Atte Paitsola, Jani Raipala

NUCU-TUOTEHALLINTAJÄRJESTELMÄ

NUCU-TUOTEHALLINTAJÄRJESTELMÄ

Atte Paitsola, Jani Raipala
Opinnäytetyö
Kevät 2019
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijät: Atte Paitsola, Jani Raipala
Opinnäytetyön nimi suomeksi: Nucu-tuotehallintajärjestelmä
Opinnäytetyön nimi englanniksi: Nucu product management system
Työn ohjaaja: Eero Nousiainen
Työn valmistumislukukausi ja -vuosi: Kevät 2019
Sivumäärä: 50 + 3 liitettä

Nucu Oy on oululainen terveysteknologia-alan startup-yritys, joka valmistaa hyvinvointialustoja. Alustat auttavat vauvoja ja pieniä lapsia rauhoittumaan sekä nukkumaan paremmin. Alustat tuottavat kuulo- ja tuntoaistimuksia. Tärkeimpänä äänimaailmana käytetään äidin sykettä. Kuullessaan ja tuntiessaan äidin rauhoittavan sykkeen alustalla vauva rauhoittuu.

Opinnäytetyön tavoitteena oli luoda järjestelmä, jolla voidaan tehdasasentaa ja päivittää Nucu-alustoja. Järjestelmän täytyi sisältää myös tarvittavat tietokanta- ja verkkoratkaisut laitetietojen tallentamiseen ja tarkkailuun. Koska alustat eivät sisällä mitään verkko-ominaisuuksia, ne eivät pysty itsenäiseen kommunikaatioon ulkoisten laitteiden tai verkkojen kanssa. Järjestelmän suunnittelussa täytyi ottaa huomioon tulevaisuuden yritystoiminnan skaalautuvuuden tuomat tarpeet ja muutokset.

Työn tuloksena syntyi Android-sovelluksesta, web-sovelluksesta sekä palvelinympäristöstä koostuva järjestelmä, joka täytti kaikki projektin alussa sille määritetyt tehtävät ja tavoitteet. Järjestelmän kaikki osat ovat jatkokehityskelpoisia ja niitä tullaan kehittämään eteenpäin.

Asiasanat: ohjelmistokehitys, järjestelmäsuunnittelu, terveysteknologia, keskoset, koliikki, Node, Android, Angular

ABSTRACT

Oulu University of Applied Sciences
Information and communication technologies, Software development

---

Authors: Atte Paitsola, Jani Raipala
Title of thesis: Nucu product management system
Supervisor: Eero Nousiainen
Term and year when the thesis was submitted: Spring 2019
Pages: 50 + 3 appendices

Nucu Oy is an Oulu-based health technology start-up, manufacturing wellbeing nests for babies. The nests help babies and small children calm down and sleep. The nest's vibrating baseplate generates a soundscape that the child can also feel. The most important soundscape is the mother's heartbeat. Hearing the mother's heartbeat from the nest, the baby calms down.

The goal of the thesis was to create a system, that allows the factory-installation and updating of Nucu nests. The system had to contain the necessary database- and web-solutions to save and monitor information related to the devices. Because the nests do not contain any networking features, they are incapable of independent communication with external devices or networks. The design of the system had to take into account the changes and requirements brought by future scaling of the business.

The result of the thesis comprised an Android-application, a web-application and a server environment, that fulfilled all the requirements and goals set at the beginning of the project. All parts of the system are designed to be developed further and they will be improved upon in the future.

Keywords: software development, system design, health technology, Node, Android, Angular

ALKULAUSE

Haluan kiittää opinnäytetyön tilaajaa Juha Hannulaa haasteellisesta ja mielenkiintoisesta projektista. Suuri kiitos myös työparilleni Atte Paitsolalle, opinnäytetyön ohjaajalle Eero Nousiaiselle sekä kaikille muille, jotka ovat olleet tukemassa opinnäytetyön tekemistä.

Oulussa 16.4.2019

Jani Raipala

Kiitän opinnäytetyön tilaajaa Juha Hannulaa mielenkiintoisesta ja palkitsevasta aiheesta. Haluan kiittää myös pariani Jani Raipalaa, opinnäytetyön ohjaajaa Eero Nousiaista, sekä muita projektissa mukana olleita.

Oulussa 16.4.2019

Atte Paitsola

SISÄLLYS

SANASTO

API	Application programming interface. Ohjelmointirajapinta.
JavaScript	Pääasiassa web-ympäristössä käytetty ohjelmointikieli.
Kotlin	Vuonna 2011 julkaistu moderni ohjelmointikieli. Yksi viral-lisista Android-kehityskielistä.
MVC	Model-view-controller. Ohjelmistoarkkitehtuurimalli, jossa käyttöliittymä erotetaan ohjelmiston muista toiminnoista.
MySQL	Relaatiotietokantaohjelmisto.
Node	Avoimen lähdekoodin JavaScript-suoritusympäristö.
Nucu	Lasten hyvinvointialustan markkinointinimike. Valmistaja Nucu Oy.
Nucu Oy	Oululainen hyvinvointialan yritys.
OTG	On-the-Go. Kaapeli, jolla mobiililaite voidaan kytkeä USB-portin kautta toisiin USB-laitteisiin..
OYS	Oulun yliopistollinen sairaala
QR-koodi	Kaksiulotteinen kuviokoodi, johon voidaan sisällyttää in-formaatiota.
Sequelize	Objekti–relaatiodata-kartoituskehys Node-ympäristöön.
TypeScript	Microsoftin ylläpitämä avoimen lähdekoodin ohjelmointi-kieli.
UI	User Interface. Ohjelmiston käyttöliittymä.

1 JOHDANTO

Tämän opinnäytetyön tilaajana toimi Nucu Oy. Työn tavoitteena oli luoda järjestelmä, jolla yrityksen suunnittelemia Nucu-alustoja pystyttäisiin tehdasasentamaan sekä päivittämään helposti. Tämän lisäksi täytyi luoda tarvittavat tietokanta- ja verkkoratkaisut Nucu-alustojen perustietojen tallentamiseen ja tarkasteluun.

1.1 Nucu Oy

Nucu Oy on oululainen hyvinvointialan startup-yritys. Yritys on perustettu vuonna 2016 ja sen toimitusjohtajana toimii Juha Hannula (kuva 1). Yrityksen ensimmäinen markkinoille tullut tuote on vauvojen ja pienten lasten hyvinvointia parantava alusta Nucu. Nucu on suunniteltu auttamaan erityisesti keskosia ja koliikista kärsiviä vauvoja. Idea Nucu-alustaan ja myöhemmin yrityksen perustamiseen on lähtöisin Hannulan kokemuksista omien keskoslastensa kanssa. Idea on hioutunut prototyypistä valmiiksi tuotteeksi yhteistyössä Oulun yliopistollisen sairaalan TestLabin ja osasto 55:n kanssa.

KUVA 1. Juha Hannula (oikealla) ja Pirkko Metsävainio (vasemmalla) (1, s. 1)

1.2 Nucu-alusta

Nucu on vauvan pesämäinen hyvinvointialusta (kuva 2). Nucu koostuu pehmeästä pesäosasta, puisesta pohjalevystä sekä patjasta. Nucun erikoisuus on sen pohjalevy. Pohjalevy toistaa haluttua äänimaailmaa ja värisee äänen tahdissa luoden sekä kuulo- että tuntoaistimuksia. (2.)

KUVA 2. Avattu Nucu-alusta (2)

Erityisen rauhoittava vaikutus saadaan käyttämällä sykeääntä sekä kohdun äänimaailmaa. Tämä turvallinen äänimaailma yhdistettynä siitä luotavaan tuntoaistimukseen auttaa erityisesti keskosia ja koliikista kärsiviä vauvoja rauhoittumaan sekä nukkumaan paremmin. Nucuja on testattu vuodesta 2016 lähtien Oulun yliopistollisen sairaalan keskolassa, jossa alustan vauvojen hyvinvointia edistävää vaikutusta on pystytty todentamaan käytännössä. 2019 keväällä alkavassa väitöstutkimuksessa Nucu-alustat ovat mukana selvittämässä äidin oman sykkeen tuntuman vaikutusta keskosen hyvinvointiin. (2.)

Nucujen eri tuoteversiot näyttävät ulkoisesti samalta, mutta niiden sisältämä ohjelmisto vaikuttaa laitteen mahdollisiin toimintoihin. Sairaalakäytössä Nucut eroavat ulkoisesti kuluttajaversioista niissä käytettävän sinisen sairaalakankaan vuoksi. Tarkempi kuvaus alustojen rakenteesta on liitteessä 1. Erityispiirteenä ensimmäisen sukupolven Nucu-alustoissa on niiden kyvyttömyys viestiä ulkoisten laitteiden tai verkkojen kanssa. Alustat eivät sisällä minkäänlaista radiotek-

niikkaa. Tämä tekee laitteista turvallisia tietoturvamielessä sekä rauhoittaa vanhempia, jotka ovat huolissaan radioaaltojen mahdollisesta vaikutuksesta vauvan terveyteen ja kehitykseen.

1.3 Opinnäytetyön tausta ja tavoitteet

Nucu Oy:llä ei ole käytössä sähköistä järjestelmää, johon Nucu-alustat tai niihin liittyvät tiedot tallennettaisiin. Yksilöityjä laitetietoja, päivitys- ja asennushistoriatietoja, asiakastietoja tai malli- ja ohjelmistoversiotietoja ei automaattisesti tallenneta, rekisteröidä tai päivitetä yksittäiseen järjestelmään, vaan niitä hallitaan manuaalisesti. Alustojen tietojen hallinta ja seuraaminen on tämän vuoksi todella työlästä ja vaikeaa. Ohjelmistojen päivitykset asiakkaalla oleviin Nucuihin on mahdotonta tehdä ilman henkilökohtaista käyntiä asiakkaan luona. Ilman automatisoitua järjestelmää, jossa tiedot pysyvät ajantasaisina, sekä mahdollisuutta päivittää asiakkaalla olevia laitteita ilman henkilökohtaista käyntiä on yritystoimintaa ja asiakaspalvelua vaikea ylläpitää ja hallita.

Opinnäytetyön tavoitteena on suunnitella järjestelmä, jolla Nucu-alustat saadaan sähköiseen järjestelmään jo valmistusvaiheen aikana. Järjestelmän on tarkoitus automatisoida alustojen rekisteröinti, ohjelmistojen asennus ja päivitys sekä kaikki tarvittava tietojen päivittäminen. Alustoihin tehdään kahdenlaista ohjelmistoasennusta: tehdasasennus sekä mallikohtaisen ohjelmiston päivitys.

Opinnäytetyön laajuuden vuoksi toteutus jaettiin seuraaviin vastuualueisiin:

- Android-sovellus
- web-sovellus
- palvelinympäristö
- tietorakenteiden suunnittelun päävastuu.

Atte Paitsolan vastuualueina ovat web-sovellus ja palvelinympäristö. Jani Raipalan vastuualueita ovat Android-sovellus ja tietorakenteiden suunnittelun päävastuu.

2 JÄRJESTELMÄN VAATIMUSTEN KARTOITUS

Ennen varsinaista järjestelmän vaatimusmäärittelyä suoritimme kartoituksen, jonka tarkoituksena oli selvittää vaatimukset, joihin täytyisi kiinnittää erityistä huomiota perusvaatimusten lisäksi. Tällä varmistettiin toteutettavan järjestelmän riittävyys teknisesti sekä toiminnallisesti myös tulevaisuudessa. Kartoituksella selvitettiin, kuinka Nucu-alustoja sekä niiden tietoja tällä hetkellä käsitellään valmistuksessa, tehdas- ja malliasennuksessa, asiakkaalla ollessa, huolto- ja ongelmatilanteissa sekä alustan palautuessa asiakkaalta. Täytyi varmistua siitä, että suunnittelemamme järjestelmä koostuu ratkaisuista, jotka voidaan integroida osaksi olemassa olevia käytännön prosesseja. Toisaalta tehtävämme oli miettiä, ovatko olemassa olevat prosessit oikeanlaisia ja tarvitseeko niitä mahdollisesti muuttaa, jotta järjestelmästä tulee vaatimusten mukainen. Järjestelmän tuli ottaa huomioon perusvaatimusten lisäksi muun muassa tulevaisuuden tuotannon skaalautuvuus, Nucu-alustoihin mahdollisesti tulevaisuudessa kehitettävät lisätoiminnot sekä lääkinnällisiä laitteita koskevat direktiivit ja määräykset.

2.1 Lähtötilanne

Nucu-alustat suunnitellaan Nucu Oy:llä ja ne valmistetaan yhteistyössä alihankkijoiden kanssa. Valmistusprosessin aikana ne tehdasalustetaan perusohjelmistolla, jotta niiden sisältämä elektroniikka toimii. Ilman tehdasalustusta Nucu-alustat eivät esimerkiksi pysty kommunikoimaan USB-väylää pitkin, jolloin mallikohtainen ohjelmistoasentaminen olisi mahdotonta. Tehdasasennuksen lisäksi alustat yksilöidään nimellä, joka teipataan niiden ohjauspaneelin tarralla (kuva 3). Valmistetut Nucu-alustat toimitetaan Nucu Oy:n konttorille. Alustoihin asennetaan asiakkaan tarvitsema mallikohtainen ohjelmisto, ja laitteen sekä asennetun ohjelmiston tiedot kirjataan manuaalisesti ylös sähköiseen muotoon. Nucu-alusta on tässä vaiheessa valmis toimitettavaksi asiakkaalle.

KUVA 3. Nucu-alustan etupaneeli ja siihen kiinnitetty tunnistetarra

Nucun ollessa asiakkaalla mahdolliset ohjelmapäivitykset ja ongelmatilanteet hoidetaan henkilökohtaisella käynnillä. Ohjelmapäivitykset tehdään kytkemällä päivitettävä Nucu-alusta USB-kaapelilla kannettavaan tietokoneeseen ja siirtämällä tarvittavat tiedostot alustaan. Tehdyt ohjelmistopäivitykset ja toimenpiteet kirjataan konttorilla laitetiedot sisältävään tiedostoon manuaalisesti. Asiakaspalautetta otetaan vastaan kirjallisesti tai suullisesti. Asiakaspalautteita ei tallenneta pysyvästi mihinkään arkistoihin tai järjestelmiin.

