Arttu Perämäki, Lilli Häyhänen, Linus Willner, Teemu Olkkonen

# Device Patrol -laitehallintaohjelmiston kehitys

Raportti 16.12.2021

Ohjelmistotuotantoprojekti 2 -opintojakso

Syventävät ammattiopinnot

Tieto- ja viestintätekniikka

TVT20K-O1

Metropolia

## Sisällys

# 1 Johdanto

Tämän projektin tarkoituksena oli perehtyä ketterien ohjelmistotuotannon muotojen työkaluihin ja työskentelytapoihin projektin hallinnassa ja ryhmän toiminnan ohjaamisessa. Tarkemman tarkastelun kohteena oli erityisesti scrum-menetelmä. Projektityöskentely kesti kaksi kahdeksan viikon mittaista opintojaksoa, jotka oli jaettu seitsemään noin kaksi viikkoa kerrallaan kestävään sprinttiin, joista jokaisesta vastuussa oli yksi projektiryhmän jäsen kukin vuorollaan.

Ketteriin menetelmiin tutustumisen lisäksi projektin tarkoituksena on ollut tuottaa Java-ohjelmointikielellä työpöytäsovellus, jossa on JavaFX-kirjastolla toteutettu käyttöliittymä ja joka käyttää tietokantaa. Opintojaksojen edetessä mukaan tuli myös käyttäjäkeskeinen näkökulma ja projektityö lokalisoitiin oman projektimme tapauksessa suomen, ruotsin ja englannin kielelle.

Projektin yhteydessä hyödynnettiin ja opeteltiin myös muita, osalle ryhmän jäsenistä uusia teknologioita, kuten Apache Maven, Docker, PostgreSQL, Hibernate, Jenkins ja kehitystyön tehostamiseksi käytettiin myös Javan Lombok-kirjastoa. Kehitystyötä tehtiin Eclipsen sijasta IntelliJ Idea ohjelmointiympäristössä projektiryhmän henkilökohtaisten mieltymysten mukaan. Projektin hallinnassa käytettiin Nektion -työkalua ja projektin mallintamiseen ja kaavioiden luomiseen käytettiin ERDPlus ja StarUML -mallinnustyökaluja.

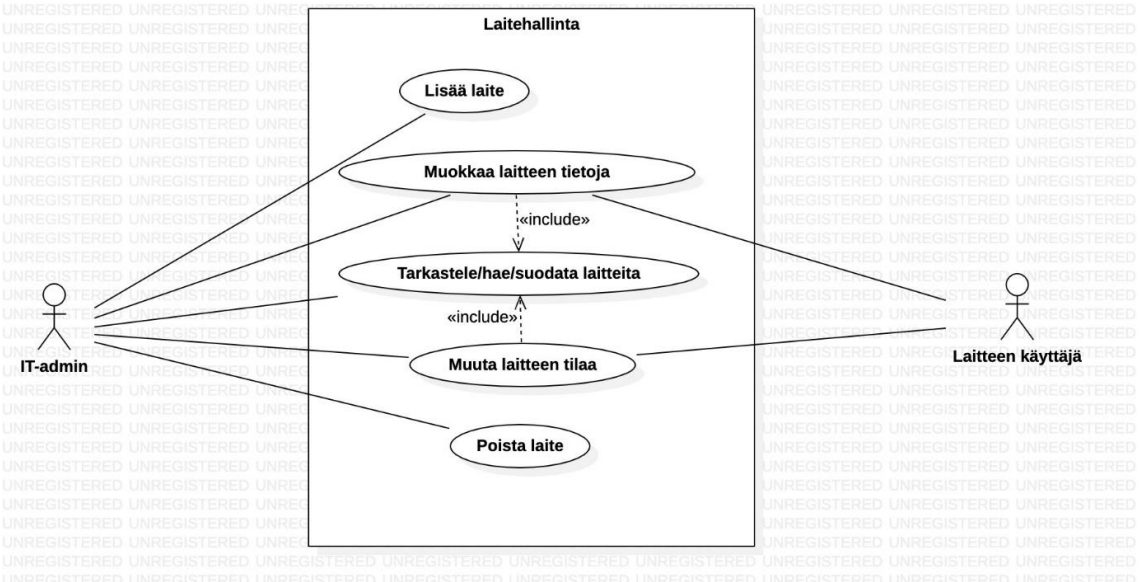
Ryhmä päätti ryhtyä toteuttamaan sovellusta, jolla pystyy pitämään kirjaa tietoteknisistä päätelaitteista, niiden käyttäjistä ja tiloista, sekä lisäämään, poistamaan ja muokkaamaan laitteiden tietoja. Sovellus on tarkoitettu erityisesti yritysten ja organisaatioiden IT-henkilöstön käyttöön tukemaan kenties useissa eri sijainneissa sijaitsevien päätelaitteiden tietojen tarkastelua, tallennusta ja hallinnointia yhteen sovellukseen ja keskitettyyn tietokantaan sijainnista riippumatta. Tämä projektin loppuraportti koostuu kahdesta osasta, joista ensimmäisessä perehdymme ensin tarkemmin ohjelmiston suunnitteluun, kehitystyöhön ja teknisen toteutuksen kuvaamiseen. Toisessa osassa luvussa 7 on ohjelmiston käyttöliittymän käyttöohje.

# 2 Projektin visio

Projektin visiota määritellessä kohderyhmäksi valittiin yksityisen, julkisen ja kolmannen sektorin toimijat ja sovelluksen varsinaisina käyttäjinä näiden yhteisöjen tietoteknisistä ratkaisuista ja kalustosta vastaavat henkilöt. Listattavien päätelaitteiden loppukäyttäjillä ei ole tarvetta päästä sovellukseen, vaikka järjestelmästä tehtiinkin sen verran yksinkertainen, että käyttö onnistuu ohjeiden avulla myös henkilöltä, joka ei ole tietotekniikan ammattilainen. Erityisen pienten organisaatioiden, kuten vaikkapa yhdistysten tapauksessa organisaation sisällä ei ehkä ole lainkaan tietoteknisistä ratkaisuista vastaavaa henkilöä.

## 2.1 Ohjelmiston käyttötapaukset

Sovelluksen käyttötapaukset ovat esitettynä kuvassa 1 näkyvässä käyttötapauskaaviossa. Yrityksen IT-vastaava on sovelluksen ainoa tosiasiallinen käyttäjä. Laitteen käyttäjä ei itse käytä sovellusta vaan on mukana prosessissa ainoastaan vastaanottaessaan ja palauttaessaan tietoteknisen päätelaitteen kokonaan tai huollon yhteydessä tai kun hänen omat tietonsa muuttuvat. Laitteista vastaava henkilö voi lisätä laitteen, muokata sen tietoja, hakea laitteita tai tarkastella listaa laitteista, muuttaa laitteen tilaa ja poistaa laitteita järjestelmästä.

