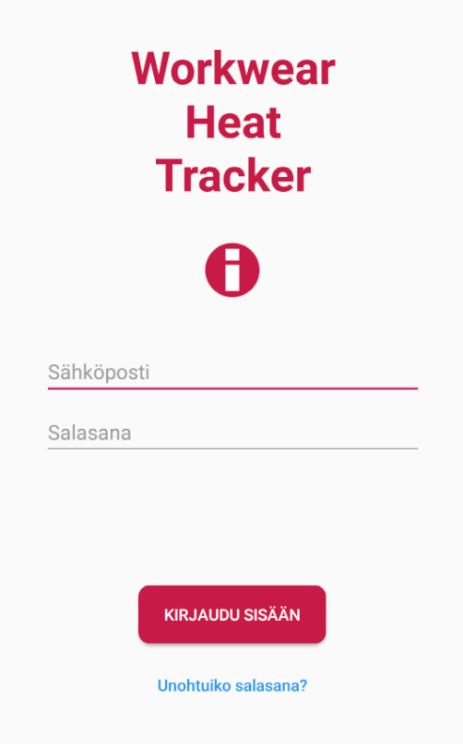
Tässä kappaleessa kuvataan tehtyä mobiilisovellusta. Joistakin kohdista nähdään myös toteutuksen taustalla oleva koodi. Kaikkea koodia ei kuitenkaan voitu tähän työhön liittää. Sovelluksesta otetuissa kuvakaappauksissa näkyvä data on testidataa, joka on syötetty käsin tietokantaan. Kyseiset luvut ovat testikäyttöön keksittyjä.

jotta sovellus voi kommunikoida toisen Bluetooth-laitteen kanssa, tässä tapauksessa älyvaatteen kanssa, tulee Bluetooth-yhteys alustaa Manifest-tiedostoon. Kuvassa 18 ensimmäinen lupa tarkoittaa Bluetoothin käyttöä ja toisella sallitaan myös Bluetooth-asetusten muokkaaminen. Käyttäjältä tulee myös kysyä lupa Bluetoothin käyttöön. Jos lupa saadaan, jatketaan koodin suorittamista eteenpäin. Jos lupaa ei saada, käyttäjä saa ilmoituksen, että lupa on annettava tarvittavien toimien suorittamista varten.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Kuva 18. Bluetooth-luvat Manifest-tiedostossa

Kaikkien aktiviteettien pohjaksi valittiin empty activity. Sovelluksen toteutuksessa hyödynnettiin tehtyä käyttöliittymäsuunnitelmaa. Suunnitelmasta poiketen kirjautuminen siirrettiin heti ensimmäiseen näkymään ja käyttäjätilin luonti jätettiin sovelluksesta kokonaan pois. Päätös poisjättämisestä tehtiin sovelluksen käyttötarkoituksen vuoksi. Sovellus tulee liittalan lasitehtaan työntekijöiden käyttöön, joten voidaan ajatella, että heille luodaan käyttäjätilit valmiiksi tietokantaan ja heille annetaan vain tunnukset sovellukseen. Hyvän käyttöliittymäsuunnittelun periaatteiden vuoksi kirjautuminen siirrettiin heti ensimmäiseen näkymään. Sovelluksen käytön kannalta on järkevämpää antaa käyttäjän suoraan syöttää kirjautumistiedot kuin laittaa käyttäjä ensin painamaan nappulaa, että pääse näkymään, jossa kirjautumistiedot lopulta syötetään. Kuvassa 19 nähdään lopullisen sovelluksen ensimmäinen näkymä. Sovelluksen alkunäytössä on myös ProgressBar, joka tulee näkyviin, kun kirjaudu sisään -nappulaa painetaan. ProgressBar lisättiin siksi, että käyttäjä tietää, että jotakin tapahtuu nappulan painamisen jälkeen.

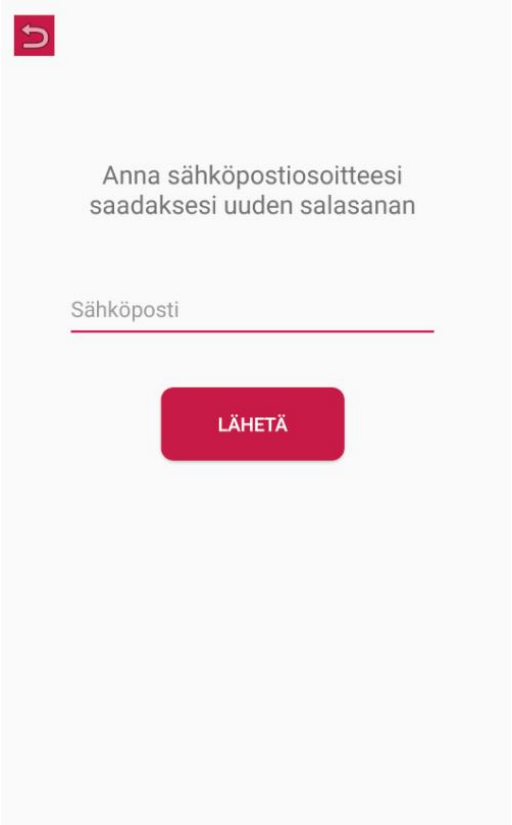


Kuva 19. Sovelluksen ensimmäinen näkymä

Jos käyttäjä on unohtanut salasanan, eikä pääse kirjautumaan sisään, voi käyttäjä klikata ”Unohtuiko salasanan?”, jolloin käyttäjä viedään kuvassa 21 nähtävään aktiviteettiin. Aktiviteetissa käyttäjä voi syöttää sähköpostiosoitteen, johon haluaa uuden salasanan lähetettävän. Kun käyttäjä klikkaa ”Lähetä”, Firebase lähettää sähköpostin salasanan palautusta varten. Firebase sisältää metodin (kuvassa 20) kyseistä toimintoa varten, joten toiminnon toteutus oli helppoa. Firebase konsolissa on myös mahdollista kustomoida kyseinen sähköposti.