2.2 Tulevaisuus

Projektin aloitushetkellä Nucu-alusta on ainoa Nucu Oy:n markkinoilla oleva tuote. Tulevaisuudessa on kuitenkin tarkoitus kehittää erilaisia Nucu-malleja sekä mahdollisesti kokonaan uusia hyvinvointituotteita. Todennäköisiä lisäominaisuuksia, joita uudet Nucu-mallit sisältävät, ovat esimerkiksi sensori- ja radiotekniikan lisääminen sekä laitekohtainen statistiikan kerääminen. Erilaisia lisälaitteita on kehitteillä laajentamaan palvelutarjontaa sekä luomaan uusia käyttömahdollisuuksia Nucu-alustoille. Yhtenä esimerkkinä on lisälaite, jolla äidin oma syke voidaan nauhoittaa ja siirtää Nucu-alustaan. Lisälaitteiden ohella luodaan myös erilaisia malliohjelmistoversioita, joiden avulla alustojen toimintoja voidaan yksilöidä paremmin vastaamaan eri asiakasryhmien tarpeita. Tulevaisuudessa tuotteita, tuotteiden eri malliversioita sekä erilaisia mallikohtaisia ohjelmistoversioita onkin huomattavasti enemmän kuin lähtötilanteessa. Myynnin lisääntyessä valmistettavien laitteiden tuotantomäärät kasvavat myös huomattavasti.

Nucu-alustat luokitellaan hyvinvointituotteeksi. Tulevaisuudessa Nucu Oy voi kuitenkin valmistaa Nucu-malleja tai täysin uusia tuotteita, jotka luokitellaan lääkinnällisiksi laitteiksi. Lääkinnällisiä laitteita koskevat direktiivit ja määräykset ovat

huomattavasti tiukempia kuin hyvinvointilaitteilla. EU:n uusi vuonna 2020 voimaan tuleva lääkinnällisiä laitteita koskeva asetus (2017/745/EU) määrittää tarkasti esimerkiksi, kuinka laitteet tulee yksilöidä, kuinka kattava laitetieto- ja historia valmistajalla täytyy ylläpitää ja kuinka laitteiden jäljitettävyys tulee hoitaa (3).

2.3 Vaatimusmäärittely

Vaatimusmäärittely on jaettu järjestelmän eri osia koskeviin sekä yleisiä käyttöliittymiä koskeviin vaatimuksiin. Käyttöliittymävaatimukset koskevat Android- sekä web-sovellusta.

2.3.1 Android-sovellus

Sovelluksen tulee olla selkeä ja yksinkertainen käyttää. Käyttäjän itse antamat syötteet tulee minimoida ja toiminnallisuudet automatisoida mahdollisimman pitkälle. Mitä vähemmän käyttäjä joutuu tekemään syötteitä, sitä vähemmän mahdollisia virhetilanteita erilaisissa asennus-, rekisteröinti- ja kopiointioperaatioissa pääsee tapahtumaan. Päätoiminnallisuudet, joita käyttäjän täytyy pystyä sovelluksella suorittamaan, ovat

- Nucu-alustan sähköinen tunnistaminen
- Nucu-alustan rekisteröiminen tietokantaan
- Nucu-alustan tehdasasennus
- Nucu-alustan mallikohtainen asennus / päivitys.

Asennuksissa tarvittavat tiedostot sovellus lataa palvelimelta ja kopioi Nucu-alustaan USB-väylää pitkin. Ohjelmistorakenne täytyy luoda tavalla, joka mahdollistaa hyvät jatkokehitysedellytykset. Luotava sovellus toimii pohjana, johon tulevaisuudessa lisätään uusia ominaisuuksia ja toiminnallisuuksia. On tärkeää noudattaa modulaarista ohjelmointitapaa ja ohjelmistoprojektirakennetta.

Sovelluksen tulee toimia laitteissa, joissa on Android 6.0 (API 23), ja sitä uudemmissa älylaitteissa. Tiedonsiirtoa ja Nucu-alustan tunnistusta varten käytettävässä älylaitteessa tulee olla USB-väylä sekä kamera. Tiedostojen lataamista varten toimiva verkkoyhteys on pakollinen.

2.3.2 Web-sovellus

Web-sovelluksen tarkoitus on näyttää käyttäjälle tietoja yksittäisistä laitteista. Sovellus on tarkoitettu asiakaspalvelun ja markkinoinnin tueksi sekä yrityksen sisäiseen käyttöön. Sovelluksen tarjoama tieto tulee olla näitä tarkoituksia varten relevanttia ja selkeästi esitettyä. Sovelluksen tulee näyttää ainakin yksittäisten laitteiden tunnistetiedot sekä laitteisiin asennettujen ohjelmistojen perustiedot.

Luotavan sovelluksen tulee toimia moderneissa verkkoselaimissa ja käyttää suojattua yhteyttä palvelimen kanssa kommunikoitiin. Sovellus noudattaa myös muita hyvien tapojen mukaisia tietoturvakäytänteitä. Sovellus ei saa käyttää resursseja tarpeettomasti eikä säilyttää dataa tarpeettomasti.

Luodun sovelluksen tulee olla projektirakenteeltaan ajantasainen sekä helposti jatkokehitettävä. Sovelluskehityksessä käytettävät työkalut ja työtavat tulee valita tavalla, jolla voidaan varmistua projektin kehitysmahdollisuuksista tulevaisuudessa mahdollisimman hyvin. Luotavan projektirakenteen tulee olla modulaarinen ja noudattaa yleisiä sekä alusta- ja työkalukohtaisia parhaita ohjelmointitapoja. Hyvien ohjelmointikäytänteiden noudattamisen tarkoitus on taata toiminnallisuuksien helppo lisääminen tulevaisuudessa sekä projektin helppolukuisuus.

2.3.3 Palvelinympäristö

Luotava palvelinympäristö on tarkoitettu Android- ja Web-sovellusten verkko-ominaisuuksien tueksi sekä tarvittavien tietojen- ja tiedostojen tallentamiseksi. Palvelinympäristön tulee sisältää tietokanta, REST-API ja jokin tiedostojen jakamiseen tarkoitettu ohjelmisto. Palvelimen on oltava tavoitettavissa verkkoyhteyden välityksellä.

Tietokanta sisältää tietoja yksittäisistä laitteista, asiakkaista, tuotemalleista, ohjelmistoista sekä niiden välisistä yhteyksistä. Tietokantarakenne täytyy suunnitella tavalla, joka mahdollistaa uusien tuotemallien ja ohjelmistoversioiden kehityksen tulevaisuudessa. Tuotevariantit voivat koostua joko muutoksista ohjelmistossa, fyysisissä ominaisuuksissa tai molemmissa. Tietokantarakenteen tulee tukea eri ohjelmistojen ja fyysisten laitteiden yhdistelmiä.

REST-API toimii rajapintana sovellusten ja tietokannan välissä. API vastaanottaa HTTP-protokolla hakuja ja vastaa niihin palauttamalla dataa JSON-formaatissa. JSON valittiin formaatiksi, koska se on universaali, tekstipohjainen formaatti, yleisesti käytössä, sekä tuettu kattavasti Android- sekä web-ympäristöissä. REST-API:n tulee sietää vikatilanteita sekä palautua niistä automaattisesti. Vikatiloissa tallennetun datan tulee säilyä muuttumattomana, vaikka API lakkaisi toimimasta.

Ohjelmistoa varten täytyy hankkia tarvittavat laiteratkaisut eli fyysinen palvelin. Palvelimen tulee olla tavoitettavissa vuorokauden ympäri mistä tahansa Internetin välityksellä. Kaiken palvelimelta lähtevän datan täytyy olla suojattua, jos se vain on mahdollista. Palvelimen tulee pystyä tekemään itsestään varmuuskopiot halutuin aikavälein.

Palvelimen täytyy olla skaalautuva tulevaisuuden tarpeita ja ohjelmistojen jatkokehitystä varten. Palvelimen on tarvittaessa pystyttävä tarvittaessa tasaamaan kuormitusta (engl. Load Balancing) ja käyttämään sisällönjakeluverkkoa (engl. Content Delivery Network). Edellä mainitut ominaisuudet takaavat palvelimen toiminnan odottamattomissa tilanteissa. Palvelimella on oltava kattavat integraatiomahdollisuudet tulevaisuudessa käytettäviä tarkkailutyökaluja ja sovelluksia varten.

2.3.4 Käyttöliittymät

Web- ja Android-sovellusten käyttöliittymien tulee noudattaa alustakohtaisia ohjeistuksia. Käyttöliittymäelementtien tyylimäärittelyjen ja mahdollisten animaatioiden yhtenäisyys yleisten ohjeistusten kanssa varmistavat intuitiivisen ja helpon käyttökokemuksen. Käyttöliittymien tulee olla mahdollisimman yksinkertaisia, helppokäyttöisiä ja responsiivisia. Ylimääräisiä elementtejä ja tietoja, joita käyttäjän ei ole tarpeellista nähdä, ei saa olla näkyvillä. Aktiiviset toiminnot tulee visualisoida selkeästi esimerkiksi animaatiolla. Aktiivisia toimintoja ovat esimerkiksi

- tiedostojen lataaminen ja kopiointi
- verkkohaut ja niiden vastausten odottelu
- muut toiminnot, joissa käyttäjän tulee odotella operaation valmistumista.

Toimintojen ollessa valmiita sekä erilaisissa virhetilanteissa käyttäjälle tulee ilmoittaa niistä selkeästi visuaalisella viestillä käyttöliittymässä. Käyttöliittymien tekstit kirjoitetaan suomen kielellä. Värimaailman tulee olla yhtenäinen Nucu Oy:n yleisen värimaailman kanssa. Päävärinä on petrolinsininen (#005f6a, kuva 4).

KUVA 4. Petrolinsininen värimaailma Nucu.fi-verkkosivustolla (2)

3 MENETELMÄT JA TYÖKALUT

3.1 Projektinhallinta

3.1.1 Ketterä projektinhallintamenetelmä Scrum

Scrum on ketterä projektinhallintamenetelmä. Sitä käytetään erityisesti ohjelmistoprojektien hallinnassa. Tuotteen vaatimukset ja ominaisuudet listataan projektin alussa kehitysjonoon (engl. Product Backlog). Scrumissa kehitystyö tehdään yleensä 1–4 viikon mittaisissa jaksoissa (engl. Sprint). Jokaisen jakson tehtävät (engl. Sprint Backlog) määritetään ennen jakson alkua kehitysjonosta. Jakson aikana tavoitteena on tuottaa käyttökelpoinen tuoteversio esiteltäväksi asiakkaalle. Näin ohjelmiston eri ominaisuuksia ja toimintoja voidaan koekäyttää ja testata projektin alusta asti. (4.)

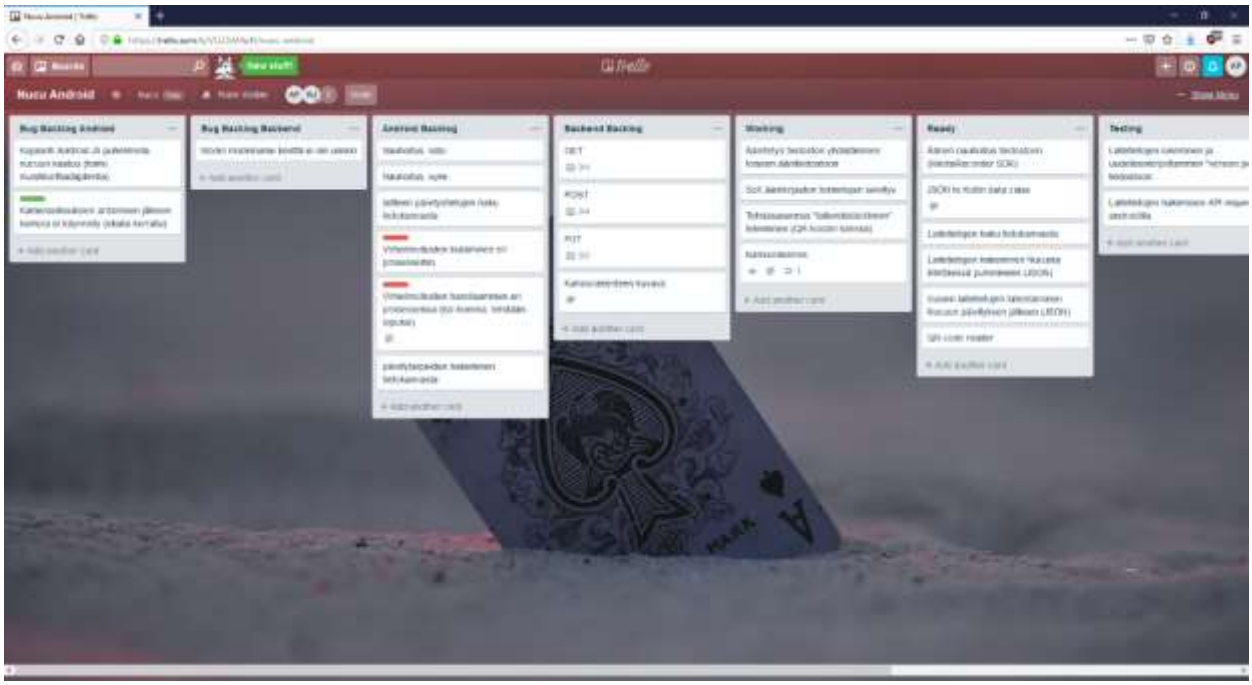
Scrumissa dokumentaation ja erilaisten palaverien tarve pyritään minimoimaan. Palaverit ja niiden maksimipituus on aina ennalta sovittuja. Scrumissa pidetään päivittäinen, enintään 15 minuutin mittainen palaveri, jossa käydään ryhmäjäsenkohtaisesti läpi

- edellisen päiväpalaverin jälkeen tapahtunut kehitystyö
- tulevan päivän tavoitteet ja työt
- mahdolliset ongelmat ja esteet. (4.)

Tämän lisäksi jokaisen kehitysjakson alussa pidetään suunnittelupalaveri, jossa kehitysjakson aikana tehtävät työt suunnitellaan. Palautetta ohjelmistosta saadaan jokaisen jakson jälkeen. Palautteeseen voidaan reagoida ketterässä Scrum-menetelmässä joustavasti ja tarvittaessa alkuperäisiä ohjelmistolle asetettuja vaatimuksia voidaan tarpeen vaatiessa päivittää ja muokata (4). Alusta asti oli selvää, että projektimme tulee tehdä ketterillä menetelmillä, koska suuri osa järjestelmän vaatimuksista ja niiden toteutuksesta tarkentuvat vasta kehitystyön aikana. Perinteistä vesiputousmenetelmää ei missään vaiheessa edes harkittu käytettäväksi projektissa.

3.1.2 Verkkopohjainen projektinhallintajärjestelmä Trello

Trello on ilmainen selainpohjainen projektinhallintajärjestelmä. Palvelussa tehtävälle projektille luodaan taulu, johon lisätään tarpeen mukaan tehtäväkortteja sisältäviä listoja (kuva 5). Listoilla ja korteilla voidaan kätevästi hallita esimerkiksi projektin eri tehtäviä ja tehtävänjakoa sekä seurata tehtävien etenemistä. Trello sisältää paljon yksityiskohtaisia ominaisuuksia, joilla tehtävien prioriteettia, kuvausta ja aikataulutusta voidaan määritellä hyvinkin tarkasti. Trello soveltuu hyvin käytettäväksi ketterissä ohjelmistoprojekteissa. Näistä syistä Trello valittiin avuksi projektin tehtävienhallinnassa. (5.)