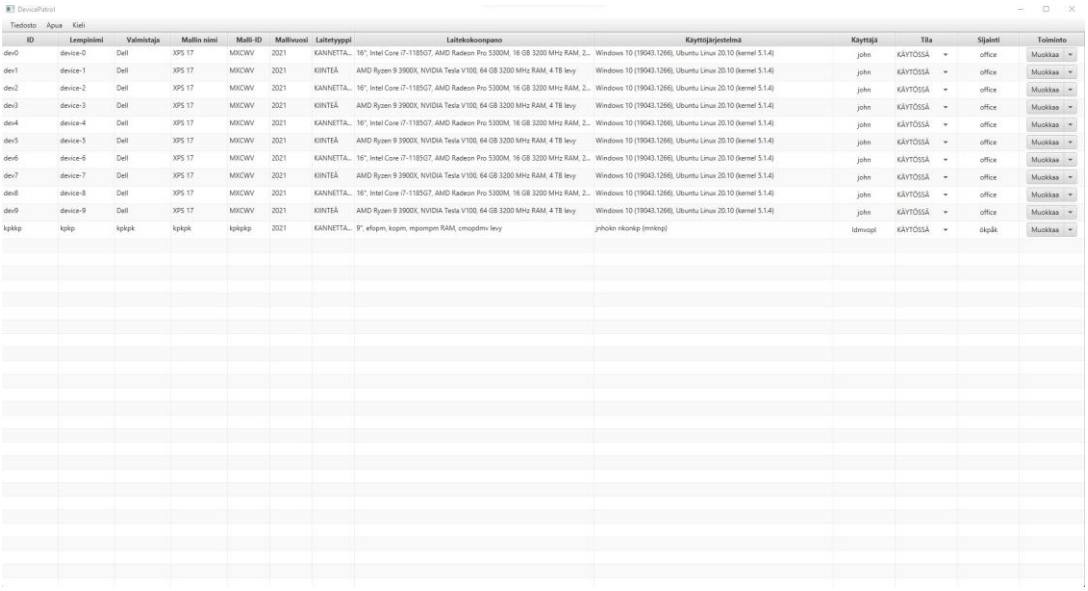



Kuva 1. Projektin käyttötapauskaavio.

## 2.2 Suunnitteluvaiheessa määritelty ohjelmiston perusrakenne

Käyttöliittymään päätettiin tehdä päänäkymä, joka listaa kaikki tietokannassa olemassa olevat laitteet tietoineen näytölle. Tietokantaan tallennetaan laitteen tekniset tiedot, tiedot käyttäjästä, toimipisteestä sekä laitteen aktiivinen statustieto, kuten onko laite jollakulla käytössä, vapaana, rikki tai huollettavana.

Teknisiä tietoja ovat muun muassa valmistaja- ja mallitiedot, tiedot laitteen muistista, prosessorista, näytönohjaimesta, käyttöjärjestelmästä ja se, onko kyseessä kannettava vai pöytätietokone. Kuvassa 2 näkyvässä päänäkymässä voi myös poistaa laitteita ja päästä muokkaamaan olemassa olevia laitteita.




Kuva 2. Device Patrol -sovelluksen päänäkymä, johon tietokannasta löytyvät tiedot laitteista tulevat näkyviin.

Päänäkymän valikosta avautuu ponnahdusikkunoita, joissa voi lisätä uusia laitteita, laitekokoonpanoja, käyttöjärjestelmiä, käyttäjiä ja sijaintitietoja, sekä muokata jo olemassa olevia edellä mainittuja tietoja. Projektin suunnitteluun käytettiin reilusti aikaa ja projektin lähestyessä loppuaan on havaittavissa, että visiosta ei olla työskentelyn aikana juurikaan poikettu vaan visio on toiminut koko ajan työskentelyä ohjaavana suunnannäyttäjänä.

# 3 Toiminnallisuuksien suunnittelu, tekninen kuvaus ja toteutus

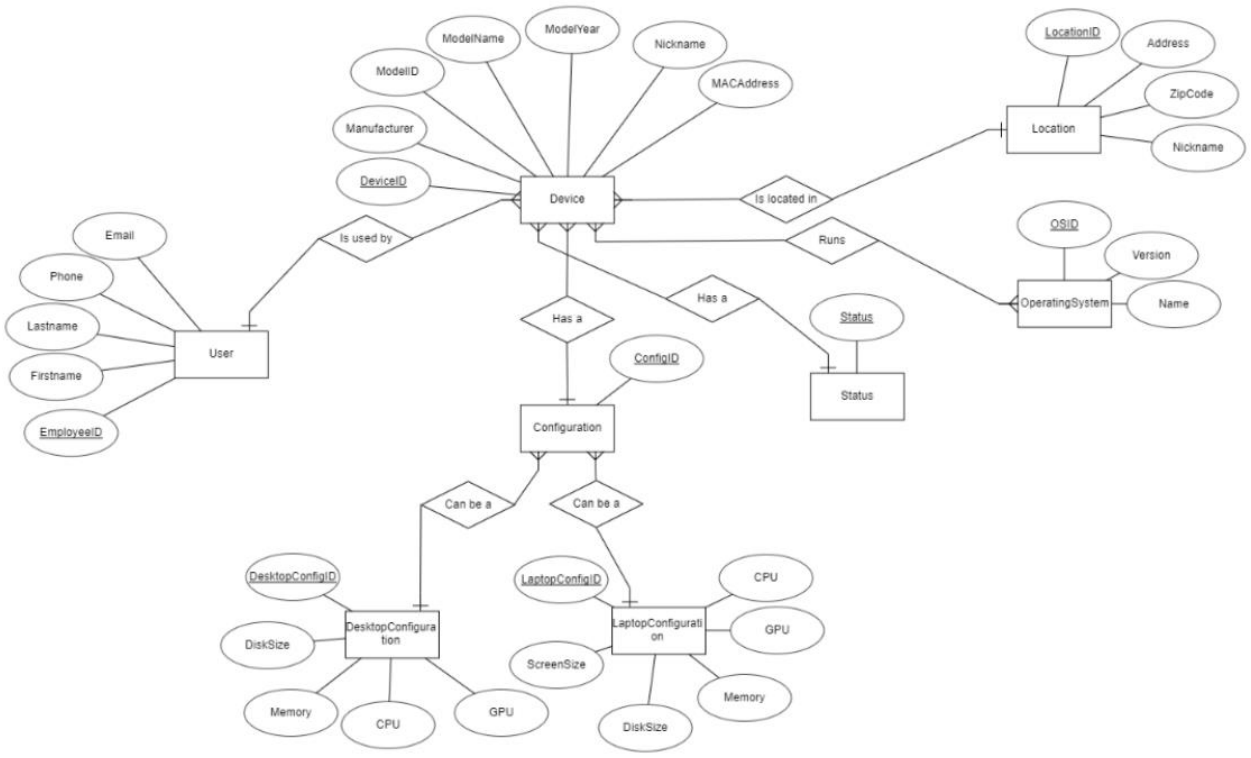
Toiminnallisuuksien ja käyttöliittymän suunnittelu tehtiin yhteistyössä projektiryhmän kesken ja jaettiin projektinhallintajärjestelmä Nektioniin tarinoiksi ja tehtäviksi ja niiden tärkeyttä ja toteutusjärjestystä arvioitiin ja suunniteltiin pitkälti jo ennalta, samoin kuin niihin käytettävää aikaa. Ennen sprinttien aloittamista suunniteltiin, mitä toiminnallisuuksia kussakin sprintissä työstetään. Aika-arvioissa ei suinkaan läheskään aina pysytty, vaan toisinaan kaikkia sprinttiin aiottuja asioita ei ehditty tehdä. Tarkempi aika-arvioiden esittäminen vaatii paljon kokemusta sovelluskehittämisestä ja työssä käytettävien teknologioiden tuntemista, jotta yllätyksellisiä elementtejä ei ilmene kehitystyötä jarruttamassa.