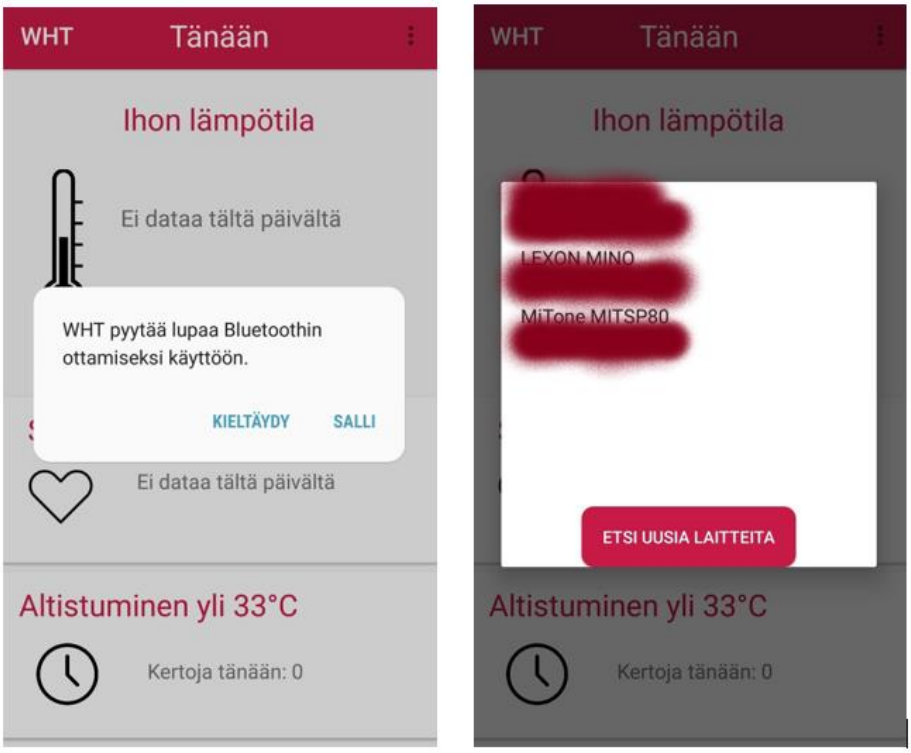
```
firebaseAuth.sendPasswordResetEmail(forgetPasswordEmail.getText().toString())
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                progressBar.setVisibility(View.INVISIBLE);
                Toast.makeText( context: ForgetPasswordActivity.this,
                        text: "Uusi salasana lähetettiin annettuun sähköpostiosoitteeseen",
                        Toast.LENGTH_SHORT).show();
            }else{
                progressBar.setVisibility(View.INVISIBLE);
                Toast.makeText( context: ForgetPasswordActivity.this,
                        text: "Salasanan lähetys epäonnistui",
                        Toast.LENGTH_SHORT).show();
            }
        });
```

Kuva 20. Firebasen metodi salasanan palautusta varten



Kuva 21. Näkymä, jossa käyttäjä voi pyytää uuden salasanan

Ensimmäisellä kirjautumiskerralla sovellus pyytää käyttäjältä lupaa sijainnin selvittämiseen, koska Bluetooth tarvitsee toimiakseen sijaintitietoja. Jos Bluetooth ei ole mobiililaitteessa jo päällä, sovellus pyytää lupaa (kuva 22, vasen) myös sen käyttöön. Jos käyttäjä sallii Bluetoothin käytön, esiin tulee popup-ikkuna (kuva 22, oikea), johon listautuu mobiilisovelluksen kanssa jo aiemmin pariutuneet laitteet. Jos haluttu laite ei ole tässä listauksessa, voi käyttäjä klikata painiketta ”etsi uusia laitteita”, jolloin sovellus skannaa löytyykö läheltä uusia laitteita. Listauksesta käyttäjä voi valita laitteen, jonka kanssa haluaa luoda Bluetooth-yhteyden. Kuvan listauksesta on peitetty osa yksityisyyden vuoksi.



Kuva 22. Lupa Bluetoothin käyttöön sekä listaus aiemmista laitepareista

Bluetooth-yhteyden hoitaa erillinen luokka BluetoothConnectionService. Luokkaan kirjoitettiin laitteiden pariutumisen hoitava sekä yhteyden ylläpitävä koodi. Pariutuminen sekä yhteyden ylläpito tapahtuvat erillisissä ketjuissa, ettei niiden toiminta hidasta muun sovelluksen toimintaa. Kun yhteys on luotu, sovellus alkaa kuuntelemaan saapuvia tavuja. Tavut muutetaan String-muotoon, jonka jälkeen saatu viesti lähetetään erillistä EventBus-kirjastoa hyödyntäen eteenpäin varsinaiseen aktiviteettiin. Kuvassa 23 on viestin eteenpäin lähetyksen hoitava osa koodista.

```
//sendMessage(readMessage);
EventBus.getDefault().post(new MessageEvent(readMessage));
```