KUVA 5. Projektin tehtävälistausta Trello-projektinhallintajärjestelmässä.

3.1.3 Microsoft Office 365

Nucu Oy tarjosi käyttöömme yrityskäyttöön tarkoitetun Office 365 -palvelun. Projektin aikana kaikki dokumentit ja tiedostot tallennettiin OneDrive for Business -pilvipalveluun. OneDrive-yritysversio on turvallisuustasoltaan parempi kuin perustasoinen OneDrive-palvelu. Emme tarvinneet projektin aikana muita palveluita tiedostojen ja varmuuskopioiden säilyttämiseen. Yleisesti ohjelmistoprojekteissa on käytössä jonkunlainen versionhallinta, kuten GitHub. Me emme sellaista kui-

tenkaan käyttäneet. Nucu Oy:llä ei ollut omaa yksityistä GitHub-tiliä, jolla versionhallinnan olisi voinut hoitaa. Toisaalta tämä ei ollut tarpeellista, koska ryhmän jäsenet eivät työskennelleet yhdessä samojen ohjelmistojen kanssa, vaan molemmilla oli kehitettävänä omat ohjelmistot.

3.2 Android-sovelluskehitys

3.2.1 Kotlin-ohjelmointikieli

Android-sovellus toteutettiin Kotlin-ohjelmointikielellä. Kotlin on JetBrainsin kehittämä ja toinen Googlen virallistamista Android-ohjelmointikielistä. Perinteisesti Android-ohjelmoinnissa käytettävään Java-ohjelmointikieleen verrattuna Kotlin tuottaa modernimpaa, lyhyempää ja ylläpidettävämpää koodia. Käyttöliittymäelementtien interaktioiden tekeminen on Kotlinilla huomattavasti kätevämpää kuin Javalla. Syntaksi on lyhyempää sekä elementteihin viittaaminen koodissa ja niihin liittyvän toiminnallisen koodin yhdistäminen on huomattavasti yksinkertaisempaa (kuva 6). Koodimäärän pieneneminen, koodin ylläpidettävyys sekä syntaksin opettelun helppous olivat syitä, joiden perusteella päädyimme käyttämään Kotlinia sovelluksen ohjelmointikielenä Javan sijasta. (6.)

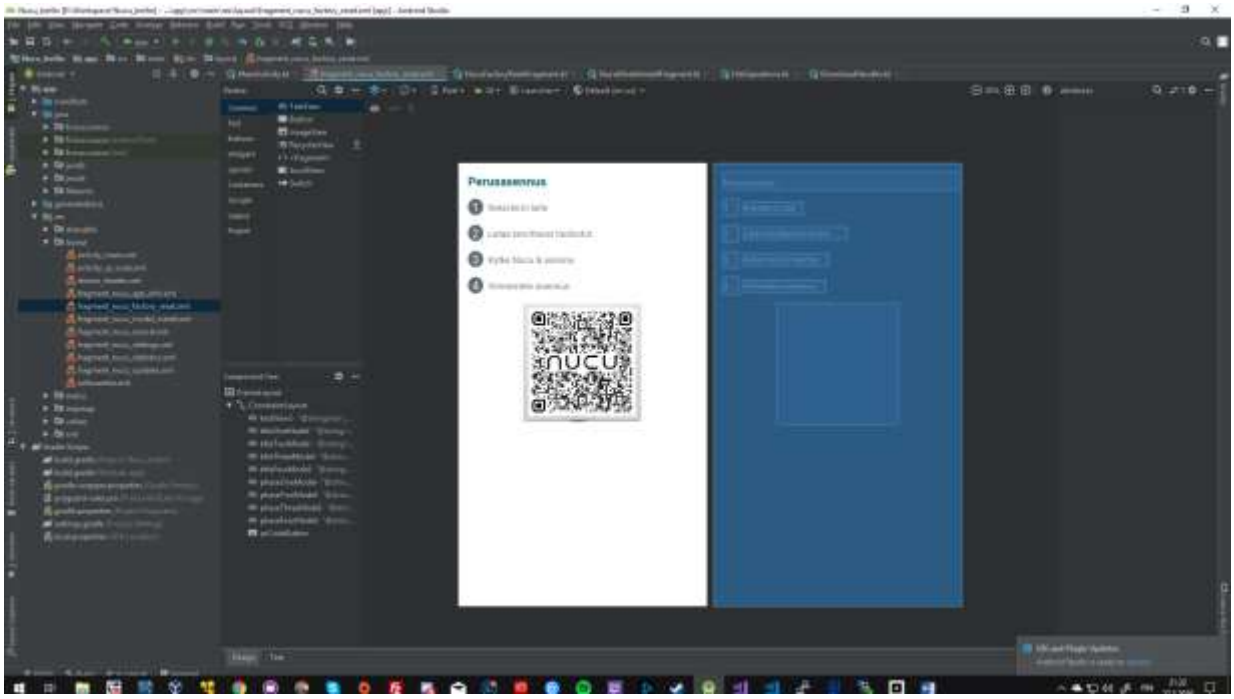
```kotlin
// KOTLIN
data class EsimerkkiHenkilo
(
    val name: String
)
```

```java
// JAVA
class EsimerkkiHenkilo {

    private String name;

    public EsimerkkiHenkilo(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

KUVA 6. Yksinkertainen luokan määrittely Kotlinilla ja Javalla

3.2.2 Android Studio -ohjelmointiympäristö

Android-sovelluksen luomisessa käytettiin Android-käyttöjärjestelmän virallista Android Studio -ohjelmointiympäristöä (kuva 7). Android Studio on ilmainen ja se toimii Windows-, Linux- sekä MacOS-käyttöjärjestelmissä. Android Studio on suunniteltu nimenomaan Android -kehitystä varten. Sovelluksen kaikki ohjelmakoodi, käyttöliittymät sekä testaus tehtiin Android Studiolla ja sen lisäosilla. Android Studio sisältää mahdollisuuden emuloida erilaisia Android-laitteita ja suorittaa niillä ohjelmiston testausta. Emulaattorin käyttö jäi projektissa kuitenkin vähäiseksi, koska suoritimme testausta alusta asti oikeilla Android-älypuhelimilla. Erityisesti USB-tietoliikenteen testaaminen emulaattorilla olisi ollut hyvin vaikeaa. (7.)

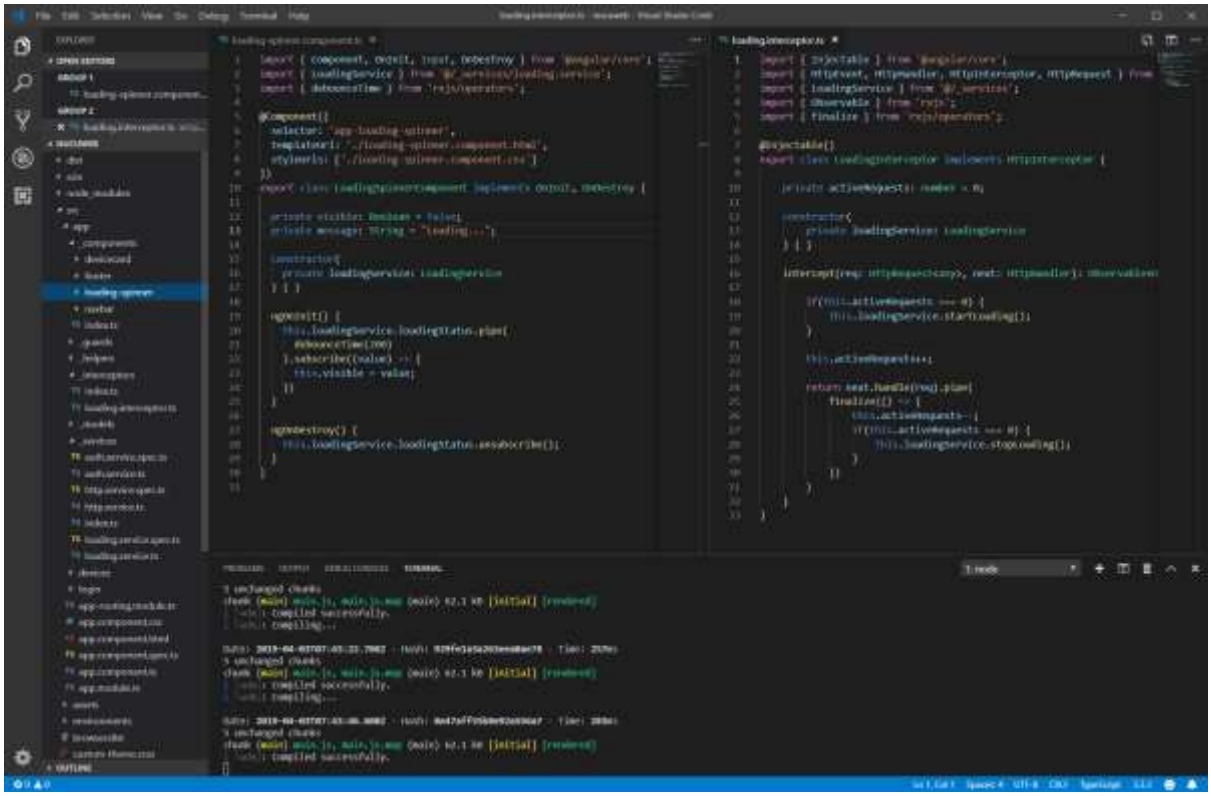


KUVA 7. Android Studio IDE

3.3 Tietokanta, REST-API ja verkkosovellus

Sekä REST-API (eng. Representational State Transfer) että verkkosovellus luotiin JavaScript-pohjaisilla ohjelmistokehyksillä. Molempien kehitykseen käytettiin Visual Studio Code -editoria (kuva 8). Koodieditoriksi valikoitui Visual Studio

Code, koska sille on saatavilla kattavat JavaScript- ja TypeScript-lisäosat ja se on ilmainen.



KUVA 8. Visual Studio Code ja projekti

3.3.1 Oliopohjaisen datan kartoittaminen relaatiotietokantaan

MySQL-tietokannan ja REST-API:n luomiseen käytettiin Sequelize objekti—relaatiodata-kartoituskehystä. Sequelizella oliopohjainen data voidaan tallentaa relaatiotietokantaan. Vastaavasti tietokannan data voidaan muuttaa takaisin olioiksi. Oliopohjainen JSON-data on yleisesti käytössä Android- sekä web-ympäristöissä. Sequelize helpottaa tietoliikenteen luomista järjestelmän osien välillä sekä yhtenäistää tietorakenteita. Tietokantaan voidaan syöttää dataa JSON-formaatissa valmiiksi määriteltyjen mallien avulla (kuva 9). Tietokannasta Sequelizen avulla haettu data saadaan suoraan JSON-formaatissa. (8.)