Tietokantaa ja projektin rakennetta suunniteltiin ja havainnollistettiin monin eri keinoin ja erilaisten kaavioiden avulla ohjelmistotuotantoprojekti 1:sen kanssa samaan aikaan käydyllä kuvaus – ja mallintamismenetelmien opintojaksolla. Kurssilla mallinettiin muun muassa projektityön tietokantaa ja ohjelmiston rakennetta eri näkökulmista monin erilaisin kaavioin.

## 3.1 Tietokannan suunnittelu ja kuvaaminen

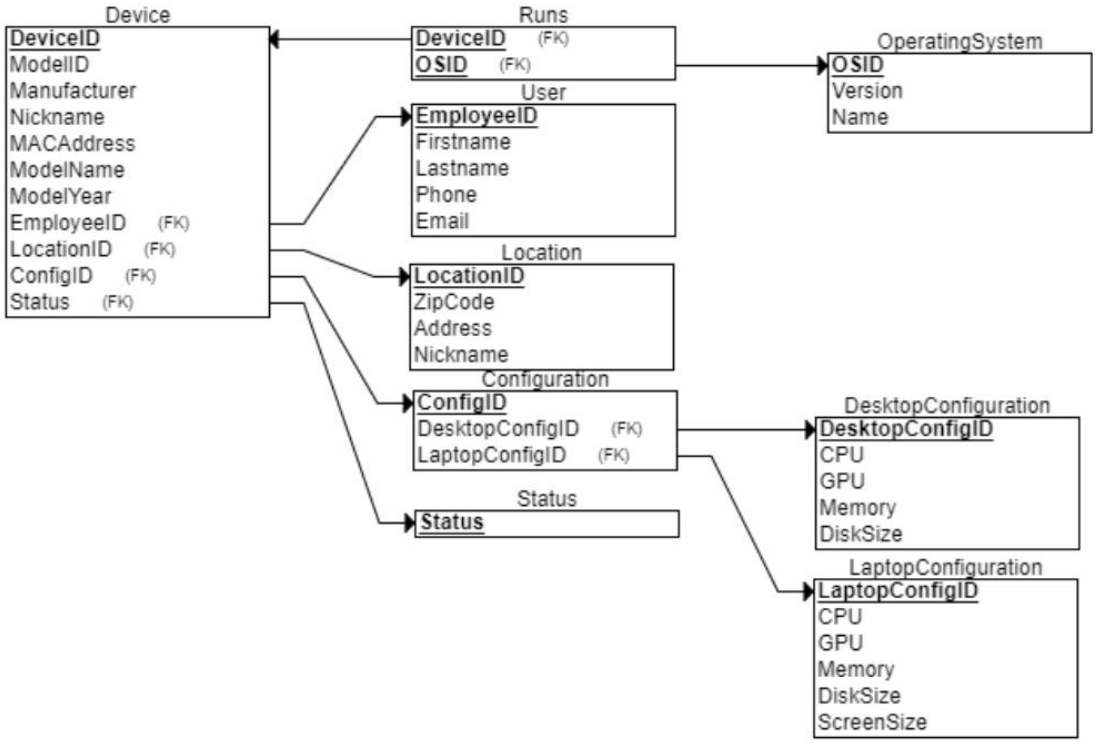
Melko varhaisessa vaiheessa kehitystyötä ohjelman tietokannan rakennetta ja ohjelmiston tarvitsemia tietoja lähdettiin pohtimaan ja suunnittelemaan ER-kaavion (Entity Relationship Model) ja relaatiotietokantakaavion luomisen avulla osana kuvaus- ja mallintamismenetelmien kurssia.

Kuvan 3 ER-kaaviossa on kuvattuna, mitä kaikkia tietoja yksilötyypit Device, User, Configuration, DesktopConfiguration, LaptopConfiguration, Status, Location ja OperatingSystem tarvitsevat ja millaisia ovat niiden väliset keskinäiset yhteydet. Kaaviota työstettiin huolella ja muokattiin niin kauan, että kokonaisuudesta tuli ryhmän mielestä järkevä ja siihen kootut tiedot olivat kaikki tarpeellisia sovelluksen toiminnan ja rakenteen kannalta, myös mahdollisesti tulevaisuudessa tapahtuvan jatkokehittämisen näkökulmasta.




Kuva 3. ER-kaavio ohjelmiston datasta.

ER-kaavion pohjalta laadittiin kuvaus- ja mallintamismenetelmien kurssille harjoitustyönä myös kuvassa 4 näkyvä relaatiotietokantakaavio, jossa samat tiedot on esitetty hieman erilaisessa muodossa. Kaaviossa näkyvät edellä mainitut yksilötyypit tietokantatauluina, joissa on nähtävissä kussakin taulussa näkyvät tiedot, jotka ohjelmisto tallentaa tietokantaan ja myös taulujen väliset suhteet tarvittavine liitostauluineen. Molemmat kuvaukset tietokannan rakenteesta auttoivat tarvittavien tietojen pohtimisessa, suunnittelutyössä ja toteutuksessa tietokannan rakenteen ymmärtämisessä ja ohjasivat kehitystyötä.




Kuva 4. Projektin relaatiotietokantakaavio.

## 3.2 Toiminnallisuuksien ja työn laajuuden suunnittelu

Tärkeimmiksi toiminnoiksi ohjelmistossamme valikoituivat yksimielisesti laitteiden listaaminen ja niiden lisääminen, poistaminen sekä muokkaaminen. Ilman näitä operaatioita ei visiomme mukaista sovellusta voisi olla olemassa. Tietokannan rakenteen vuoksi myös liitosolioiden, esimerkiksi käyttäjien ja toimipisteiden tietoja pitää pystyä käsittelemään, jotta kokonaisuus olisi järkevä sovelluksen käyttötarkoituksen ja luonteen puolesta.