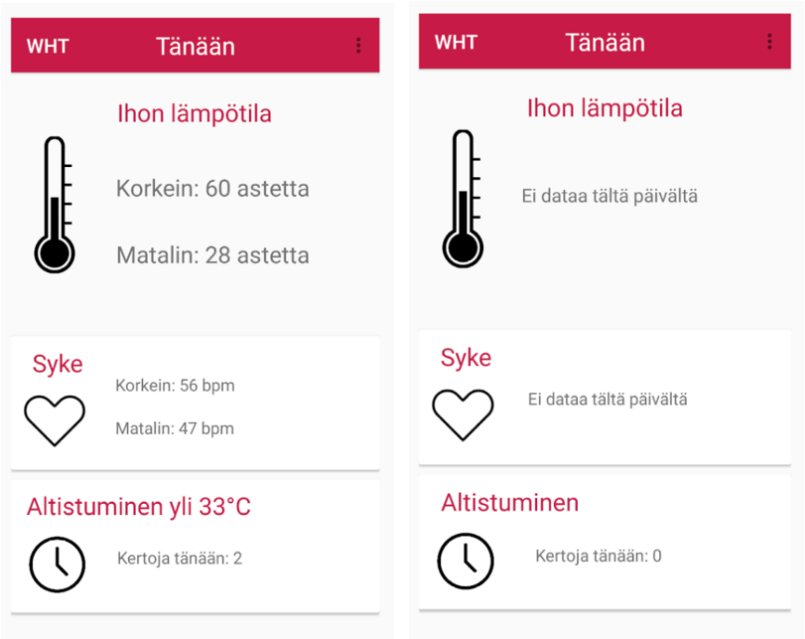
Kuva 23. Viestin lähetys aktiviteettiin

Vastaanotettu viesti käsitellään ja muotoillaan vasta aktiviteetin puolella. Koska lämpötila ja pulssi tulevat samassa String-muuttujassa, tulee ne ensin erottaa toisistaan. Tätä varten Arduinon puolella lähetettävään viestiin lisättiin erottajaksi pilkku. Erotuksen jälkeen muuttujat muunnetaan koodissa oikeaan muotoon (float ja int), että data voidaan tallentaa tietokantaan. Alla olevassa kuvassa (kuva 24) pala koodista, joka hoitaa yllä mainitun.

```
@Subscribe
public void custonEventReceived(MessageEvent event){
    String item = event.message;
    if (item != null) {
        String[] msgParts = item.split( regex: ",");
        String tempPart = msgParts[0];
        Log.d( tag: "kasittely",  msg: "custonEventReceived: täällä" + tempPart);

        float temperature = Float.parseFloat(tempPart);
        Log.d( tag: "kasittely",  msg: "custonEventReceived: " + temperature);
        String bpmPart = msgParts[1];
        int bpm = Integer.parseInt(removeLast(bpmPart));
        Log.d( tag: "kasittely",  msg: "custonEventReceived: " + temperature + bpm);
        saveToDatabase(temperature, bpm);
    }
}

public String removeLast(String bpm){
    if (bpm != null && bpm.length() > 0 && bpm.charAt(bpm.length()-1) == ';'){
        bpm = bpm.substring(0, bpm.length()-1);
    }
    return bpm;
}
```

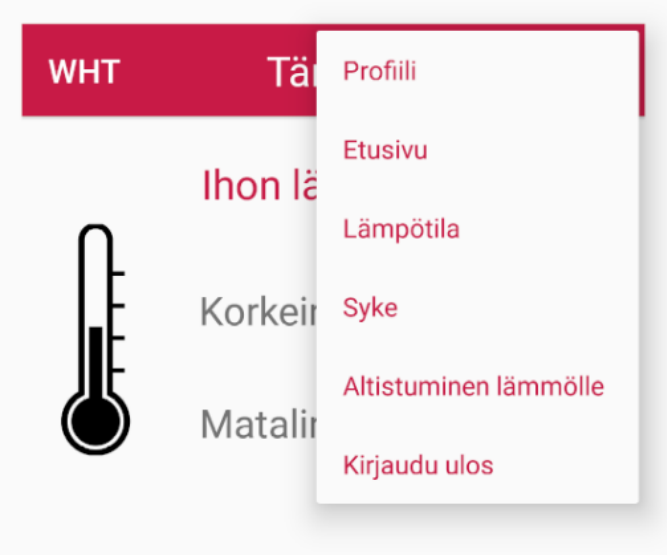
Kuva 24. Vastaanotetun viestin käsittely koodissa

Etusivulla käyttäjä näkee kuluvan päivän aikana mitatun korkeimman ja matalimman lämpötilan, korkeimman ja matalimman sykkeen sekä montako kertaa työntekijä on ollut yli 33:n celsiusasteen lämpötilassa (kuva 25, vasen). Jos dataa ei vielä siltä päivältä ole, se kerrotaan käyttäjälle etusivulla (kuva 25, oikea). Tietojen hakeminen tietokannasta toteutettiin niin, että se tapahtuu reaaliajassa. Näkymä siis päivittyy aina, kun tietokantaan tulee uusi kirjaus. Näkymä ei päivity, jos tietokannassa tapahtuisi jo olemassa olevan tiedon muokkaus tai poisto. Näitä toiminnallisuuksia ei otettu tässä vaiheessa huomioon, koska sovelluksessa käyttäjällä ei ole mahdollisuutta muokata tai poistaa olemassa olevia tietoja. Näitä toiminnallisuuksia ei pidetty oleellisina, joten ne jätettiin pois.