```
const Henkilo = sequelize.define('henkilo', {
    etuNimi: Sequelize.STRING,
    sukuNimi: Sequelize.STRING,
    henkiloTunnus: {
        type: Sequelize.STRING,
        unique: true,
        primaryKey: true
    },
    ika: {
        type: Sequelize.INTEGER,
        validate: {
            min: 0
        }
    }
})
```

KUVA 9. Yksinkertaisen tietokantarakenteen määrittely Sequelizella

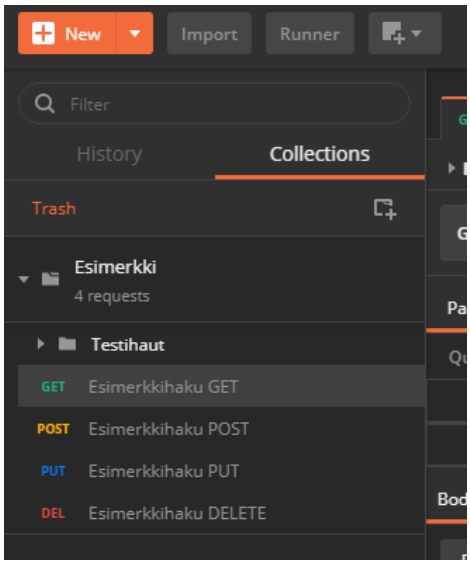
3.3.2 Modernit web-kehitystyökalut TypeScript ja Angular

Web-sovelluksen luontiin käytettiin TypeScript-pohjaista Angular-kehitysympäristöä. TypeScript on JavaScript-ohjelmointikielen jatke ja laajentaa sen toimintoja huomattavasti. Angular on nykyaikainen sovelluskehitysympäristö, joka mahdollistaa responsiivisten verkkosovellusten kehittämisen. Projektissa käytettiin Angulariin sisältyvää käyttöliittymäkirjasto Angular Material:ia (9).

3.4 Testaus

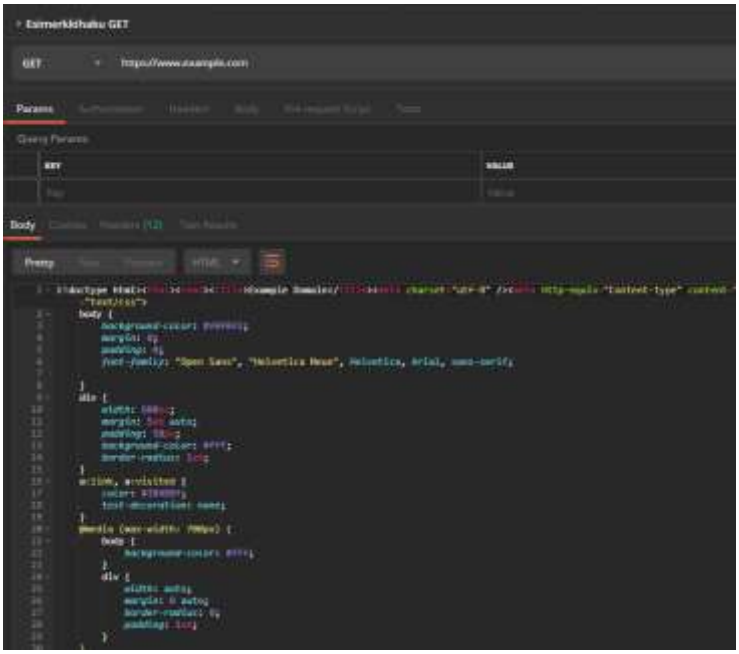
3.4.1 Verkkosovellusten kehitystyökalu Postman

REST-API:n testaus suoritettiin Postman-ohjelmalla. Postman on verkkosovellusten kehitykseen suunniteltu työkalu. Postmanin avulla verkkohakukokoelmia voidaan jakaa työryhmän kesken (kuva 10).



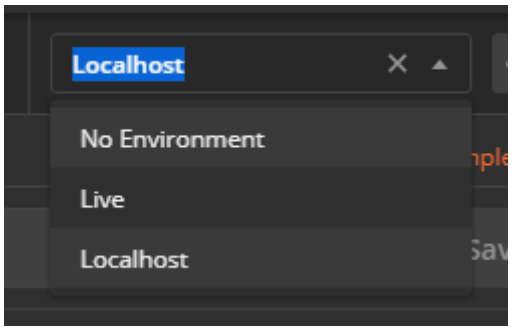
KUVA 10. Postman-kokoelma

Projektia varten luotiin verkkohakukokoelma, jota päivitettiin tarvittaessa sitä mukaa, kun REST-API:a kehitettiin. Samaa kokoelmaa voitiin käyttää sekä Android- että web-sovelluksen verkkohakujen testaukseen. Yhteisen kokoelman ansiosta ryhmän jäsenten ei täytynyt luoda omia kokoelmiaan ja tieto oli aina ajantasaista. Koska kokoelman sisältämät API-kutsujen vastaukset ovat samat kaikille käyttäjille, ongelmien jäljittäminen järjestelmän tiedonkulussa oli helppoa ja virheet löytyivät yleensä helposti (kuva 11). (10.)



KUVA 11. Postman-kutsu ja saatu vastaus

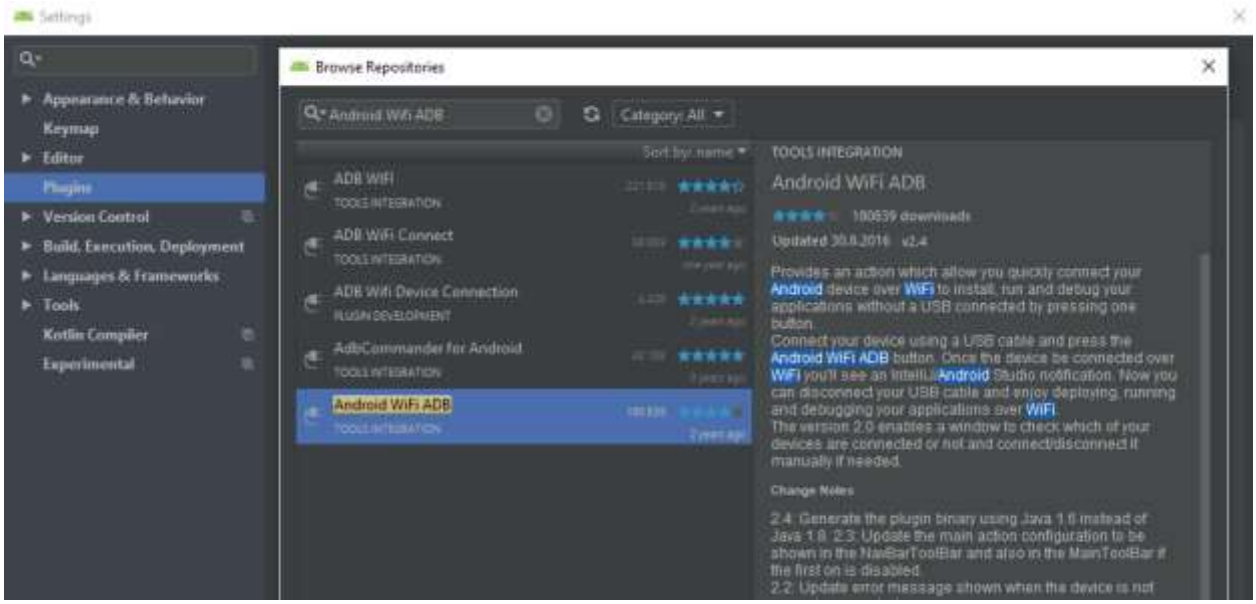
Postmanin ominaisuuksiin kuuluvat myös ympäristöt (engl. Environment, kuva 12). Ympäristöjen avulla oikean palvelimen ja kehittäjän tietokoneella olevan ohjelmiston testaamisen välillä vaihtaminen tapahtuu helposti.



KUVA 12. Ympäristöjen vaihtaminen

3.4.2 Android-sovelluksen langaton testaus

Android-sovelluksen testauksessa apuna käytettiin Android Studioon asennettavaa Android Wifi ADB -lisäosaa. Lisäosa on ilmainen ja sen voi asentaa helposti Android Studion lisäosaosiosta (engl. Plugins, kuva 13).



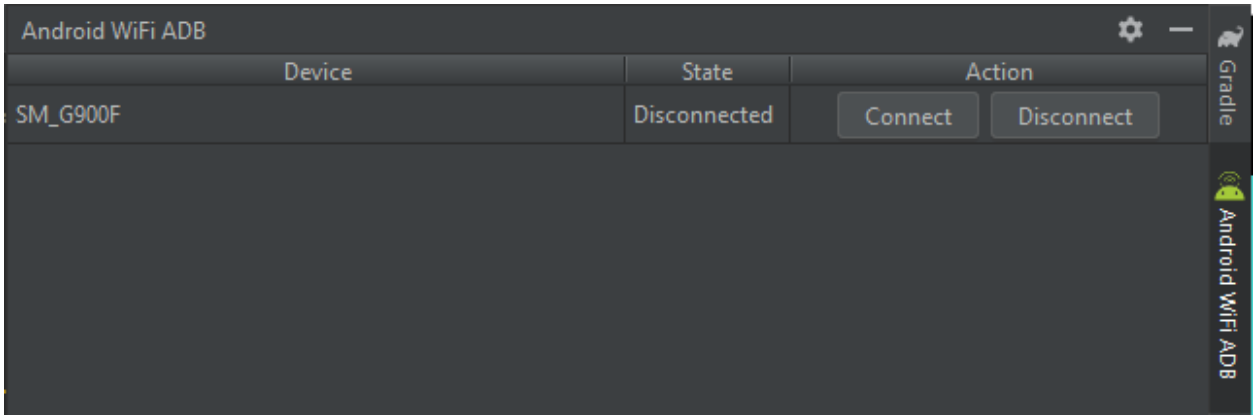
KUVA 13. Android Studion lisäosaosion haku- ja asennusikkuna

Lisäosan ansiosta Android Studio voi kommunikoida Android-laitteiden kanssa langattomasti. Tämä tarkoittaa sitä, että kehitettävä ohjelmisto voidaan asentaa Android-laitteeseen ja tämän jälkeen lukea ohjelman testaustietoja reaaliajassa

ilman langallista yhteyttä laitteen ja tietokoneen välillä. Ilman langatonta testausta olisi USB-tietoliikenteen testaaminen ollut melkeinpä mahdotonta, koska silloin USB-portti oli varattu massamuistilaitteelle. Android-laitteen ja Android Studion langattoman yhteyden luominen on varsin helppoa ja vaatii ainoastaan kolme vaihetta:

1. Yhdistä Android-laite ja Android Studio (tietokone) samaan langattomaan verkkoon.
2. Kytke puhelin kiinni tietokoneen USB-porttiin ja odota, että Android WiFi ADB tunnistaa sen.
3. Paina Android Wifi ADB:ssa Connect -painiketta (kuva 14). Langaton yhteys on nyt luotu ja puhelimen voi irrottaa tietokoneesta.

Android Wifi ADB mahdollistaa useiden laitteiden yhtäaikaisen testaamisen langattomasti.



KUVA 14. Android Wifi ADB:n laiteyhdistyspaneeli

4 TOTEUTUS

Järjestelmän päätehtävät eli Nucu-alustojen sähköinen tunnistus, rekisteröinti järjestelmään, tehdasasennus sekä käytettävän malliohjelmiston päivitys toteutettiin Android-sovelluksella. Rekisteröityjen laitteiden tietojen tarkastelua ja hallintaa helpottamaan luotiin moderni Angular-web-sovellus. Järjestelmän tietoliikennettä, tietojen tallennusta ja muita tehtäviä hoitamaan pystytettiin kattava palvelinympäristö.

4.1 Android-sovellus

4.1.1 Nucu-alustan sähköinen tunnistaminen

Nucu-alustojen yksilöinti ja tunnistaminen olivat keskeisiä järjestelmälle asetettuja vaatimuksia. Ilman vahvaa tunnistusta ei Nucu-alustoja voitaisi rekisteröidä tai niihin ei voitaisi asentaa sopivaa ohjelmistoa, koska järjestelmä ei tietäisi, minkä laitteen kanssa se on tekemisissä. Tulevaisuudessa, kun laitehistoriaa aletaan keräämään järjestelmään, täytyy Nucu-alustat pystyä tunnistamaan jokaisen niille suoritettavan operaation yhteydessä. Laitetunnistuksen todettiin olevan tärkeää monenlaisissa käyttötapauksissa nyt ja tulevaisuudessa, joten se päätettiin toteuttaa kahdella tavalla: langattomalla tunnistuksella sekä USB-väylän kautta tehtävällä langallisella tunnistuksella.

Langaton tunnistus

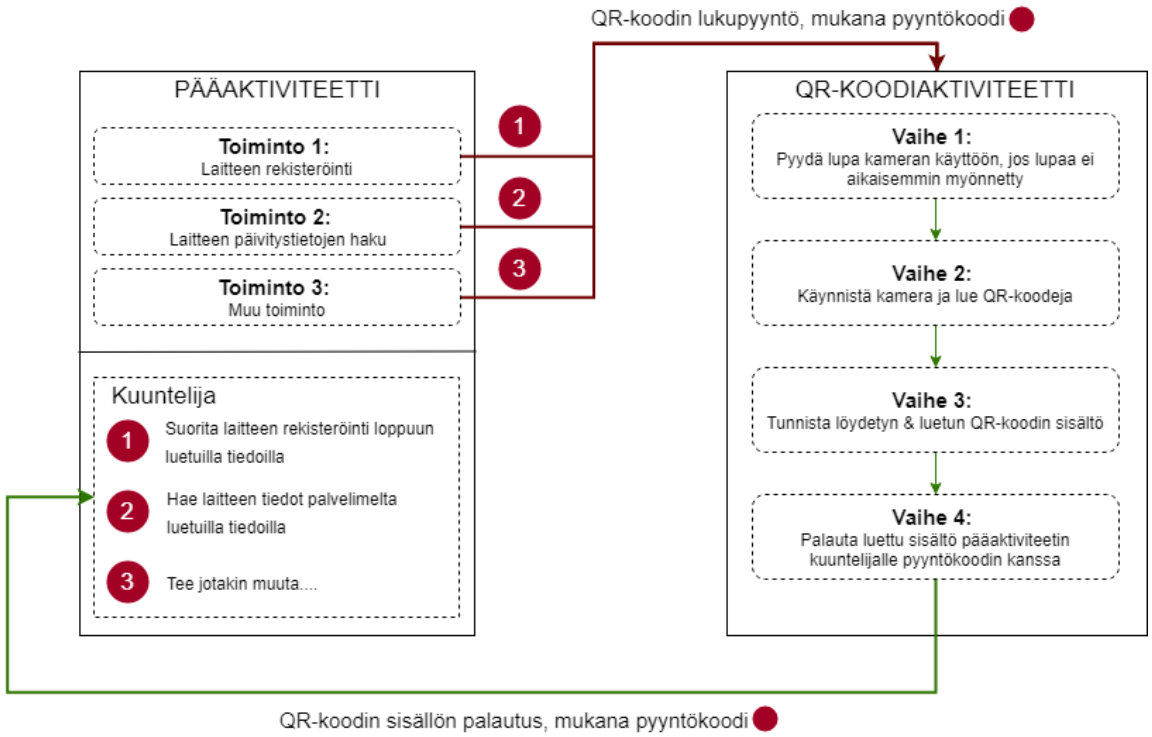
Langaton tunnistus, jossa Nucu-alustaa ei tarvitse kytkeä USB-kaapelilla kiinni Android-laitteeseen, toteutettiin QR-koodin lukemisella. Päätettiin, että jokaiseen Nucu-alustaan luodaan QR-kooditarra (kuva 15), jossa sen yksilöivät perustiedot ovat JSON-muotoisessa tekstijonossa. Tarrat luodaan ja liimataan Nucu-alustoihin valmistusprosessin loppuvaiheessa ennen rekisteröintiä ja tehdasasennusta. Lisätietoa QR-kooditarran luomisesta ja valinnasta käytettäväksi tunnistusprosessissa on liitteessä: Liite 2 Langaton tunnistus.

KUVA 15. Nucu-alustaan liimattu testivaiheen QR-koodi

QR-koodin lukemista varten Android-sovelluksessa luotiin sille oma aktiviteetti (engl. Activity) (11). Tämän aktiviteetin tehtävänä on

- pyytää käyttäjältä lupa käyttää Android-laitteen kameraa, jos lupaa ei vielä annettu
- käynnistää kamera ja alkaa lukemaan QR-koodeja
- tunnistettuaan QR-koodin tarkistaa, onko sen sisältämä tieto oikeanlaista (Nucu QR-koodi)
- oikeanlaisen QR-koodin löydettyään sammuttaa kamera ja palauttaa luettu tieto ohjelmiston pääaktiviteetissa olevalle kuuntelijalle (engl. Broadcast receiver).

Koska QR-koodin lukeminen ja luetun tiedon palauttaminen tehdään omassa aktiviteetissa, voidaan sitä pyytää helposti mistä tahansa näkymästä tai toiminnosta ohjelman pääaktiviteetista (engl. Main Activity) (kuva 16). Toimintaperiaate on, että QR-koodiaktiviteetti käynnistetään tarvittaessa pyynnöllä, jonka mukana lähetetään aina pyyntökoodi. Pyyntökoodin tarkoitus on auttaa tunnistamaan, mistä toiminnosta QR-koodin lukupyyntö on peräisin. QR-koodiaktiviteetti palauttaa tämän pyyntökoodin sekä luetun QR-koodin sisällön pääaktiviteettiin, jossa kuuntelija kaappaa palautuksen ja suorittaa halutun jatkotoimenpiteen riippuen palautetusta pyyntökoodista. QR-koodiaktiviteetti avautuu aina koko näytölle (kuva 17).



KUVA 16. QR-koodi- ja pääaktiviteetin välinen toimintaperiaate-esimerkki, kun joku toiminto pyytää QR-koodin lukemista

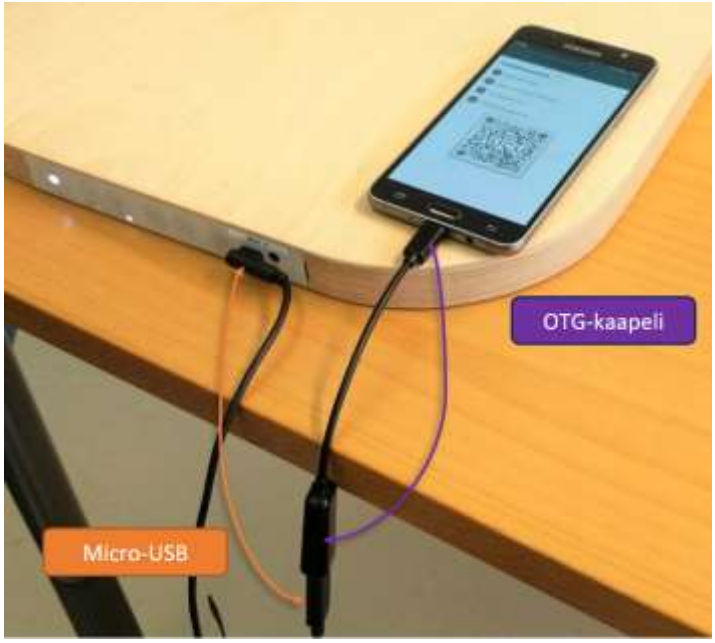