Joidenkin aikataulu- ja kommunikaatiohaasteiden vuoksi käyttöliittymän kehitystyössä ei päästy aina etenemään sillä tahdilla, kuin oli alun perin suunniteltu. Toisinaan tehtäviä järjesteltiin uudelleen ja kustakin tehtävästä vastaavaa henkilöä saatettiin vaihtaa sprinttien loppupuolella sen turvaamiseksi, että projektissa päästään etenemään. Tavoite oli päästä siihen pisteeseen, että projektityöskentelyn lopuksi kasassa on toimiva ja demokelpoinen tuote, joka on kunnolla testattu, täyttää sille asetetut vaatimukset ja jota päästään esittelemään hyvillä mielin muille ryhmille.

Muita pohtimiamme mahdollisia toiminnallisuuksia olivat muun muassa erilaiset laitteiden haku-, järjestely- ja suodatustoiminnot, joissa tietokantaa voidaan hyödyntää entistä tehokkaammin. Myös mahdollisuudesta laajentaa mobiililaitteisiin keskusteltiin, samoin kuin ulkoasun parantamisesta, mikäli aikaa jää. Näiden toimintojen toteuttaminen jäi kuitenkin vielä toistaiseksi tekemättä aikataulun toimiessa ryhmän kehitystyötä vastaan. Edellä mainitut toiminnallisuudet kuitenkin toimivat hyvinä jatkokehitysideoina tulevaisuuden varalta.

### 3.3 Tuotteen testaaminen

Osana projektityöskentelyä, kuten kaikessa sovelluskehittämisessä pitäisi, oli tuotteen testaaminen laadukkaan sovelluksen valmistumisen takaamiseksi. Suurin osa testeistä on manuaalitestejä, vaikka myös automaattista testaamista käytettiin. Kaikkea ohjelmistossamme, esimerkiksi UI-kontrolleriluokkia (user interface) ei ole järkevästi mahdollista testata automaattitestein niiden kompleksisuuden vuoksi.

Testaamisessa olivat apuna Apache Maven, sekä Jenkins, jotka tarjoavat tukea projektien jatkuvalle integraatiolle ja helpottavat ryhmätyöskentelyä ja useiden kehittäjien koodin yhdistymistä järkevällä tavalla. Kehitystyössä pohdimme mitä on järkevää testata ja mahdollisimman suuri testikattavuusprosentti ei ollut suurin prioriteettimme, halusimme testien määrän sijaan keskittyä testaamaan järkevästi. Lopullisessa versiossa testikattavuus oli luokkien osalta noin 60 prosenttia ja viivakattavuus noin 30 prosenttia.

Täysin ongelmitta ei testaamisenkaan kanssa selvitty ja Jenkinsin, sekä Mavenin kanssa törmättiin välillä pieniin takaiskuihin, joiden selvittely saattoi paikoin hidastaa työskentelyä ja viedä aikaa muulta kehitystyöltä. Jenkins ei ollut alun perin täysin tuttu osalle projektiryhmäläisistä, joten sen toimintaa oli hyvä nähdä, joskin nykyään monissa versionhallintatyökaluissa on myös tuki jatkuvalle integraatiolle ja näiden käyttö yleistyy työelämässä jatkuvasti. Tästä syystä myös niiden käytön salliminen projektityöskentelyssä voisi tarjota opiskelijalle hyviä eväitä tulevaisuuteen.

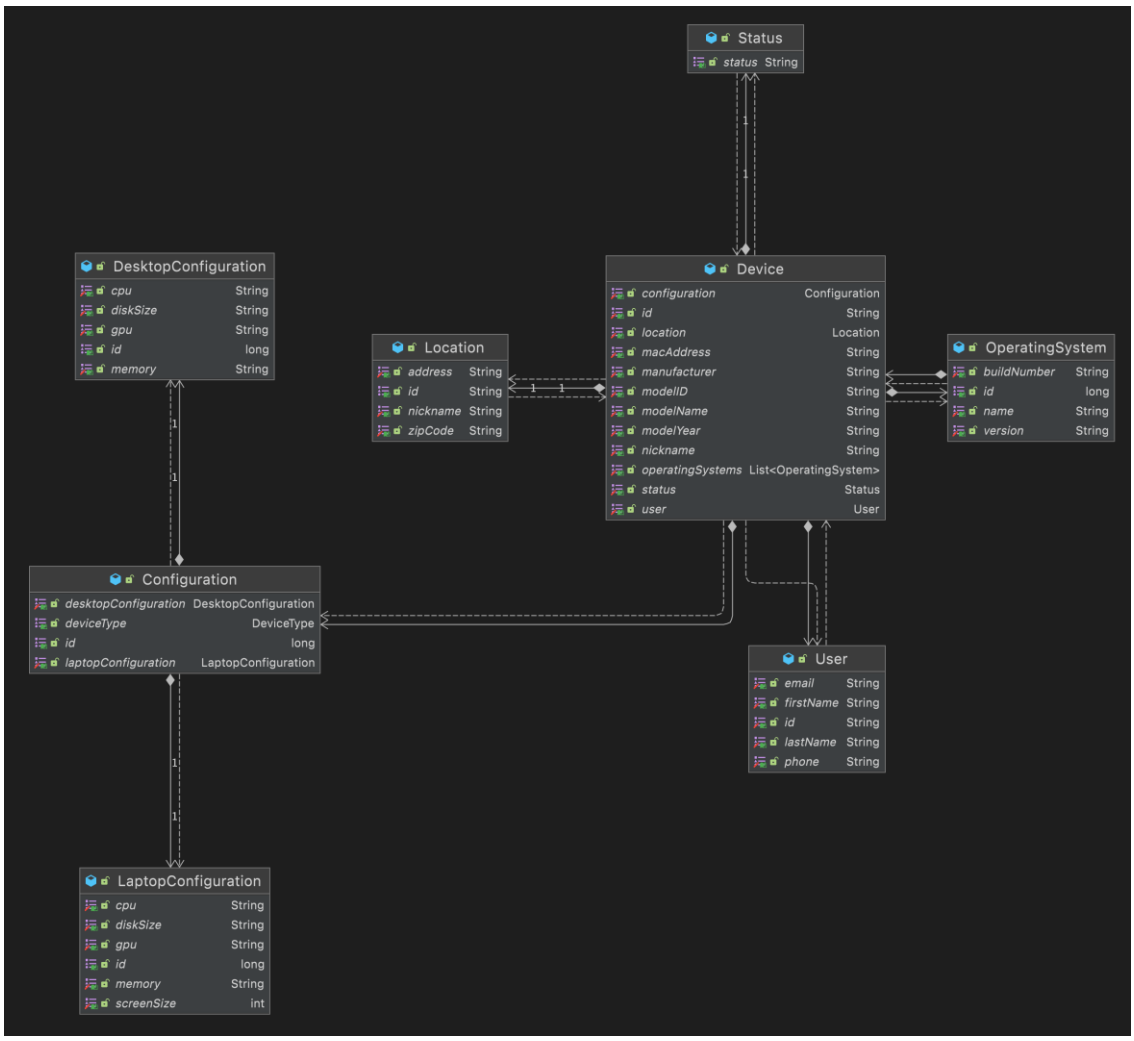
# 4 Kuvaus projektin arkkitehtuurista

Projektia mallinnettiin projektikurssin ohessa käynnissä olleella kurssilla kuvaus- ja mallintamismenetelmät, jossa projektista, sen rakenteesta ja arkkitehtuurista tehtiin harjoitustöinä useita erilaisia kaaviokuvia ERDPlus ja StarUML työkaluilla. Tässä luvussa on esiteltynä osa kurssilla aikaansaaduista tuotoksista, jotka ovat keskeisiä kuvaamaan projektimme rakennetta.