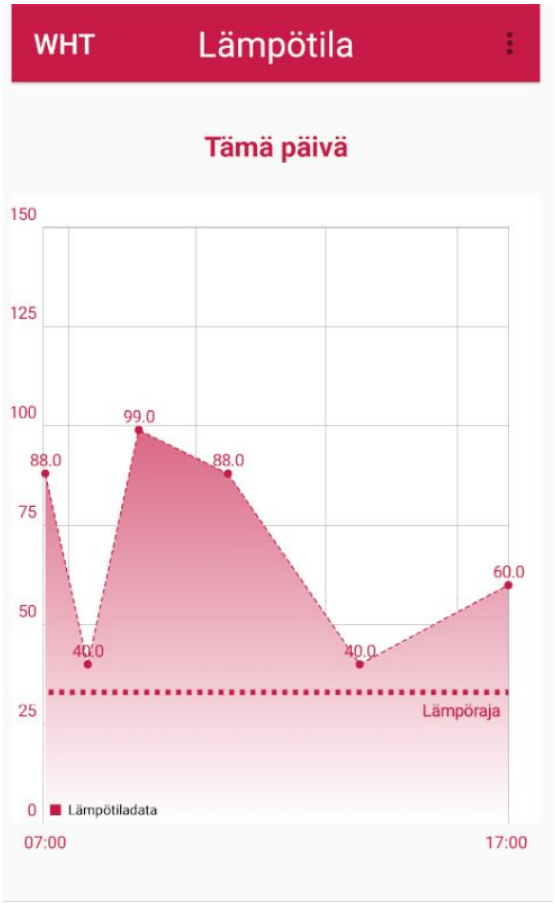
Kuva 25. Etusivu (vasen) ja etusivuilman kuluvan päivän dataa (oikea)

Jokainen näkymä sovelluksen sisällä sisältää toolbarin, jonka vasemmassa reunassa näkyy sovelluksen nimi, keskellä kyseisen näkymän nimi ja oikeassa reunassa menu (kuva 26), jonka avulla käyttäjä pääsee navigoimaan sovelluksen sisällä. Etusivulta pääsee navigoimaan myös klikkaamalla kuvakkeita (lämpömittari, syke- ja altistumiskohdat). Eri navigointitavat lisättiin sovellukseen sen vuoksi, että kaikki ihmiset eivät toimi samalla tavalla ja eri navigointityylit on käytettävyyden kannalta hyvä ottaa huomioon.



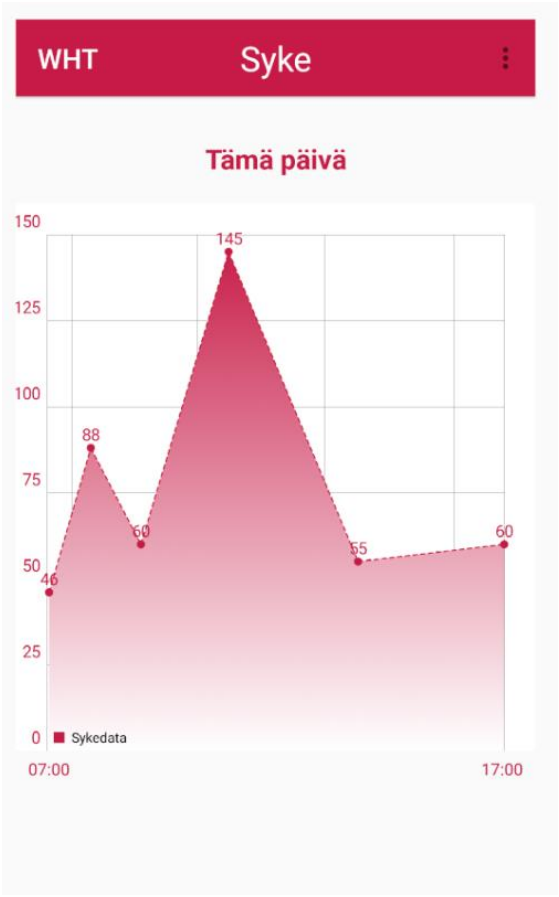
Kuva 26. Menu

Kuluvan päivän aikana mitatut lämpötilat päätettiin esittää viivakaaviossa (kuva 27), koska siitä näkee nopealla silmäyksellä, kuinka korkeita lämpötiloja päivän aikana on mitattu. X-akselille laitettiin kellonajat aamuseitsemästä iltaviiteen ja y-akselilla näkyy lämpötilat nollasta 150:een celsiusasteeseen. X-akselille valittu aikaväli valittiin, koska työpäivän ajateltiin olevan aamukahdeksasta neljään iltapäivällä ja jos työntekijä aloittaakin päivän hieman ennen kahdeksaa, saadaan myös ne lämpötilat viivakaavioon. Nämä kellonajat ovat vain esimerkkinä, koska oikeaa työpäivän pituutta ei ollut tiedossa, eikä se toteutuksen kannalta ollut edes olennaista. Kellonajat ovat helposti muutettavissa vastaamaan oikeaa työpäivää jälkikäteen. Lämpötilaraja asetettiin 33:n celsiusasteen kohdalle, koska kuten jo aiemmin mainittiin, kyseisen rajan jälkeen työntekijöiden tulee pitää 15:n minuutin mittainen tauko kerran tunnissa.