Ohjelmistorakenteellisesti ratkaisu on varsin yksinkertainen, mutta toimiva. QR-koodiaktiviteettia voidaan käyttää nyt ja tulevaisuudessa sovelluksessa helposti. Teknisesti QR-koodiaktiviteetti ja kaikki sen kameran käyttöön sekä QR-koodien lukemiseen luodut toiminnot toteutettiin Google Mobile Vision-ohjelmistorajapinnalla (12).

KUVA 17. Koko näytölle avautunut QR-koodiaktiviteetti valmiina lukemaan QR-koodin

Langallinen tunnistus

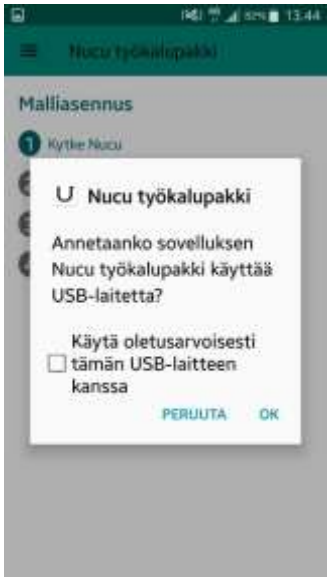
Langallisessa tunnistuksessa Android-laite kytketään Nucu-alustaan käyttäen OTG- ja Micro-USB kaapeleita (kuva 18). Android-laite toimii isäntänä (engl. Host) ja antaa virran siihen kytketylle Nucu-alustalle. Tämän vuoksi täytyy OTG-kaapeli kytkeä Android-laitteeseen ja Micro-USB-kaapeli sen jälkeen OTG-kaapelin ja Nucu-alustan väliin. Nucu-alustan tunnistetiedot luetaan sen massamuistissa sijaitsevasta JSON-muotoisesta laitetunnistetiedostosta. Sovellus ohjelmoitiin kuuntelemaan USB-laitteiden kytkemis- ja irrottamistapahtumia, jolloin laitetietojen lukeminen voitiin automatisoida laitekytkennän yhteydessä. Aina kun USB-laite kytketään Android-laitteeseen, sovellus

- tunnistaa USB-laitteen kytkemistapahtuman
- pyytää lupaa käyttää USB-laitetta, jos lupaa ei ole annettu (kuva 19)
- suorittaa selvityksen, jolla voidaan todeta kytketyn laitteen olevan Nucu-alusta
- tunnistaessaan Nucu-alustan lukee sen sisältämän laitetunnistetiedoston
- tallentaa kytketyn Nucu-alustan laitetiedot ohjelman muistiin, jolloin ne ovat saatavilla tarvittaessa.



KUVA 18. Android-laite kytkettynä Nucu-alustaan OTG- ja Micro-USB kaapeleilla

Tunnistaessaan USB-laitteen irrotustapahtuman sovellus tyhjentää irrotetusta laitteesta kerätyt tiedot.



KUVA 19. Android-sovellus kysyy lupaa käyttää kytkettyä USB-laitetta

Laitetunnistetiedosto, josta Nucu-alustan tiedot luetaan, luodaan alustan ensirekisteröinnin yhteydessä (kuva 20). Tiedosto päivitetään aina ajan tasalle, kun alustaan tehdään päivityksiä. QR-koodiin verrattuna laitetunnistetiedosto sisältää huomattavasti enemmän tietoja laitteesta. QR-koodi sisältää vain tarvittavat minimitiedot, joilla laite voidaan varmasti tunnistaa. Laitetunnistetiedosto on käytännössä suora kopio laitetiedoista, jotka on tallennettu palvelimelle. Tällä varmistetaan, että kattavat tiedot laitteesta ovat saatavilla langallisella tunnistuksella silloinkin, kun verkkoyhteyttä ei ole saatavilla.

KUVA 20. Perustietoikkuna Android -sovelluksessa luettaessa Nucu-alustan tiedot laitetunnistetiedostosta

4.1.2 Nucu-alustan rekisteröiminen ja tehdasasennus

Nucu-alustojen rekisteröintiä ja tehdasasennusta varten sovellukseen luotiin Tehdasasennus-näkymä. Rekisteröinti yhdistettiin tehdasasennuksen kanssa yhdeksi prosessiksi. Prosessi alkaa SD-muistikortin liittämisestä Android-laitteeseen muistikortinlukijan ja OTG-kaapelin avulla (kuva 21) ja päättyy tehdasasennetun SD-kortin kytkemiseen Nucu-alustaan. Prosessin alussa on muutama manuaalinen käyttäjän suorittama vaihe, jonka jälkeen kaikki loput vaiheet ennen SD-kortin asentamista Nucu-alustaan ovat automatisoituja.

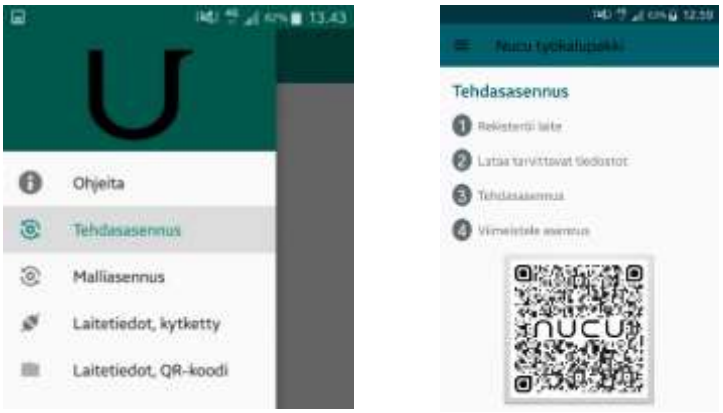
Erikoisen tehdasasennusvaiheesta tekee SD-kortin ja muistikorttiadapterin käyttö. Nucu-alustojen elektroniikka ei toimi ennen tehdasasennuksessa siihen kopioitavia ajureita. Tämän vuoksi SD-kortti tehdasasennetaan muistikortinlukijan avulla ja vasta tämän jälkeen kiinnitetään alustassa olevaan SD-korttipaikkaan. Tehdasasennuksen jälkeen Nucu-alusta voidaan tunnistaa USB-massamuistilaitteena sen omasta etupaneelissa sijaitsevasta USB-portista.

KUVA 21. SD-kortti kytketty Android-laitteeseen muistikortinlukijan sekä OTG-adapterin avulla

Asennusprosessin vaiheet

Manuaaliset vaiheet:

1. Kytketään SD-muistikortti Android-laitteeseen kiinni OTG-kaapelilla ja muistikortinlukijalla.
2. Navigoidaan Tehdasasennus-näkymään sovelluksen navigaatiovalikosta (kuva 22, vasemmalla).
3. Aloitetaan laitteen rekisteröintiprosessi painamalla QR-koodipainiketta (kuva 22, oikealla).
4. Luetaan rekisteröitävän ja tehdasasennettavan Nucu-alustan QR-koodi.

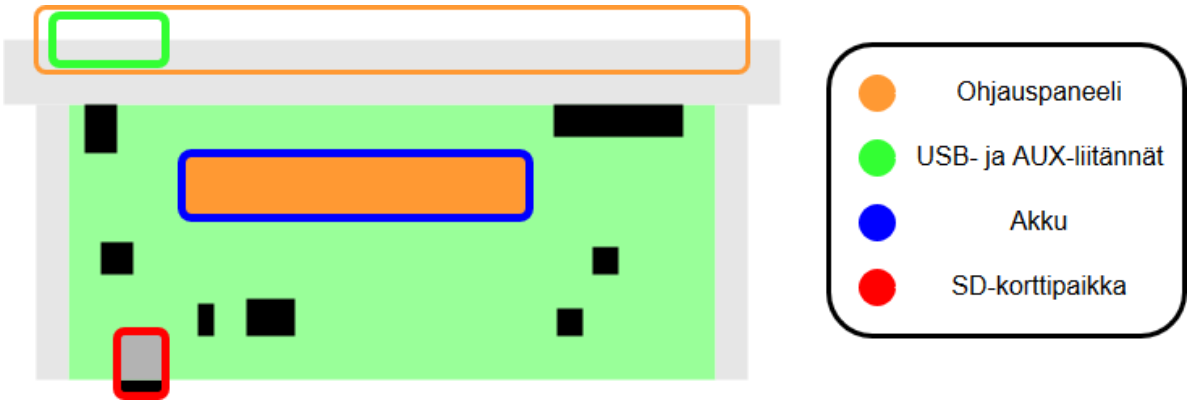


KUVA 22. Navigaatiovalikko sekä Tehdasasennus-näkymä

Automaattiset vaiheet:

5. Sovellus lähettää luetut tunnistetiedot palvelimelle, jossa Nucu-alusta rekisteröidään järjestelmään.
6. Palvelin palauttaa tiedot ladattavista tehdasasennustiedostoista.
7. Tarvittavat tiedostot ladataan palvelimelta.
8. SD-kortti tehdasasennetaan ladatuilla tiedostoilla.
9. Tiedot onnistuneesta asennuksesta lähetetään palvelimelle ja tallennetaan Nucu-alustan ohjelmistotietoihin sekä laitehistoriaan.
10. Päivitetyt laitetiedot palautetaan palvelimelta. Sovellus luo tiedoista JSON-muotoisen laitetunnistetiedoston.
11. Laitetunnistetiedosto kopioidaan SD-kortille ja käyttäjälle ilmoitetaan asennuksen olevan valmis ponnahdusikkunaviestillä.

Näiden vaiheiden jälkeen SD-kortti on tehdasasennettu ja siihen on luotu JSON-muotoinen laitetunnistetiedosto laitteen langallista tunnistusta varten. SD-kortti on valmis asennettavaksi Nucu-alustassa olevaan SD-korttipaikkaan (kuva 23).



KUVA 23. Kaaviokuva Nucu-alustan sisältämästä elektroniikasta. Alavasemmalla punaisella merkitty SD-korttipaikka, johon tehdasasennettu SD-kortti kiinnitetään.

4.1.3 Nucu-alustan mallikohtainen asennus

Mallikohtainen asennus suoritetaan aina tehdasasennettuun Nucu-alustaan. Mallikohtainen asennus tarkoittaa Nucu-alustan asiakaskohtaisen ohjelmiston

asentamista tai päivittämistä. Tehdasasennus saa Nucu-alustan toimintakuntoiseksi, malliasennus vaikuttaa alustan toiminnallisuuksiin. Projektin toteutushetkellä käytössä oli kolme mallikohtaista ohjelmistoa. Erona ohjelmistoissa olivat muun muassa niiden sisältämät ääniraidat sekä äänenvoimakkuuksien erot. Yhdessä ohjelmistoversiossa Nucu-alustan etupaneelin painikkeet on poistettu käytöstä kokonaan.

Mallikohtainen asennus on prosessina hyvin samanlainen kuin tehdasasennus. Asennusta varten sovellukseen luotiin sille oma Malliasennus-näkymä. Prosessi alkaa Nucu-alustan kytkemisellä Android-laitteeseen (kuva 24) ja päättyy onnistuneen asennuksen tietojen päivittämiseen palvelimelle sekä Nucu-alustan massamuistissa olevaan laitetunnistetiedostoon. Erona tehdasasennukseen ovat laitetunnistuksen tekeminen QR-koodin luennan sijaan langallisesti Nucu-alustan laitetunnistetiedostosta sekä asennuksen suorittaminen suoraan kytkettyyn alustaan.

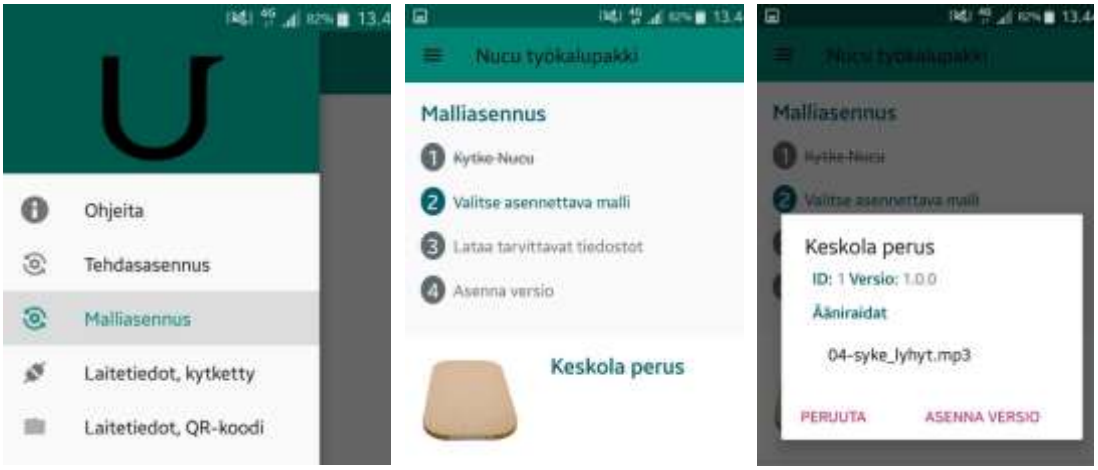
KUVA 24. Nucu-alusta kytkettynä Android-laitteeseen valmiina mallikohtaiseen asennukseen.

Malliasennukset alkuvaiheet ovat manuaalisia ja vaativat käyttäjän toimintaa. Asennuksen loppuvaiheet ovat automatisoituja.

Asennusprosessin vaiheet