## 4.1 Ohjelman toiminnan kannalta keskeiset luokat

Projekti toteutettiin MVC-mallin (model, view, controller) mukaisesti ja sen luokkakaavio koostuu kolmesta kokonaisuudesta, jotka ovat DTO (Data Transfer Object), kontrollerit ja DAO (Data Access Object). DTO:n alla on kuvattuna luokka Device, joka sisältää kunkin laitteen kaikki perustiedot. Luokan Configuration alla on aliluokat LaptopConfiguration ja DesktopConfiguration, jotka määrittelevät, mitä kaikkia teknisiä tietoja kuuluu kullekin laitetyypille. Jos sovellusta haluttaisiin laajentaa vaikkapa mobiililaitteisiin, voisi Configuration-luokan alle tehdä uusia laitekokoonpanoja kullekin laitetyypille relevanteilla tiedoilla. Luokka Status kertoo laitteen tilan, eli onko se käytössä, vapaana, rikki tai huollossa.

OperatingSystems on oma luokkansa, sillä kahdella muuten keskenään samankaltaisella laitteella voi olla eri käyttöjärjestelmä tai yhdellä yksittäisellä laitteella voi olla asennettuna useita eri käyttöjärjestelmiä käyttäjän tarpeen mukaan. Location -luokka kertoo toimipisteen, jonka alle laite on rekisteröity. Tämä mahdollistaa sen, että tuote skaalautuu hyvin kaikenkokoisten organisaatioiden käyttöön, oli kyseessä sitten pienempi yhdistys tai monikansallinen yritys, jossa on suuri määrä eri toimipisteitä. Luokka User pitää sisällään tarvittavat tiedot laitteen käyttäjästä. Graafinen kuvaus yllä esitellystä DTO-kotonaisuudesta näkyy kuvassa 5.

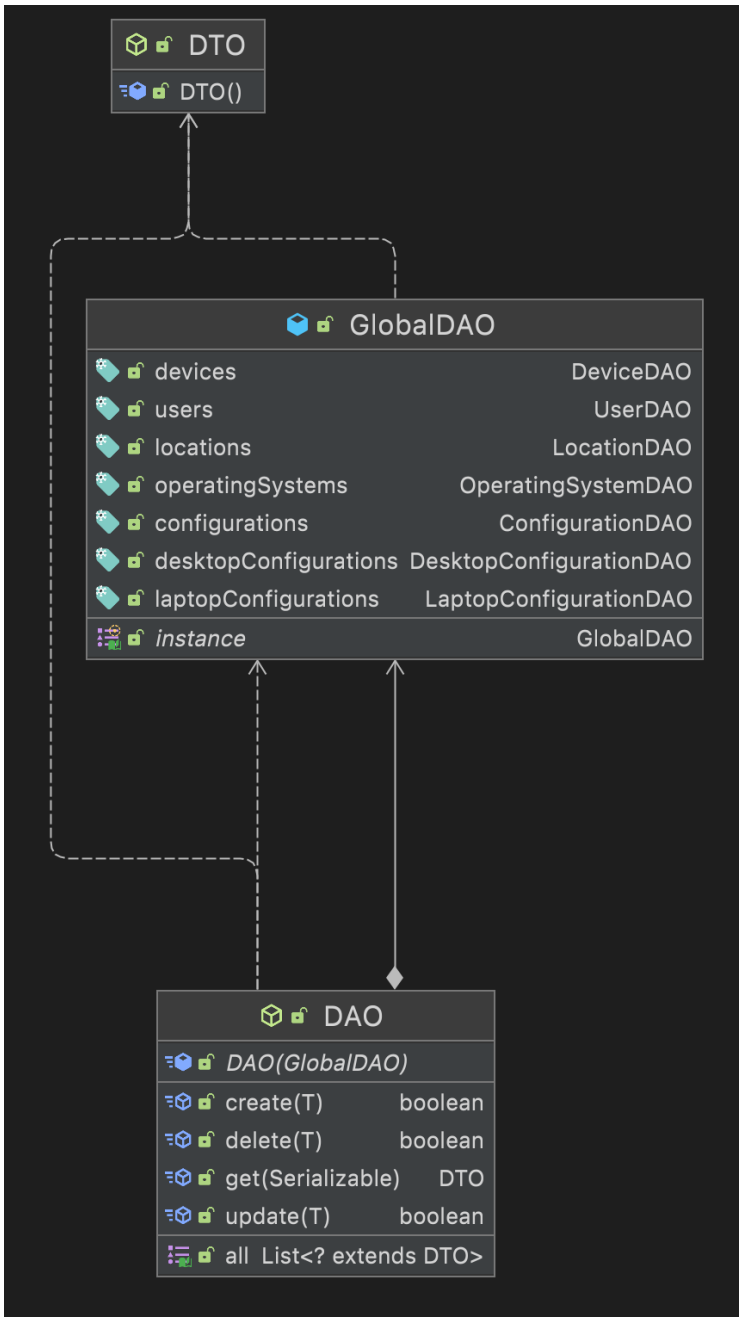



Kuva 5. Projektin DTO-kokonaisuutta kuvaava luokkakaavio.

Projektin suunnitellessa todettiin, että tietoja on pystyttävä hakemaan, lisäämään, päivittämään ja poistamaan. Kontrolleriluokkia tehtiin jokaiselle näkymälle, jotka ovat osana ohjelmistoa. Näkymät luotiin SceneBuilderin avulla JavaFX-kirjastoa käyttäen. Näillä aikaan saadut fxml-tiedostot kuvailevat eri näkymien rakenteen ja ulkoasun. MainViewController huolehtii logiikasta, jolla päänäkymään saadaan listattua laitteet, sekä generoidaan valikoiden ja nappien toimintaperiaate, jotta listanäkymässä pystytään siirtymään helposti näkymiin, joissa laitteiden tietoja on mahdollista muokata ja poistaa.