Kuva 27. Lämpötilat viivakaaviossa

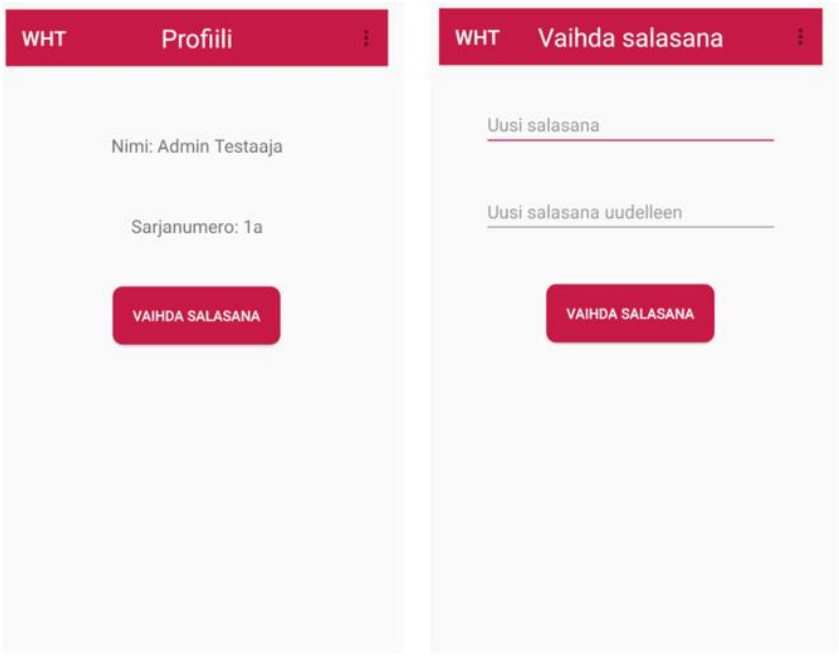
Myös sykedata esitetään kuluvalta päivältä viivakaaviossa (kuva 28). Sykekaavio on samankaltainen kuin edellä esitelty lämpötilakaavio. Erona on se, että rajaviivaa ei sykekaaviossa ole ja arvot ovat ilman desimaaleja. Sekä syke- että lämpötilakaavio piirretään samasta luokasta MyLineChart. Samaa luokkaa käytetään siksi, ettei samaa koodia tarvitse kirjoittaa montaa kertaa sovelluksen eri aktiviteeteissa. Viivakaavion piirtävää luokkaa kustomoitiin osittain sopimaan molempien aktiviteettien tarpeisiin. Luokkaa kutsuttaessa välitetään integer-muotoinen luku, jonka avulla tunnistetaan, kumpi aktiviteetti kyseistä luokkaa kutsuu. Tämän luvun avulla luokka osaa tehdä tarvittavat kustomoinnit, kuten esimerkiksi piirtää lämpötilanrajan.



Kuva 28. Sykekaavio

Jokaisella käyttäjällä on oma profiilinsa (kuva 29, vasen), joka sisältää käyttäjän nimen sekä älykkään työvaatteen sarjanumeron. Toteutusvaiheessa profiili ei olisi ollut välttämätön, mutta se tehtiin valmiiksi jatkokehitystä varten. Profiilista myös näkee, kuka sovellukseen on kirjautunut sekä oman älyvaatteensa sarjanumeron. Sarjanumero on vain esimerkkinä, eikä sen muotoa tässä työssä mietitty sen tarkemmin.

Profiilin kautta käyttäjä pääsee myös vaihtamaan sovelluksen salasanan. Vaihda salasana -painiketta klikkaamalla käyttäjä viedään aktiviteettiin (kuva 29, oikea), jossa käyttäjän tulee syöttää uusi salasana kaksi kertaa. Jos salasanan vaihto onnistuu, se ilmoitetaan käyttäjälle ja sen jälkeen siirrytään takaisin käyttäjän profiilin. Jos vaihto ei onnistu, siitä ilmoitetaan käyttäjälle.



Kuva 29. Profiili (vasen) ja vaihda salasana -näkymä (oikea)

Koska älykkään työvaatteen tarkoitus on seurata lämpötilaa ja sitä, kuinka usein lämpötila kohoaa yli turvarajan, on altistumiset yli 33:n celsiusasteen lämpötilalle koottu yhteen vieritettävään näkymään. Toteutuksessa käytettiin RecyclerView -näkymää (kuva 30). Näkymään koottu data on järjestetty siten, että uusin on ylimpänä.

Kuva 30. Listausnäkymä

7 YHTEENVETO

Työn tavoitteena oli tehdä toimiva prototyyppi älykkäästä työvaatteesta, joka mittaa lämpötilaa työntekijän välittömässä läheisyydessä sekä työntekijän pulssia ja sen jälkeen lähettää saadun datan tehtävään mobiilisovellukseen. Lopputuotoksena syntyi tavoitteen mukainen prototyyppi, jonka suorittaa tavoitteen mukaiset toiminnot sekä prototyypin kanssa toimiva mobiilisovellus. Tässä opinnäytetyössä rikottiin alakohtaisia rajoja aivan kuten Design Factoryn ajatuksena onkin. Opinnäytetyössä yhdistyivät tietojenkäsittelyn, liiketalouden sekä automaatiotekniikan piirteet. Opinnäytetyö oli ensimmäinen Design Factoryn toimeksiantama opinnäytetyö.

Jatkokehitystä ajatellen prototyypissä sekä mobiilisovelluksessa on potentiaalia. Työhyvinvointia voitaisiin edistää analysoimalla saatuja tuloksia ja selvittämällä työntekijöiden työkuormitusta sekä stressitasoja. Työhyvinvointi kuitenkin linkittyy myös yrityksen tulokseen, kuten jo aiemmin työssä mainittiin, ja parantamalla työntekijöiden työhyvinvointia, yritys myös panostaa omaan tulokseensa.

Älykkään työvaatteen rinnalle voitaisiin myös ottaa myös käyttöön stressitasoja sekä palautumista erikseen mittaava laite kuten esimerkiksi Moodmetric-älysormus. Moodmetric eroaa muista markkinoilla olevista puettavan teknologian ratkaisuista siinä, että sen ei ole tarkoitus mitata fyysistä kuntoa tai palautumista vaan ihmisen ihon sähköjohtavuuden muutosta. Tätä muutosta seuraamalla laite tarkkailee ihmisen virittymistä, mikä voi johtua erilaisista tunnereaktioista. (Moodmetric, n.d.) Yhdistämällä älykkään työvaatteen prototyyppi tällaiseen stressitasoja mittaavaan laitteeseen voitaisiin selvittää, löytyykö stressitasojen sekä prototyypin avulla kerätyn datan väliltä suhdetta. Voidaan pohtia altistaako korkea lämpötila suuremmalle sykevaihtelulle ja näin ollen nostaako se työntekijän stressitasoja. Suuret vaihtelut sykkeessä sekä korkea stressitaso voivat altistaa työntekijän erilaisille sairauksille ja näin myös aiheuttaa suurentuneen määrän sairauspoissaoloja. Tällaisilla kehittyneemmillä mittaustavoilla voisi olla suuri merkitys työhyvinvoinnin kehittämisessä.