Manuaaliset vaiheet:

1. Kytketään Android-laite kiinni Nucu-alustan USB-porttiin käyttäen OTG-kaapelia ja micro-USB-kaapelia (laitteen tunnistetiedot luetaan automaattisesti USB-laitteen kytkemistapahtumassa).
2. Navigoidaan Malliasennus-näkymään (kuva 25, keskellä) sovelluksen navigaatiovalikosta (kuva 25, vasemmalla) Sovellus hakee palvelimelta vaiheessa 1 saaduilla laitetiedoilla asennettavissa olevien ohjelmistojen tiedot automaattisesti näkymän avautuessa.
3. Aloitetaan malliasennus valitsemalla asennettava ohjelmisto listalta ja painamalla Asenna versio -painiketta (kuva 25, oikealla).



KUVA 25. Malliasennuksen manuaalisten vaiheiden käyttöliittymänäkymät

Automaattiset vaiheet:

4. Valitun ohjelmiston ääni- ja ohjelmistotiedostot ladataan palvelimelta.
5. Ladatut tiedostot asennetaan Nucu-alustaan.
6. Tiedot onnistuneesta asennuksesta tai päivityksestä lähetetään palvelimelle ja tallennetaan Nucu-alustan ohjelmistotietoihin sekä laitehistoriaan.
7. Päivitetyt laitetiedot palautetaan palvelimelta. Sovellus luo tiedoista JSON-muotoisen laitetunnistetiedoston.
8. Laitetunnistetiedosto kopioidaan SD-kortille ja käyttäjälle ilmoitetaan asennuksen olevan valmis ponnahdusikkunaviestillä.

Malliasennus on nyt valmis ja Android-laitteen voi irrottaa Nucu-alustasta.

4.1.4 Sovelluksen käyttöliittymä

Teema

Käyttöliittymän teemassa otettiin huomioon Nucu Oy:n käyttämä pääväri: petrolinsininen. Pääväristä luotiin värikartta, jonka perusteella sovelluksen käyttämät värit määritettiin. Tällä tavoin käyttöliittymän peruselementit ja animaatiot saavat automaattisesti oikeanlaisen värimaailman ja sovelluksen ulkoasu näyttää yhtenäiseltä yrityksen käyttämien muiden medioiden ulkoasujen kanssa (kuva 26).

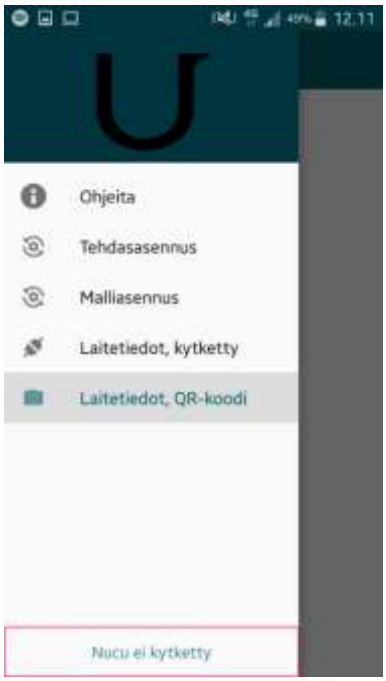
```
<color name="colorPrimary">#008577</color>
<color name="colorPrimaryDark">#00574B</color>
<color name="colorAccent">#D81B60</color>
<color name="colorAlert">#ff005d</color>
<color name="colorOk">#00E660</color>
<color name="titleTextColor">#005f6a</color>
<color name="inactiveColor">#B41D2D31</color>
```

KUVA 26. Osa ohjelmiston colors.xml-tiedostosta, jossa sovelluksen käyttämiä värejä määritetään

Navigaatio

Sovelluksen navigaatio toteutettiin vetolaatikkoelementillä (engl. Drawer, kuva 27). Navigaatiolaatikko on melko yleinen navigointitapa mobiililaitteissa, joten se on intuitiivinen käyttää. Navigaatiolaatikkoon sovitettiin navigaatiolinkkien lisäksi Nucu Oy:n logon u-kirjain sekä graafinen elementti, joka kertoo, onko Android-laitteeseen kytketty Nucu-alustaa. Navigaatiolaatikkoon jäi vielä tilaa tulevaisuudessa mahdollisesti kehitettävien toimintojen navigaatiolinkeille.

Yleisesti ajatus sovelluksen navigaatiossa oli, että toimintoihin pääsee käsiksi nopeasti yhden valikon kautta. Syviä navigaatiorakenteita, joissa käyttäjä joutuu etsimään toimintoja näkymien sisältä käsin, pyrittiin välttämään.

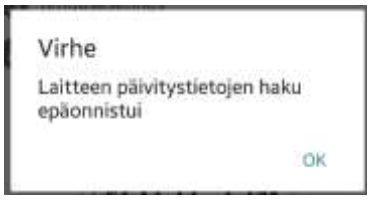


KUVA 27. Android-sovelluksen navigaatiolaatikko

Animaatiot, käytettävyys ja ilmoitukset

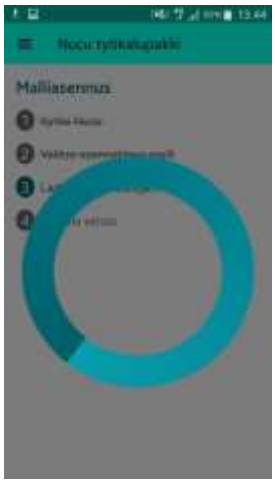
Monet sovelluksen toiminnot, kuten tiedostojen lataaminen palvelimelta, asennustehtävät sekä osa verkkohauista, kestävät ajallisesti kauan. Suuri osa toiminnoista myös automatisoitiin niin, että käyttäjän tarvitsee toiminnon aloituksen jälkeen ainoastaan odotella sen valmistumista. Tämän vuoksi toimintojen ja operaatioiden käynnistyminen, eteneminen, lopetus ja mahdolliset virhetilanteet halutaan viestiä käyttäjälle selkeästi.

Virhetilanteissa käyttäjää informoidaan tapahtuneesta virheestä yksinkertaisella ponnahdusikkunalla (kuva 28). Ikkunan sisältämästä viestistä käy ilmi, minkälainen virhetilanne on kyseessä, ja tarvittaessa ohjeistetaan mahdollisiin jatkotoimenpiteisiin. Samoin erilaisten prosessien ja tehtävien valmistuttua käyttäjälle näytetään vastaavanlainen ilmoitusikkuna, joka kertoo, mikä tehtävä on valmistunut. Esimerkiksi tehdasasennuksen ja malliasennuksen valmistuttua käyttäjälle näytetään ponnahdusikkuna, joka ilmoittaa onnistuneesta asennuksesta.



KUVA 28. Esimerkki ilmoitusikkunasta virhetilanteessa

Pitkien automatisoitujen operaatioiden ja verkkohakujen aikana käyttäjälle näytetään koko ruudun peittävä latausanimaatio (kuva 29). Animaatiolla viestitään selkeästi jonkin tehtävän olevan vielä kesken.



KUVA 29. Latausanimaatio, joka ilmoittaa käyttäjälle suoritettavan tehtävän olevan vielä kesken

Tehtävissä, joissa vaadittiin käyttäjän suorittamia toimintoja monissa eri vaiheissa, pyrittiin selkeyttä lisäämään informatiivisilla ja tehtävän edetessä päivittyvillä käyttöliittymäelementeillä (kuva 30). Tehtävien eri vaiheet eroteltiin toisistaan pienillä ulkoasullisilla eroilla. Näin käyttäjä oli tietoinen meneillään olevan tehtävän vaiheista ja häneltä odotetuista toimenpiteistä.

KUVA 30. Mallikohtaisen ohjelmiston asennusnäkymä ja asennuksen eri vaiheet

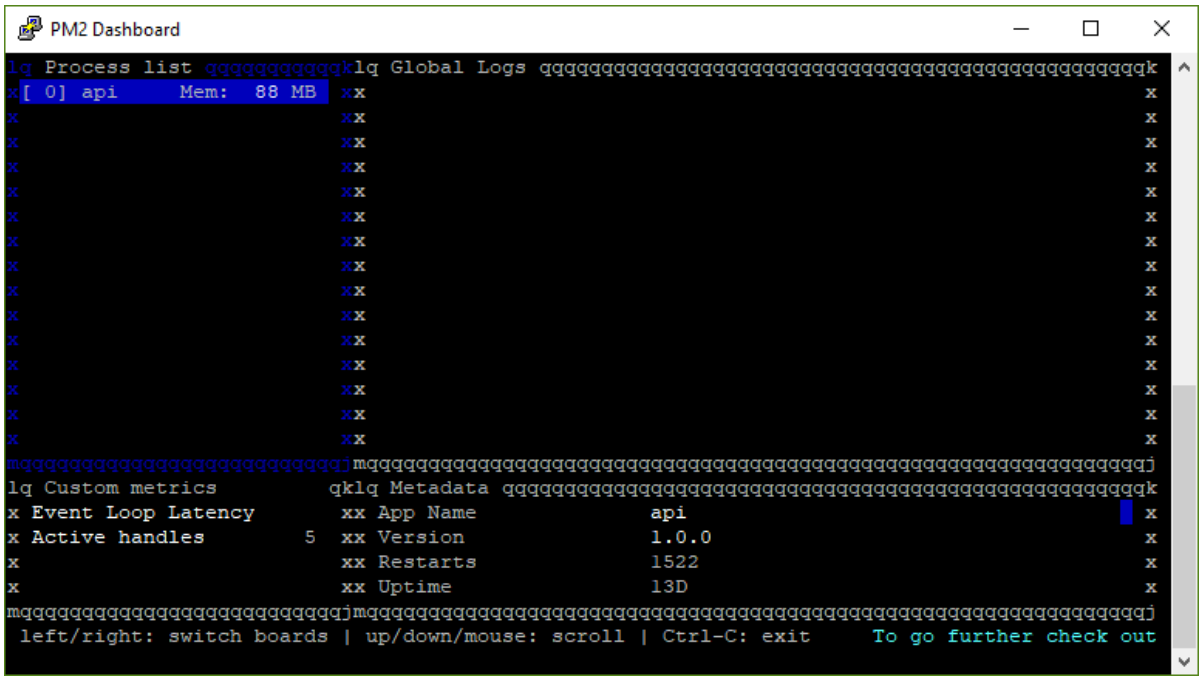
4.2 Palvelinympäristö

Vaatimusten mukaisesti palvelimeksi hankittiin virtuaalinen pilvipalvelin, johon käyttöjärjestelmäksi asennettiin Linux Ubuntu 18.04. Virtuaalinen pilvipalvelin takaa järjestelmän saatavilla pysymisen ja vähentää yrityksen taakkaa palvelimen ylläpidossa verrattuna fyysisen palvelimeen. Virtuaaliselle palvelimelle voidaan tarvittaessa lisätä resursseja, kuten muistia, suoritustehoa tai tallennustilaa. (13.)

4.2.1 Tukiohjelmistot

Tiedostojen jakoon ja web-rajapintojen paljastamiseen valittiin NGINX-verkkopalvelin. NGINX mahdollisti verkkorajapintojen konfiguroinnin, kuormantasauksen, tiedostojenjaon sekä TLS-suojauksen helposti. NGINX valittiin tiedostojen jakoon itse kehitetyn JavaScript-palvelimen sijasta nopean vasteajan, paremman suoritustehon sekä vakauden takia.

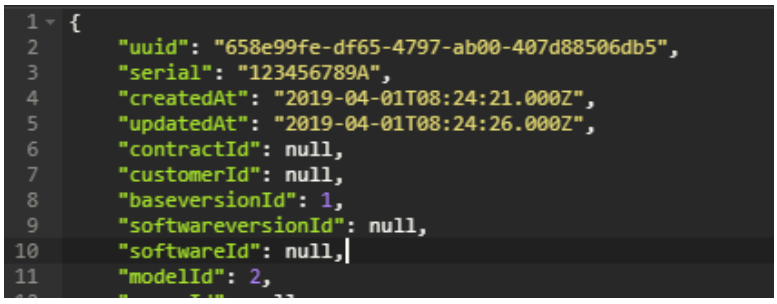
Palvelimelle asennettiin PM2-prosessinhallintaohjelmisto, joka vastaa REST-API:n käynnissä pysymisestä. Tilanteissa, joissa REST-API:n toiminta lakkaa, PM2 käynnistää sen uudelleen. Ohjelmistoon kuuluu myös konsolipohjainen monitoripaneeli (kuva 31), jonka avulla REST-API:n tilaa voidaan seurata. (14.)



KUVA 31. PM2 -prosessinhallintaohjelman monitoripaneeli

4.2.2 MySQL-tietokanta ja REST-API

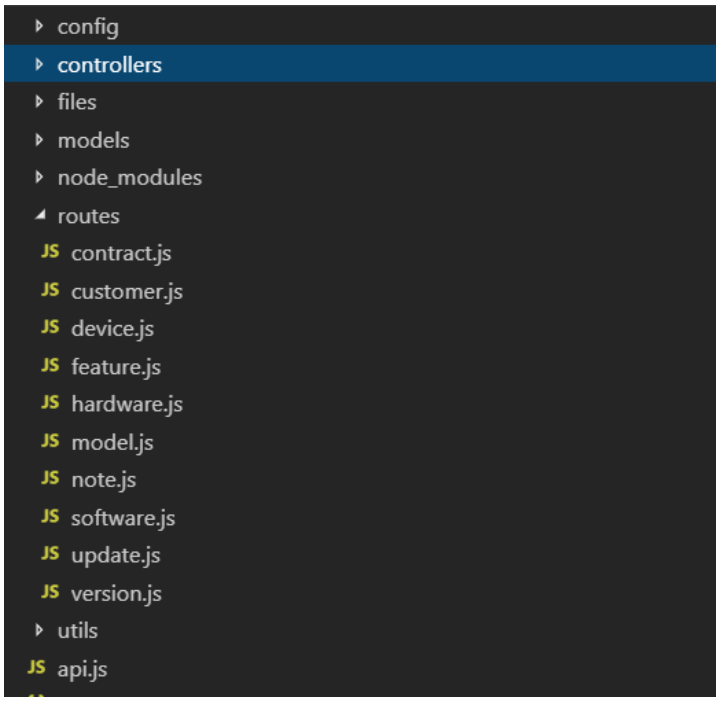
Kaikkien järjestelmän tietojen tallennusta ja säilytystä varten luotiin MySQL-tietokanta. Tietokannan kanssa kommunikoimiseen luotiin REST-API, jonka päävaatimus oli palauttaa sovelluksien tarvitsemaa data tietokannasta tai tallentaa sitä (kuva 32). Keskeisintä dataa ovat laitteiden tiedot, sekä niiden historia. Näiden tietojen perusteella yksittäiset laitteet voidaan tunnistaa Android-ohjelmistossa Nucu-alustojen päivitystä varten ja web-sovelluksessa asiakaspalvelutapauksissa. (15).



KUVA 32. Osa yksittäisen laitteen tiedoista