DeviceEditorController huolehtii uusien laitteiden lisäämisestä ja olemassa olevien laitteiden tietojen päivittämisestä. Liitosolioiden lisäämisestä ja tietojen päivittämisestä vastaavat kutakin liitosoliota varten luodut kontrollerit, joita ovat HardwareConfigurationEditorController laitekonfiguraatioille, OperatingsystemEditorController käyttöjärjestelmille,

UserEditorController käyttäjille, sekä LocationEditorController sijainneille. Lisäksi kontrolleriluokka PrettyDeviceViewerController huolehtii päänäkymän päälle kutsuttavasta sivupalkista, josta on mahdollisuus nähdä yksittäisen laitteen tiedot tarkemmin ja helpommin luettavassa muodossa. Liitosolioiden päivittämistä varten tehtiin myös ManagementViewController, joka generoi taulun tietokannassa olemassa olevista liitosolioista ja sisältää painikkeet, joita tarvitaan liitosolioiden muokkaamiseen. Kuvassa 6 näkyy, miten DAO:n alla on kuvattuna tietokannan rakenne ja sen eri taulut ja näiden sisällä olevat metodit ja operaatiot.



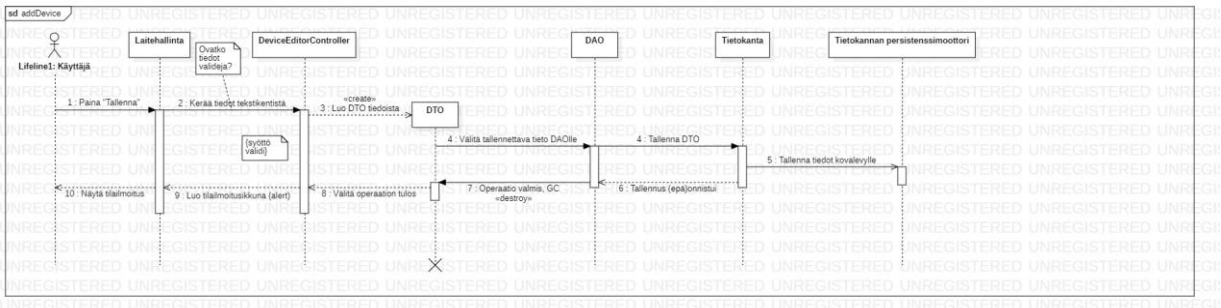

Kuva 6. Projektin DAO-kokonaisuutta kuvaava luokkakaavio.

## 4.2 Muita projektin kuvaamiseen käytettyjä kaavioita

Projektista luotiin myös sekvenssikaavio, jossa kuvataan prosessia, kun käyttäjä uuden laitteen tiedot syötettyään klikkaa laitteen lisäysnäkymässä nappulaa ”Tallenna”. Tämän seurauksena tiedot kerätään syöttökentistä, ja niiden kelvollisuus tarkistetaan vielä kerran, vaikka ne on tarkistettu jo syöttövaiheessa tekstikenttien validoinnin yhteydessä siltä osin, kun se on järkevää rajoittamatta liiaksi sitä, mitä käyttäjä voi kenttiin syöttää. Käyttöliittymä ei päästä käyttäjää tallentamaan tietoja, jos syöttö ei ole kelvollinen ja prosessi katkeaa ja näyttää käyttäjälle virheilmoituksen.

Jos syöttö on kelvollinen, kontrolleriluokka luo DTO:n keräämistään tiedoista, joka välitetään DAO:lle tallennusta varten. DAO parsii DTO:n ja antaa tietojen tallentamiseen tarvittavat SQL-käskyt tietokannalle. Tietokanta lisää tietojen muutokset muutosjonoon, ja ne tallennetaan kovalevylle asynkronisesti.

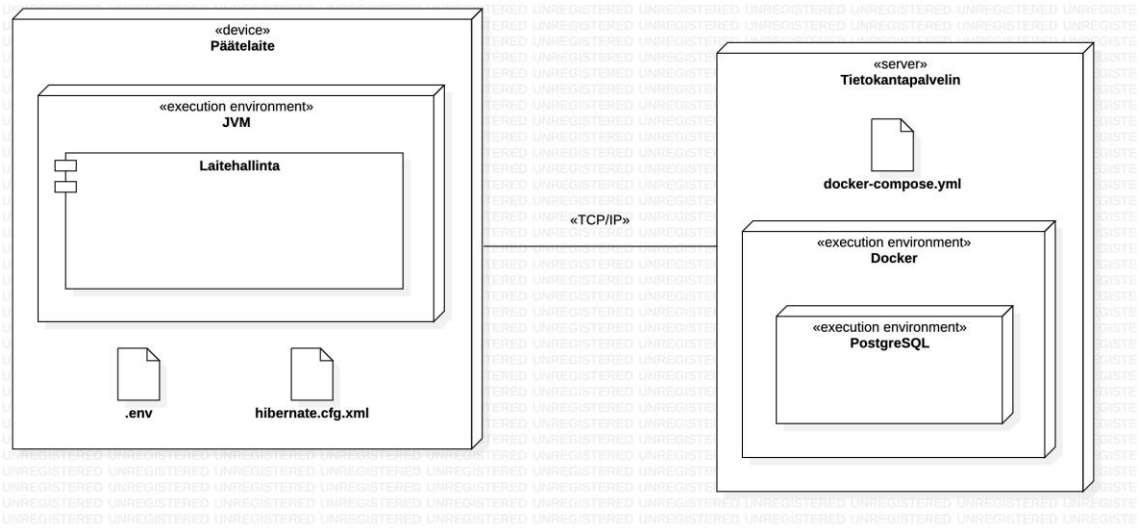
Kun tietokanta ilmoittaa, että joko kaikki operaatiot onnistuivat, tai että jokin niistä epäonnistui, lähtee asiasta paluuviesti kontrollerille metodien paluuarvojen kautta. DTO päätyy samassa prosessissa roskankeräykseen sen luovan funktion instanssin olemassaolon päättyessä. Kontrolleri ilmoittaa lopuksi tallennusoperaation onnistumisen tilan käyttäjälle. Edellä kuvailtu prosessi näkyy kuvan 7 sekvenssikaaviosta.




Kuva 7. Ohjelman tietojen tallentamista kuvaava sekvenssikaavio.

Sijoittelukaavio kuvassa 8 kuvastaa ohjelmiston eri osien fyysisen sijainnin ja rakenteen valmiissa ohjelmistotuotteessa. Ohjelmistotuote pyörii käyttäjän päätelaitteella, jolla sijaitsevat myös ohjelmiston tarvitsemat konfigurointitiedostot. Suoritusympäristönä on JVM (Java virtual machine), jossa ohjelmistotuote sijaitsee. Lisäksi ohjelmisto tarvitsee

pääsyn tietokantapalvelimelle, jossa pyörii PostgreSQL tietokanta Docker-kontin sisällä. Kehitystyön päätteeksi tietokanta jäi paikalliseksi, sillä demon ja tuotteen toiminnan kannalta tämä on riittävä ratkaisu. Jos tuote kehitettäisiin todellisen asiakkaan käyttöön, keskitetty tietokanta olisi tarpeen.