LÄHTEET

Admin. (2019). Smart clothing to address pain points in occupational health. Haettu 15.9.2019 osoitteesta https://www.myontec.com/meet-the-health-transformer-using-smart-clothes-to-address-pain-points-in-occupational-health/

Aluehallintovirasto. (2018). Työsuojeluviranomainen muistuttaa: Tauotus ja jäähdytys auttavat jaksamaan helteellä. Haettu 21.10.2019 osoitteesta https://www.avi.fi/web/avi/-/tyosuojeluviranomainen-muistuttaa-tauotus-ja-jaahdytys-auttavat-jaksamaan-helteella

Arduino. (n.d.a). What is Arduino? Haettu 2.5.2019 osoitteesta https://www.arduino.cc/en/Guide/Introduction

Arduino. (n.d.b). Frequently asked questions. Haettu 2.5.2019 osoitteesta https://www.arduino.cc/en/Main/FAQ#toc13

Arduino. (n.d.c). Language referation. Haettu 2.5.2019 osoitteesta https://www.arduino.cc/reference/en/

Arduino. (n.d.d). LilyPad Arduino Main Board. Haettu 2.7.2019 osoitteesta https://www.arduino.cc/en/Main/ArduinoBoardLilyPad/

Berezhnoi, R. (2019). What is UI design and why is it important? Haettu 25.8.2019 osoitteesta https://f5-studio.com/articles/what-is-user-interface-design-and-why-is-it-important/

Bracey, K. (2018). What is Figma? Haettu 3.8. osoitteesta https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272

Brown, P. (n.d.). The future of health care may reside in your smart clothes. Haettu 25.8.2019 osoitteesta https://eu.mouser.com/applications/healthcare-may-reside-in-smart-clothing/

Dichone, P. (2019). The Comprehensive 2019 Android Development Masterclass, Udemy. Haettu 28.8.2019 osoitteesta https://www.udemy.com/android-development-java-android-studio-masterclass/learn/lecture/14562192#overview

Every interaction. (2018). User interaction. Haettu 24.8.2019 osoitteesta https://www.everyinteraction.com/definition/user-interface/

Ferguson, A. (2019). Ready to wear: Wearable technology could boost workplace safety, but concerns remain. *Safety+Health.* Haettu 15.9.2019 osoitteesta https://www.safetyandhealthmagazine.com/articles/18093-

ready-to-wear-wearable-technology-could-boost-workplace-safety-but-concerns-remain

Google Developers. (n.d.a). Activity. Haettu 23.7.2019 osoitteesta https://developer.android.com/reference/android/app/Activity

Google Developers. (n.d.b). ROOM Persistence Library. Haettu 28.8.2019 osoitteesta https://developer.android.com/topic/libraries/architecture/room

Google Developers. (n.d.c). Cloud Firestore. Haettu 14.9.2019 osoitteesta https://firebase.google.com/docs/firestore

Google Developers. (n.d.d). Application Fundamentals. Haettu 22.10.2019 osoitteesta https://developer.android.com/guide/components/fundamentals

Google Developers. (n.d.e). SQLiteOpenHelper. Haettu 22.7.2019 osoitteesta https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper

Gokey, M. (2016). Why smart clothes, not watches, are the future of smart wearables. *Digital trends.* Haettu 16.8. osoitteesta https://www.digitaltrends.com/wearables/smart-clothing-is-the-future-of-wearables/

Guru99. (n.d.). What is database? What is SQL? Haettu 22.7.2019 osoitteesta https://www.guru99.com/introduction-to-database-sl.html

Haavisto, P. (2018). Mikä on älykästä vaatteissa? *Työ, terveys ja turvallisuus.* Haettu 16.8.2019 osoitteesta https://www.tttlehti.fi/mika-on-alykasta-vaatteissa/

HAMK (2019). HAMK Vision and Strategy 2030. Häme University of Applied Sciences. Haettu 11.11.2019 osoitteesta: https://www.hamk.fi/wp-content/uploads/2019/02/HAMK_strategy2030.pdf

Ilmarinen, V. (2019). Arduino ohjeita (mm Bluetooth), Moodle. Hämeen Ammattikorkeakoulu. Haettu 2.5.2019 osoitteesta https://moodle.hamk.fi/course/view.php?id=20711

Jussila, J., Silpola V., Laurikainen, J., Salminen, J. (2019). Hämeen innovaatiotoiminnanuudet kujeet – HAMK Design Factory ja tuotekehityslaboratorio. E-julkaisussa *Kestävää osaamista – Biotalouden opettajat työelämälähtöisen oppimisen rakentajina*, 13-14. Haettu 11.11.2019 osoitteesta https://www.theseus.fi/bitstream/handle/10024/261336/HAMK_Ryymin_Vainio_%28toim%29.pdf?sequence=2&isAllowed=y

Kolehmainen, A. (2019). Google kehottaa käyttämään kotlinia – Android-java sai naulan arkkuunsa? *Tivi*. Haettu 23.7.2019 osoitteesta https://www.tivi.fi/uutiset/google-kehottaa-kayttamaan-kotlinia-android-java-sai-naulan-arkkuunsa/3818e838-3eae-4ca6-a879-e17cae6d89c4