REST-API:n projektirakenteessa käytettiin MVC-mallia (kuva 33), jossa määritettiin mallit (engl. Models), reitit (engl. Routes) ja kontrollerit (engl. Controllers).

Mallit kuvaavat käsiteltävää dataa eli tietokantarakenteita ja niiden välisiä suhteita. Määritetyistä malleista luotiin tietokantarakenne automaattisesti Sequelizen avulla. Reitit ovat ”osoitteita” eli Internetiin avoimia rajapintoja, joiden kautta sovellukset voivat pyytää tarvitsemaansa dataa. Kontrollerit käsittelevät mallien sisältämän datan ja sisältävät kaiken logiikan, joka tarvitaan datan hakemiseen tai tallentamiseen sekä oikeaan muotoon saattamiseksi.

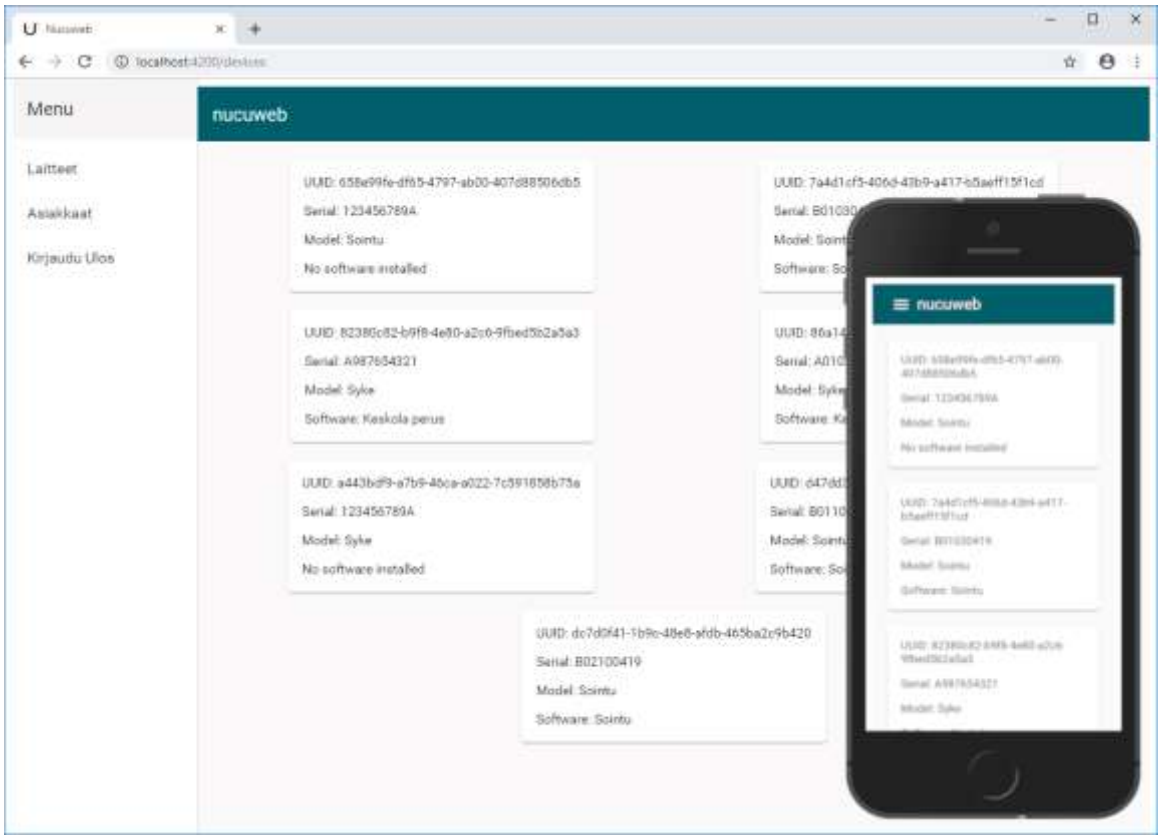


Kuva 33. Projektirakenne ja MVC Malli

MVC-mallin käytöllä varmistettiin API:n jatkokehitettävyys ja helppolukuisuus (16).

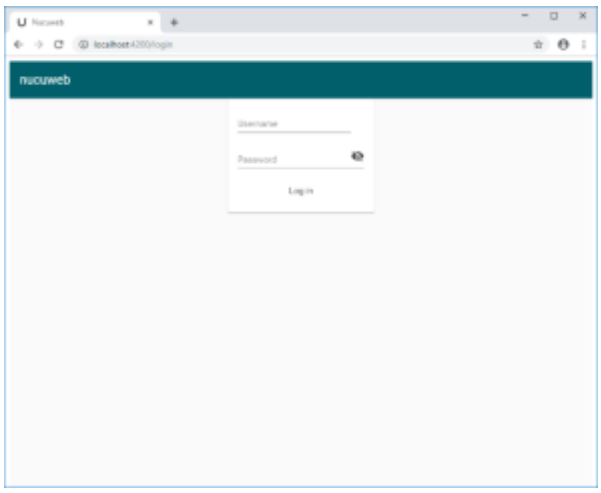
4.3 Web-sovellus

Keskeisin vaatimus web-sovellukselle oli laitteiden tietojen näyttäminen käyttäjälle helppolukuisessa formaatissa (kuva 34). Yksinkertaisen käyttöliittymän luomiseksi käytettiin Angular Material -komponenttikirjastoa, joka sisältää valmiiksi luotuja pohjia käyttöliittymäelementeille. Kirjasto sisältää myös valmiita väriteemoja, mutta projektia varten määriteltiin oma väriteema Nucu Oy:n vaatimusten mukaisesti.



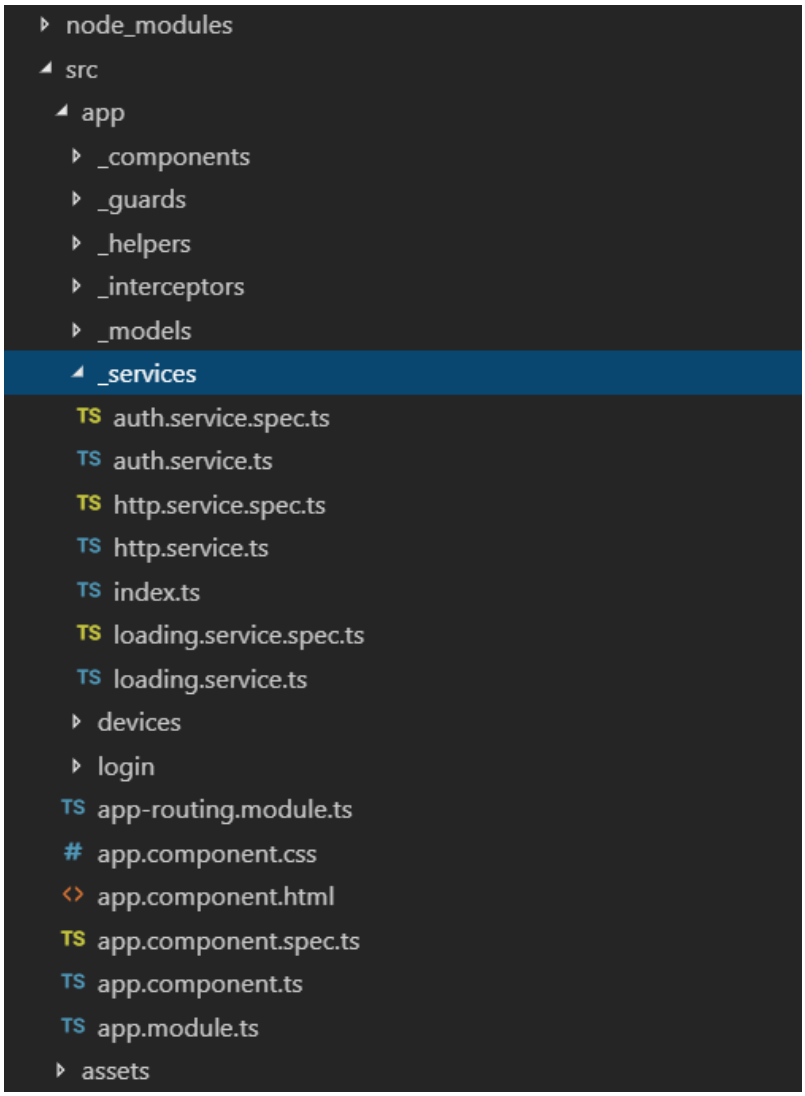
KUVA 34. Web-sovellus ja laitetietoja.

Sovellukseen luotiin pohja kirjautumistoiminnolle tulevaisuutta varten (kuva 35). Projektin päätösvaiheessa kirjautuminen oli vain simuloitu, mutta projektin kannalta oli tarpeellista, että simuloitu toiminnallisuus luotiin. Vaikka kirjautuminen ei ollut täysin toiminnassa, se luotiin hyvien tapojen mukaiseksi ja sitä voidaan käyttää tuotantokäytössä minimaalisin muokkauksin.



KUVA 35. Kirjautuminen

Web-sovellus luotiin Angularin käytänteiden mukaisella komponenttirakenteella. Toiminnoiltaan erityyppiset projektin osat sijoitettiin omiin kansioihinsa selkeyden varmistamiseksi (kuva 36).



KUVA 36. Web-sovelluksen projektirakenne

5 JÄRJESTELMÄN JATKOKEHITYS

Toteutettu järjestelmä suunniteltiin jatkokehitys huomioon ottaen. Tietyt järjestelmän osat jätettiin tarkoituksella jatkokehitykseen niiden asettamien vaatimuksien tai käytännön rajoitteiden takia. Suurimmat kaikkia järjestelmän osia koskevat ominaisuudet, jotka jäivät jatkokehitykseen, ovat

- käyttäjien tunnistus ja kirjautuminen
- käyttöliittymien yhtenäistäminen ja grafiikan suunnittelu
- tuotanto- ja kehitysympäristöjen eristäminen toisistaan
- laajamittainen testaus.

Tunnistuksen lisääminen järjestelmään on teknisestä näkökulmasta suhteellisen helppoa. Ongelmaksi kehitysvaiheessa nousikin tulevaisuuden käyttäjäryhmien määrittely. Luodut työkalut ovat tarkoitetut pääasiassa yrityksen sisäiseen käyttöön, mutta tulevaisuudessa ohjelmistoista voi olla ominaisuuksiltaan rajattuja versioita eri käyttäjäryhmille kohdistettuna. Tästä syystä käyttäjien tunnistuksen suunnittelu on työläs prosessi. Jos kirjautumistietoja kerätään yksittäisiltä käyttäjiltä, täytyy jo suunnitteluvaiheessa ottaa huomioon EU:n tietosuoja (17).

5.1 Android-sovellus

Android-sovelluksen jatkokehitys alkaa heti projektin päätyttyä. Yrityksen palveluvalikoimaan on suunnitteilla monenlaisia lisäominaisuuksia, joissa sovellus on tavalla tai toisella mukana. Suurin tulevaisuuden työ sovelluksen kanssa tulee kuitenkin olemaan laajamittainen laite- ja virhetestaus. Kehitysvaiheessa käytimme testeissä vain kourallista laitteita eri käyttöjärjestelmäversioilla. Käyttöliittymien toimivuudesta saadaan palautetta tulevaisuudessa, kun ohjelmisto on laajemmin käytössä. Käyttöliittymää parannellaan tulevaisuudessa saadun palautteen mukaan.

Android-sovellukseen tulevaisuudessa vaikuttava muutos on Nucu-alustoihin mahdollisesti asennettava Flash-muisti. Tässä tapauksessa tehdasasennusprosessi hieman muuttuu. Asian tullessa ajankohtaiseksi tarvittavat muutokset sovellukseen ohjelmistoarkkitehtuurin puolesta on mahdollista toteuttaa.

5.2 Web-sovellus

Web-sovellukselle on suunniteltu ominaisuuksia, jotka ovat tarpeellisia jo käyttöönottovaiheessa. Itse laitteiden listauksen lisäksi web-sovellukseen luodaan toiminnot asiakasyritysten ja laitteiden myynnissä tehtyjen sopimuksien lisäämiseksi. Näiden työkalujen avulla asiakaspalvelu helpottuu ja mahdollisuudet yksityiskohtaisemman tilastotiedon keräämiseksi paranevat.

5.3 REST-API ja tietokanta

Tietokannan tärkein lähitulevaisuuden kehitettävä ominaisuus on migraatiot. Migraatioilla tarkoitetaan tietokantaan tehtäviä muutoksia, joista voidaan palautua. Migraatiot toimivat relaatiotietokantojen versionhallintana. Tietokantarakenne täytyy myös päivittää tukemaan Android- ja web-sovellusten tulevia ominaisuuksia.

REST-API:lle ei ole määritetty tarkkoja ominaisuuksia tulevaisuudessa kehitettäväksi, mutta sen tarkoitus on tukea Android- ja web-sovellusten toimintoja. Sovelluksien ominaisuuksia suunnitellessa otetaan huomioon REST-API:lta vaadittavat ominaisuudet.

6 YHTEENVETO

Työn päätavoite oli luoda Nucu Oy:lle järjestelmä, jolla yritys pystyy tehdasasentamaan sekä päivittämään valmistamiaan Nucu-alustoja helposti. Laiterekisteriä varten täytyi luoda tarvittavat tietokanta- ja verkkoratkaisut alustojen perustietojen tallentamiseen ja tarkasteluun.

Opinnäytetyön tuloksena syntyi järjestelmä, joka sisältää Android-ohjelmiston, web-sovelluksen sekä palvelinohjelmiston. Android-ohjelmisto sisältää alustojen tehdasasennukseen, rekisteröintiin ja tunnistamiseen sekä malliversio-ohjelmiston asentamiseen ja päivittämiseen liittyvät toiminnallisuudet. Tietoliikenne Android-puhelimen ja Nucu-alustan välillä tapahtui käyttämällä USB-väylää. Langatonta laitetunnistusta varten luotiin QR-koodin luomiskäytänteet. QR-koodin lukemisominaisuudet sisällytettiin Android-sovellukseen.

Web-sovellus luotiin työkaluksi, jolla voidaan tarkastella rekisteröityjen laitteiden tietoja. Tarvittaessa web-sovelluksella on tarkoitus myös muuttaa ja päivittää rekisteröityjen laitteiden tietoja. Sovellus luotiin erityisesti asiakaspalvelun tueksi sekä yrityksen sisäiseen käyttöön tiedonkeräykseen. Web-sovellus on luotu modernilla Angular-ohjelmistokehyksellä.

Edellä mainittuja sovelluksia varten luotiin palvelinympäristö, joka sisältää REST-API:n, tietokannan sekä tiedostojen jakeluun tarvittavan ohjelmiston. Palvelinympäristö täyttää Android- sekä web-sovelluksen tarpeet tiedon ja tiedostojen välitykselle. Palvelinratkaisut suunniteltiin tulevaisuuden tarpeet huomioon ottaen.

Järjestelmää kehitettäessä otettiin huomioon myös yrityksen nykyiset käytänteet laitteiden käyttöönotossa ja toimituksessa. Käytänteitä suunniteltiin ja päivitettiin tulevaisuutta silmällä pitäen. Järjestelmä tukee ja helpottaa yrityksen päivittäistä toimintaa laitteiden käsittelyssä ja käyttöönotossa. Kuvaus järjestelmän osista ja niiden välisestä tietoliikenteestä on liitteessä 3.