Kuva 8. Projektin sijoittelukaavio.

## 5 Lokalisointi ja käyttäjäkeskeisyys

Yksi ohjelmistotuotantoprojekti 2-kurssin keskeisistä näkökulmista oli ajatella ja harjoitella tuotteen lokalisaatiota. Tarkoitus oli suunnitella ja toteuttaa tuki käyttää ohjelmistoa eri kielillä ja lisäksi ottaa huomioon erilaiset mahdolliset kulttuurisidonnaiset näkökulmat, kuten esimerkiksi erilaiset yksiköt, desimaalierottimet, vakiintuneet ikonit ja muut seikat, jotka liittyvät ohjelmiston käyttöön ympäristön vaikutuksesta katsoen.

### 5.1 Ohjelmointiprojektin lokalisointi

Alun perin teimme kehitystyötä englannin kielellä ja työn edetessä lokalisointivaiheeseen teimme käyttöliittymästä myös suomen ja ruotsinkieliset versiot. Kehitystyön tapahtuessa jo valmiiksi englanniksi oli opintojakson englanninkieliseen loppupresentaatioon helpompi lähteä, kun koodissa käytettävä termistö oli jo valmiiksi oikealla kielellä. Lokalisointi tapahtui tekemällä properties-tiedostot jokaiselle kielelle, joissa on jokaiselle

aiemmin kovakoodatulle käyttöliittymässä näkyvillä olevalle tekstille vastine kullakin kielellä. Tämä mahdollistaisi jatkossa myös uusien kielien lisäämisen suhteellisen pienellä vaivalla.

Sovelluksemme luonteen ja valittujen kielivaihtoehtojen alueellisen samankaltaisuuden vuoksi kielio minaisuuksien lisäksi ei ollut tarpeen tehdä juuri muita muutoksia työn rakenteeseen. Länsimaalaiset kulttuurit ovat monella tapaa toistensa kaltaisia, tekstiä luetaan samalla tavalla ja monet yleisessä käytössä olevat ikonit käyvät ristiin eri kulttuurien kesken. Myös värejä käytetään kuvaamaan samankaltaisia asioita.

Tietokoneiden tekniset tiedot ja tapa ilmoittaa ne ovat suhteellisen universaaleja ja alalla tyypillisesti käytetään muutenkin monissa asioissa englannin kieltä fyysisestä sijainnista huolimatta. Myös yksiköt, joita käytetään kuvaamaan laitteiden ominaisuuksia, ovat pitkälti samat eri kulttuureissa. Esimerkiksi kannettavan tietokoneen näytön koko ilmaistaan meillä Suomessakin tuumina, eikä senttimetreinä, vaikka muuten käytämmekin yleensä metristä järjestelmää melkein kaiken mittaamiseen.

Ohjelmistomme ei myöskään sisällä ainakaan toistaiseksi sellaista numeerista dataa, jossa desimaalierottimia ja muita vastaavia seikkoja tarvitsisi ottaa huomioon. Opintojakson hengen mukaisesti nämä asiat kuitenkin käytiin läpi ajatuksella ja ohjelmistoa mietittiin myös näistä näkökulmista.

## 5.2 Ohjelmiston käyttäjäkeskeisyys

Vaikka käyttäjäkeskeisen suunnittelun kurssi ei varsinaisesti ollutkaan mukana ohjelmistotuotantoprojekti 2:ssa, käytettiin kurssilla monessa tehtävässä esimerkkinä omia projektituotoksia. Yksinkertaisuus käyttöliittymässä oli alusta saakka yksi suurimmista pyrkimyksistämme ja tämä ohjasi kaikkea tekemistämme, sekä suunnittelu- ja kehitystyötä.

Käyttäjäkeskeisen suunnittelun kurssilla opimme paljon monista tiedostamattomistakin asioista, jotka tuntuvat reaalimaailmassa itsestään selviltä. Ohjelmistoomme ei tullut suuria muutoksia kurssilla opitun perusteella, vaikka pieniä parannuksia tehtiinkin kurssilla saadun palautteen perusteella. Monissa kurssin tehtävissä päädyimme tekemään tehtäviä muista aiheista, koska käyttäjäkeskeisyys ja kohderyhmä oli otettu jo tuotteen kehitystyössä niin voimakkaasti huomioon, että henkilökohtaisen kehityksen kurssin asioiden kannalta oli parempi harjoitella ongelmien löytämistä muista systeemeistä.

Ohjelmiston käyttöliittymän yksinkertaisuus ja erilaisten toimintojen rajallinen määrä ja samankaltaisuus esimerkiksi tietojen lisäämisen ja muokkaamisen välillä teki käyttäjän pitämisestä kehitystyön keskiössä helppoa.

# 6 Yhteenveto

Projektin edetessä vastaan on tullut monenlaisia haasteita ja toisinaan aikatauluttaminen on tuottanut ryhmälle päänvaivaa. Visiossa ja suunnitelmissa pysyttiin hyvin ja ohjelmisto, käyttöliittymä tai projektin rakenne eivät juurikaan muuttuneet matkan varrella. Yhteistyö arvalla valitun ryhmän kanssa voi toisinaan luoda työskentelyyn omia haasteitaan itse valittuun ryhmään nähden. Haasteiden yli kuitenkin on päästy, joten kuten pitkin projektityöskentelyä, vaikka erimielisyyksiltä ja kommunikaatiohaasteilta ei olekaan täysin vältytty.

Varsinaisen sovelluksen kehitystyön lisäksi oli opintojakson tarkoituksena opetella projektinhallintaa scrum-menetelmällä ja uusien työkalujen hyödyntämistä. Nektion projektinhallintatyökalun kankea käyttöliittymä ja sen päivittämiseen kulutettu aika aiheutti ryhmälle toisinaan harmaita hiuksia, samoin se, että kaikki sprintin yksityiskohdat eivät ole tähän mennessä hankitulla osaamisella ja kokemuksella aina mitenkään ennakoitavissa ja on todella vaikea arvioida tiettyihin tehtäviin käytettävää aikaa.

Mielekästä oli uuden oppiminen uusien teknologioiden myötä, vaikka niistä toisinaan aiheutui myös ylimääräistä päänvaivaa. Projektista puhuminen englannin kielellä ja lokalisaatioasiat olivat hyvä uusi oppi kehittäjän työkalupakkiin ja varmasti myös tärkeitä asioita tulevaisuuden ja työelämän kannalta.