Kuivalahti, L. (2018). Älyvaatelisää työturvallisuutta – tulevaisuuden työtakki varoittaa liian kovasta pakkasesta tai vaarallisista kaasuista. *Maaseudun tulevaisuus.* Haettu 26.8.2019 osoitteesta https://www.maaseuduntulevaisuus.fi/tiede-tekniikka/artikkeli-1.261164

Kunnari, I., Jussila, J., Tuomela, V., & Raitanen, J. (2019). Co-creation pedagogy from cSchool towards HAMK Design Factory. *HAMK Unlimited Journal.* 31.10.2019. Haettu 11.11.2019 osoitteesta https://unlimited.hamk.fi/ammatillinen-osaaminen-ja-opetus/co-creation-pedagogy

Moodmetric. (n.d.). Usein kysyttyä. Haettu 14.11.2019 osoitteesta https://www.moodmetric.com/fi/usein-kysyttya/

Pasanen, P. (2018). Mikä ihmeen käyttöliittymä? Blogipostaus 25.7.2018. Haettu 24.8.2019 osoitteesta https://fixui.fi/kaytettavyys/mika-ihmeen-kayttoliittymasuunnittelu/

Robomaa. (n.d.). Beginner Kit for Arduino (Best Starter Kit). Haettu 28.5.2019 osoitteesta http://robomaa.fi/beginner-kit-for-arduino-v30

Rucker, M. (2019). The future of condition-specific smart clothing. Haettu 25.8.2019 osoitteesta https://www.verywellhealth.com/the-future-of-condition-specific-smart-clothing-4125706

Saksa, T. (2018). Johdanto käyttöliittymäsuunnitteluun, Moodle. Hämeen Ammattikorkeakoulu. Haettu 23.8.2019 osoitteesta https://moodle.hamk.fi/course/view.php?id=19507&section=2

Salminen, L. (2018). Johdatus relaatiotietokantoihin. Haettu 22.7.2019 osoitteesta https://moodle.hamk.fi/mod/folder/view.php?id=635188

Sparkfun. (n.d.). LilyPad Arduino 328 Main Board. Haettu 2.7.2019 osoitteesta https://www.sparkfun.com/products/13342

SQLite. (n.d.a). About SQLite. Haettu 22.7.2019 osoitteesta https://sqlite.org/about.html

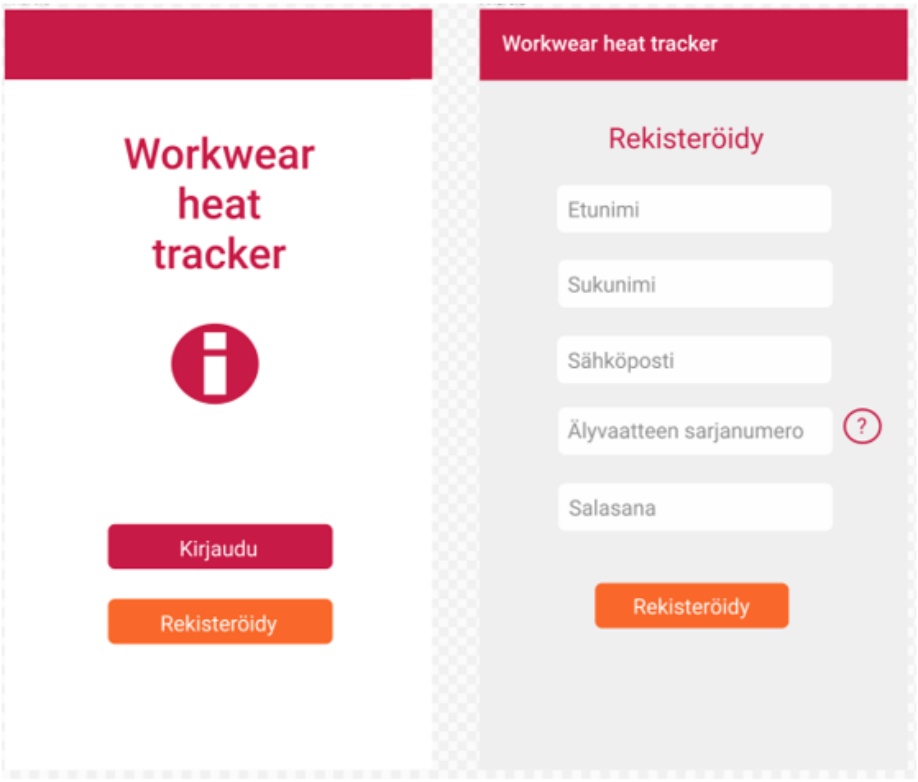
SQLite. (n.d.b). What is SQLite? Haettu 22.7.2019 osoitteesta https://sqlite.org/index.html