Opinnäytetyölle asetetut tavoitteet saavutettiin ja suunnitellut ominaisuudet saatiin toimimaan asetetussa aikataulussa. Kokonaisuudessaan projekti onnistui hyvin.

Projektin aikana ainoaksi merkittäväksi ongelmaksi nousi eri Android-käyttöjärjestelmän versioiden USB-kommunikaatio ja niiden väliset erot. Kaikkia Android-sovelluksen ominaisuuksia ei pystytty testaamaan kaikilla vaatimusmäärittelyssä asetetuilla käyttöjärjestelmän versioilla. Kokonaisuus on kuitenkin testattu toimivaksi versiolla 6.0.1.

Android-sovellusta ei toistaiseksi voitu julkaista Nucu Oy:n asiakkaiden käyttöön. Kun ohjelmisto on todettu toimivaksi kaikilla vaatimusmäärittelyn mukaisilla Android-versioilla, voidaan se julkaista asiakkaiden käyttöön. Siihen asti ohjelmisto on ainoastaan yrityksen sisäisessä käytössä.

Opinnäytetyössä ei ollut mielekästä kuluttaa resursseja mittavaan testaukseen, joten laajamittainen testaus jätettiin tulevaisuuteen.

Projektin hyvään lopputulokseen ja onnistumiseen vaikutti hyvä ryhmätyöskentely. Projektin jäsenet ovat tehneet useampia projekteja yhdessä aiemminkin. Erinomainen projektin ohjaus sekä Nucu Oy:n tarjoama työympäristö ja jatkuva tuki edesauttoivat projektin sujuvaa edistymistä.

LÄHTEET

1. PPSHP. 18.6.2018. OYS TestLab yhdistää yritysideat ja sairaalan. Saatavissa: https://www.ppshp.fi/Sairaanhoitopiiri/Ajankohtaista/Pages/OYS-TestLab-yhdistaa-yritysideat-ja-sairaalan.aspx. Hakupäivä 24.2.2019.

2. Nucu.fi 2019. Saatavissa: https://www.nucu.fi. Hakupäivä 24.2.2019.

3. Direktiivi 2017/745/EU: Euroopan parlamentin ja neuvoston asetus lääkinnällisistä laitteista. Euroopan unionin virallinen lehti 5.5.2017. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32017R0745. Hakupäivä 4.4.2019.

4. What is Scrum? 2019. Scrum.org Saatavissa: https://www.scrum.org/resources/what-is-scrum. Hakupäivä 4.4.2019.

5. Trello.com, 2019. Saatavissa: https://trello.com/. Hakupäivä: 4.4.2019.

6. Kotlin Language Documentation, 2019. Kotlin. Saatavissa: https://kotlinlang.org/docs/kotlin-docs.pdf. Hakupäivä: 4.4.2019.

7. Android Studio, 2019. Android Developers. Saatavissa: https://developer.android.com/studio/. Hakupäivä: 4.4.2019.

8. Sequelize, 2019. Saatavissa: http://docs.sequelizejs.com/. Hakupäivä: 4.4.2019.

9. Angular Material 2019. Material Design components for Angular. Saatavissa: https://material.angular.io/. Hakupäivä: 02.04.2019.

10. Postman, 2019. Saatavissa: https://www.getpostman.com/. Hakupäivä 4.4.2019.

11. Introduction to Activities, 2019. Android Developers. Saatavissa: https://developer.android.com/guide/components/activities/intro-activities. Hakupäivä: 4.4.2019.

12. Google Mobile Vision, 2019. Mobile Vision. Saatavissa: https://developers.google.com/vision/. Hakupäivä: 4.4.2019.

13. Linux Ubuntu, 2019. Ubuntu Suomi. Saatavissa: https://www.ubuntu-fi.org/. Hakupäivä 4.4.2019.

14. PM2 Overview. Saatavissa: https://pm2.io/doc/en/runtime/overview/. Hakupäivä 4.4.2019.

15. MySql, 2019. Oracle. Saatavissa: https://www.oracle.com/technetwork/database/mysql/index.html. Hakupäivä: 4.4.2019.

16. Simple Example of MVC (Model View Controller) Design Pattern for Abstraction, 8.4.2008. Saatavissa: https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design. Hakupäivä: 4.4.2019.

17. Direktiivi 2016/679/EU: Euroopan parlamentin ja neuvoston asetus luonnollisten henkilöiden suojelusta henkilötietojen käsittelyssä. Euroopan unionin virallinen lehti 4.5.2016. https://eur-lex.europa.eu/legal-content/FI/ALL/?uri=CELEX:32016R0679. Viitattu: 4.4.2019.

18. ISO/IEC 18004:2015: QR Code bar code symbology specification, 2015. Saatavilla: https://www.iso.org/standard/62021.html. Hakupäivä: 4.4.2019.

19. Merkintöjen testaus ja kehitys. Cajo Technologies. Saatavilla: https://cajotechnologies.com/fi/sovellukset/merkintojen-testaus-ja-kehitys/. Hakupäivä: 4.4.2019.

Alustan perusrakenne

A) NucuSense-pohjalevy. Pohjalevy värisee äänen tahdissa, luoden kuulo- sekä tunto-aistimuksia.

B) Patja. Pohjalevyn päälle asetettava pehmuste.

C) Reunaosion pehmuste.

D) Vaihdettava kangaspäällyste. Sairaalakäytössä materiaalina on sininen sairaalakangas.

E) Pohjalevyyn integroitu hallintapaneeli, jolla Nucun toimintoja ja liitäntöjä hallitaan

Langattomassa laitetunnistuksessa käyttöön valittiin QR-koodi, koska se on ISO standardoitu (18) ja sen käyttö osana laitetunnistus- ja rekisteröintiprosessia oli suhteellisen suoraviivaista. QR-koodi sisältää tunnistetiedot Nucu-alustasta, joiden perusteella tarkemmat tiedot voidaan tarvittaessa hakea palvelimelta tai rekisteröinnin yhteydessä uusi laite rekisteröidä tietokantaan.

Suurin tekninen haaste QR-koodin käytössä oli selvittää, minkälaisella tekniikalla se voitaisiin parhaiten lisätä Nucu-alustojen puiseen pohjalevyyn. Huomioonotettavaa oli QR-koodin luonnin helppous ja nopeus tuotantoprosessin aikana, hygieenisyys, miltä koodi näyttää ja kuinka helppo mahdollisesti vioittunut koodi on korvata uudella. Yhtenä mahdollisena ratkaisuna pohdittiin lasermerkinnällä tehtävää QR-koodin polttamista. Nucu Oy lähetti yhden puisen alustan testausta varten lasermerkintäjärjestelmiä valmistavan Cajo Technologies:in testilaboratorioon (19).

1. Laserilla tehdyt testimerkinnät

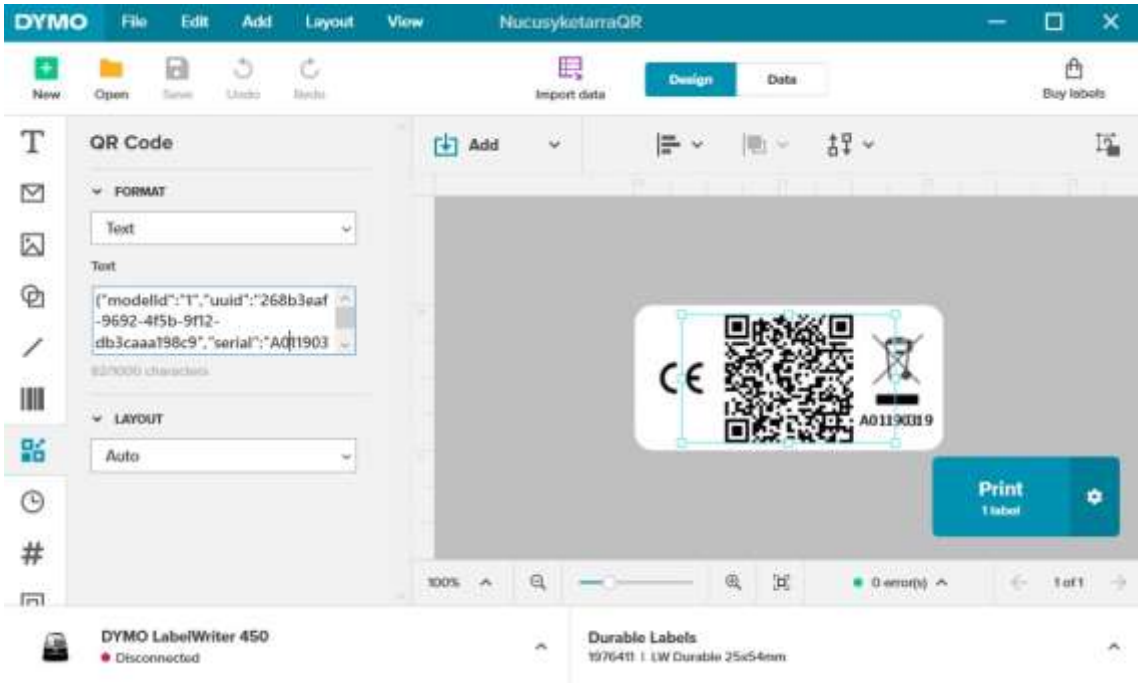
Lasermerkinnän testitulokset olivat hyviä. QR-koodit pystyttiin lukemaan ja yleisesti kaikki laserpoltetut merkinnät näyttivät hyviltä. Ongelmaksi muodostui kuitenkin tekniikan hygieenisyys ja mahdollinen koodin uusinnan vaikeus sen vioittuessa. Lasermerkintä poltti puualustan lakatun pinnan läpi selvästi tuntuvat urat, joihin voi helposti jäädä erilaisia epäpuhtauksia. Ainakin toistaiseksi lasermerkintää ei päädytty käyttämään QR-koodien lisäämiseen Nucu-alustoihin.

Lopulta päädyimme käyttämään tulostettavaa laadukasta tarraa, joka liimataan alustan puisen pohjan etuosaan. QR-koodin lisäksi tarraan lisättiin muitakin merkintöjä: CE-merkki, alustan sarjanumero sekä RoHS-merkki. Tarrojen tuotantoa varten Nucu Oy

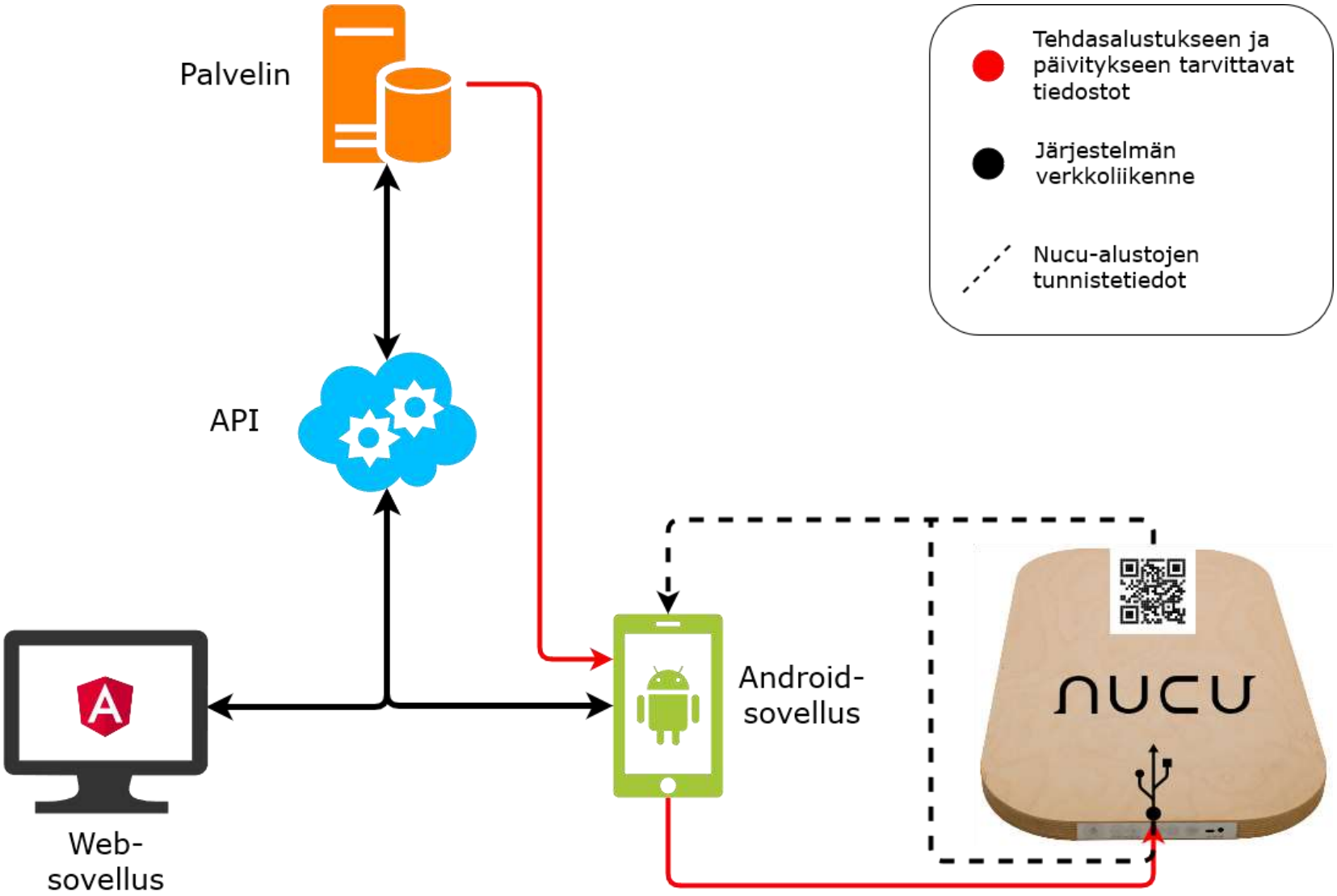
hankki DYMO-tarratulostimen. Tarratulostimen mukana tulevalla ohjelmistolla pystyimme luomaan ja tulostamaan tarrat.

2. DYMO-tarratulostin

QR-koodin sisältämän JSON-muotoisen tekstisisällön loimme manuaalisesti. Sisältö ja logiikka, jolla teksti luodaan, on suunniteltu niin, että myöhemmin pystymme automatisoimaan sen luomisen.



3. DYMO-ohjelmisto tarrojen luomista ja tulostamista varten


Palvelin
API
Web-
sovellus
Android-
sovellus
nucu
Tehdasalustukseen ja
päivitykseen tarvittavat
tiedostot
Järjestelmän
verkkoliikenne
Nucu-alustojen
tunnistetiedot