Dokumentaatio ja sovellus saatiin hyvään vaiheeseen opintojaksojen puitteissa, viimeiseen demoon mennessä saatiin kasalle toimiva sovellus ja myös muilla projektin ohessa käydyillä kursseilla tehdyt ryhmätyöt saatiin lopulta kunnialla tehtyä ja mallintamista, lokalisaatiota ja monia muita oppeja päästiin harjoittelemaan myös osana projektityöskentelyä. Jatkokehitysideoiksi tulevaisuuteen jäävät ainakin erilaiset tarkemmat haku- ja suodatustoiminnot, sekä tuki mobiililaitteille.

# 7 Device Patrol -ohjelmiston käyttöliittymän käyttöohje

Tässä luvussa perehdymme tarkemmin ohjelmistotuotteemme Device Patrolin käyttöohjeeseen. Käyttöliittymästä löytyvät eri näkymät ja painikkeet toimintoineen on kuvattu seuraavissa alaluvuissa, jotta tuotteen käyttöönotto ja itse tuotteen käyttäminen olisi helppoa ja miellyttävää.

## 7.1 Device Patrolin käyttöönotto

Device Patrol ei tällä hetkellä toimi itsenäisenä sovelluksena vaan tarvitsee toistaiseksi toimiakseen ohjelmointiympäristön ja paikallisen tietokannan. Tuotteen toimivuus on testattu ainoastaan IntelliJ IDEA -ohjelmointiympäristöllä. Sovellusta voi käyttää jar-tiedostolla it-asset-management-app-1.0-SNAPSHOT-shaded. Paikallista tietokantaa ajetaan Dockerilla, tätä varten repositorion juuressa on Docker-compose.yml -tiedosto. Aja komento `docker compose up -d`.

## 7.2 Päänäkymän käyttäminen

Kun käyttäjä avaa sovelluksen, aukeaa näkyviin suoraan listanäkymä, joka näyttää kaikki tietokannassa olemassa olevat laitteet tietoineen. Laitteita voi tässä päänäkymässä järjestää uudelleen jokaisen tietoja sisältävän sarakkeen perusteella klikkaamalla kunkin sarakkeen otsakepalkkia.

Tila-sarakkeen tietoa pystyy muuttamaan vetovalikosta suoraan listanäkymässä. Kullakin laitteen tietoja sisältävällä rivillä on toimintonappinappi, joka sisältää vaihtoehdot laitteen muokkaamiselle ja valikko-osion, joka sisältää vaihtoehdot Näytä ja Poista.

Näytä -painiketta klikattaessa valitun laitteen yksityiskohtaisemmat tiedot aukeavat näytön vasempaan laitaan erilliseen näkymään, josta tiedot ovat helpommin tarkasteltavissa. Näkymästä pääsee pois klikkaamalla palkin oikeasta ylälaidasta ruksipainiketta.

Poista-painiketta klikattaessa poistuu valitun yksittäisen laitteen kaikki tiedot tietokannasta. Tiedot käyttäjästä, käyttöjärjestelmästä, laitekokoonpanosta ja sijainnista jäävät tietokantaan käytettäväksi uudelleen.

Muokkaa-painiketta painaessa käyttäjä voi muokata valittua laitetta, jolloin aukeaa ponnahdusikkuna, joka sisältää valmiiksi muokattavaksi valitun laitteen tiedoilla täytetyt kentät laitteen tallennetuille tiedoille. Peruuta-painiketta painaessa muokatut tiedot katoavat, eivätkä tallennu tietokantaan. Muokkausikkuna sulkeutuu samalla. Painamalla Tallenna, tiedot tallentuvat tietokantaan. Mikäli pakollisia kenttiä on tyhjänä tai niihin syötetty tieto on muodoltaan vääränlaista, ilmoittaa ohjelma siitä varoituksella, eivätkä tiedot tallennu, ennen kuin ne on korjattu ja painettu tallenna-painiketta uudelleen.

### 7.3 Ylävalikko ja loput muokkausnäkymät

Ohjelman yläpalkissa on valikkoelementtejä. Tiedosto-painikkeesta aukeavat vaihtoehdot Uusi ja Ylläpidä. Uusi-valikolla on alaelementteinä Laite, Kokoonpano, Käyttöjärjestelmä, Käyttäjä ja Sijainti. Klikkaamalla Laite, avautuu samanlainen muokkausikkuna, joka aukeaa päänäkymän muokkauspainikkeessa, mutta uutta laitetta lisättäessä lomake on aluksi tyhjä ja tiedot täytetään itse.

Vapaiden tekstikenttien lisäksi näkymä sisältää vetovalikoita, joista voi valita jo tietokannassa olemassa olevia ominaisuuksia laitteelle. Mikäli valmiit vaihtoehdot eivät vastaa uuden laitteen tietoja, on vetovalikoiden vieressä vasemmalla puolella pieni painike, josta voi lisätä uusia laitekokoonpanoja, käyttöjärjestelmiä, käyttäjiä ja sijainteja. Näistä lisäyspainikkeista aukeavat lisäysnäkymät ovat täysin samanlaisia kuin Tiedosto → Uusi valintojen kautta aukeavat lisäysnäkymät laitekokoonpanoille, käyttöjärjestelmille, käyttäjille ja sijainneille. Peruuta-painikkeita painaessa täytetyt tiedot nollaantuvat ja ikkuna sulkeutuu. Tallenna-painiketta painaessa tiedot tallentuvat tietokantaan, jos ne ovat kelvollisia. Muussa tapauksessa ohjelma antaa virheilmoituksen.

Ylävalikon vaihtoehdosta Ylläpidä avautuu taulukkonäkymä, jossa on listattuna kaikki tietokantaan tallennetut laitekokoonpanot, käyttöjärjestelmät, käyttäjät ja sijainnit. Jokaisen sarakkeen perässä on toimintonäppäin, jota klikkaamalla valitun solun sisältöä voi muokata. Näistä painikkeista aukeavat näkymät ovat samanlaisia, kuin uusia lisätessä, mutta muokkausnäkymässä kentät on esitäytetty valitun laitteiden senhetkisillä tiedoilla.

Ohjelmiston yläpalkissa on myös valikko Kieli, josta käyttäjä voi valita, haluaako käyttää ohjelmaa suomeksi, ruotsiksi vai englanniksi. Klikkaamalla haluttua kieltä kaikki sovelluksen valikot, painikkeet ja tekstikenttien selitteet vaihtuvat valitun kielen mukaiseksi.

Halutessaan poistua jostakin näkymästä tai sovelluksesta kokonaan, voi käyttäjä klikata aina kulloisenkin ikkunan oikeasta ylälaidasta ruksipainiketta. Mikäli kyseessä on tietojen lisääminen tai muokkaaminen, peruuttaa ruksin painaminen käynnissä olevat muutokset. Jos käyttäjä haluaa poistua sovelluksesta ja klikkaa ruksia, sovellus sammuu. Tietokantaan tallennetut tiedot jäävät voimaan ja seuraavalla käyttökerralla ne näkyvät sovelluksen näkymissä.