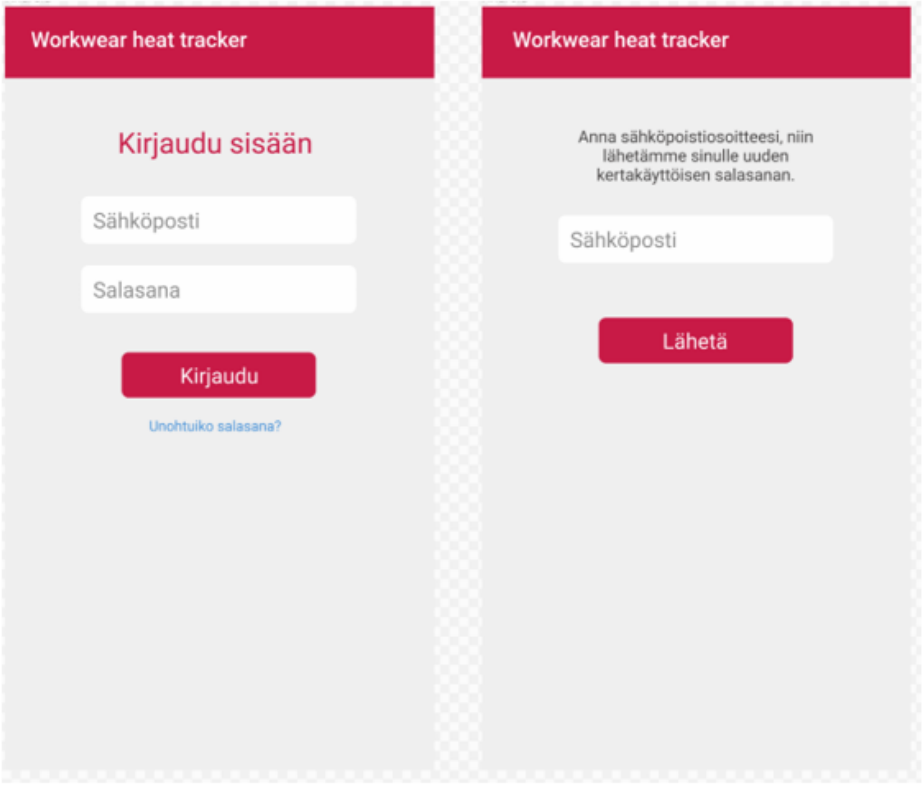
SQLite Tutorial. (n.d.). What is SQLite? Haettu 22.7.2019 osoitteesta www.sqlitetutorial.net/what-is-sqlite/

Quickbase. (n.d.). A Timeline of Database History. Haettu 22.7.2019 osoitteesta https://quickbase.com/articles/timeline-of-database-history

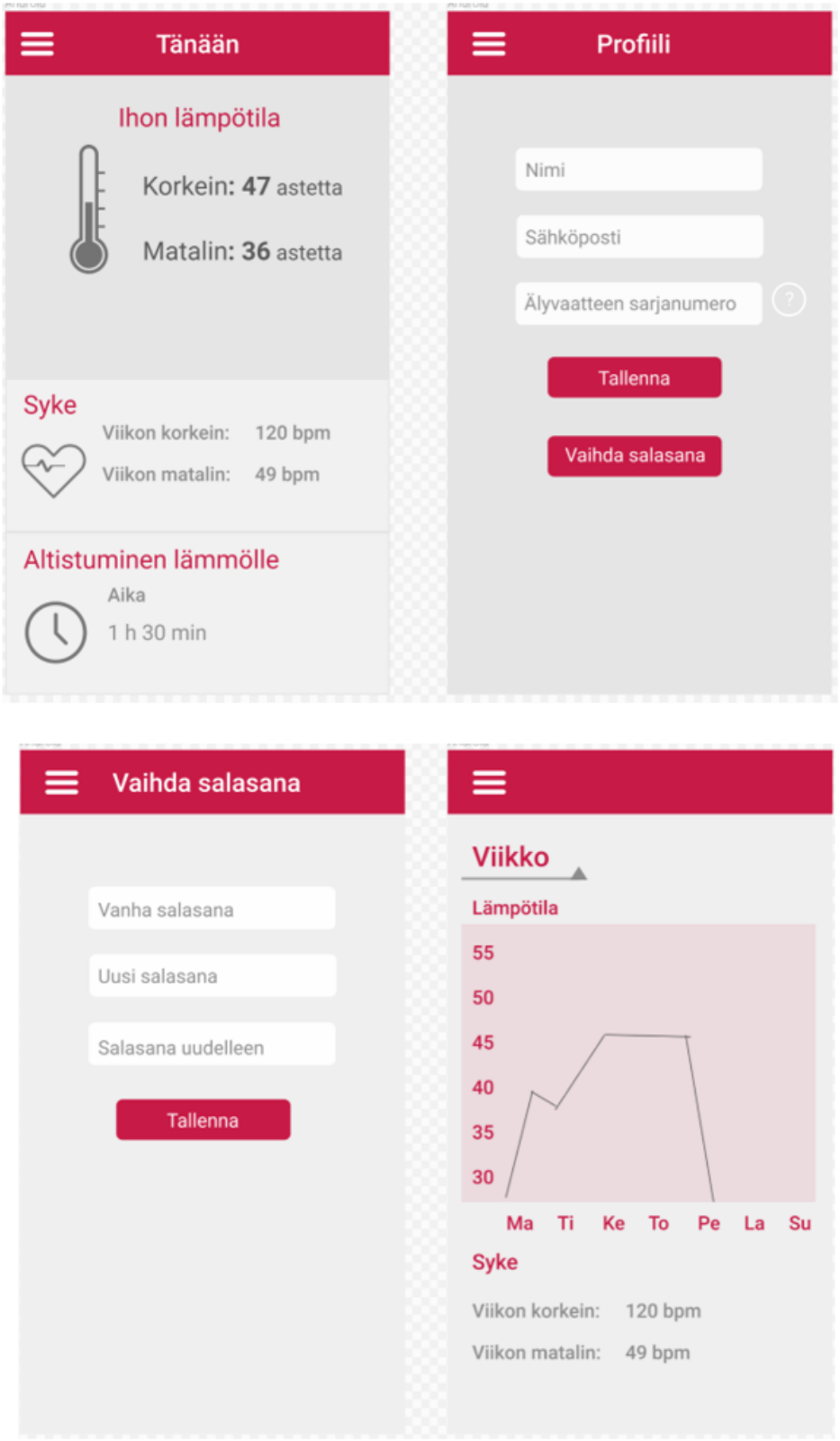
Liite 1/1

KÄYTTÖLIITTYMÄSUUNNITELMA





KÄYTTÖLIITTYMÄSUUNNITELMA



Liite 1/3

KÄYTTÖLIITTYMÄSUUNNITELMA