OULUN AMMATTIKORKEAKOULU

Juho Kettunen

YLLÄPITOSOVELLUS PUUAUTOMAATTIJÄRJESTELMÄÄN

YLLÄPITOSOVELLUS PUUAUTOMAATTIJÄRJESTELMÄÄN

Juho Kettunen
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Juho Kettunen
Opinnäytetyön nimi suomeksi: Ylläpitosovellus puuautomaattijärjestelmään
Työn ohjaaja: Teemu Korpela
Työn valmistumislukukausi ja -vuosi: Syksy 2021
Sivumäärä: 29

Opinnäytetyön tilasi Yritetään yhdessä ry:n Kestävän kehityksen keskus. Tavoitteena oli kehittää selaimella toimiva ylläpitosovellus Kestävän kehityksen keskuksen Halkonen-puuautomaattijärjestelmään. Automaatista nuotiopaikalla asioiva voi ostaa nuotioon tarvittavat polttopuut. Ylläpitosovelluksella nuotiopaikan ylläpitäjä voi luoda uusia automaatteja, seurata niiden tilaa sekä merkitä automaatin täytetyksi.

Sovellus rakennettiin käyttäen Googlen kehittämää alustariippumatonta Flutter-kehitysalustaa. Tietokantaan ja käyttäjien todennukseen käytettiin Firebase-pilvialustan Firestore- ja Authentication-tuotteita. Toteutus aloitettiin suunnittelemalla sovelluksen vaatimuksia, käyttöliittymää, toimintoja ja tietokantaa.

Tuloksena valmistui websovellus, jossa on kaikki suunnitellut ominaisuudet. Sovelluksella ylläpitäjät voivat luoda uusia ylläpitäjiä, joille voidaan antaa erilaisia oikeuksia. Ylläpitäjät voivat myös luoda ja muokata automaatteja järjestelmässä sekä merkitä automaatteja täytetyksi.

Asiasanat: Flutter, mobiilisovellukset, Firebase, ohjelmointi, web-sovellukset

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

Author: Juho Kettunen
Title of thesis: Administration application for a firewood vending machine
Supervisor: Teemu Korpela
Term and year when the thesis was submitted: Autumn 2021
Pages: 29

The goal of this thesis was to create a web application to manage a firewood vending machine system. Campers and hikers can buy firewood from these vending machines. With the admin app, the managers of the vending machines can mark them as filled, create or edit new vending machines and monitor their status.

The app was build using the Flutter software development kit by Google. The backend of the application was build using Firebase cloud platform products. With Firebase Firestore as the database and Firebase Authentication for user authentication.

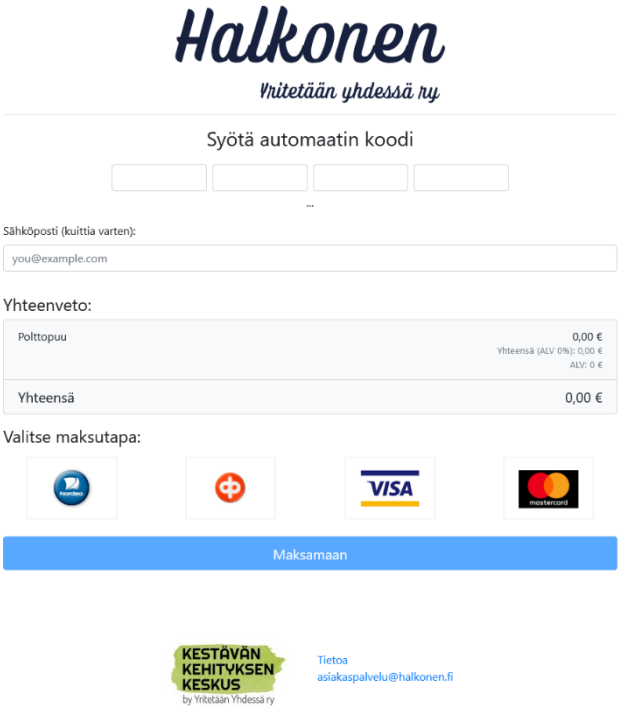
The result was a web application with all the planned main components. The app allows managers of the vending machines to create, edit and mark them filled. The managers can have different privileges such as filler, admin or owner depending on their role in their organization.

Keywords: Flutter, web applications, Firebase, programming, mobile applications

SISÄLLYS

1 JOHDANTO

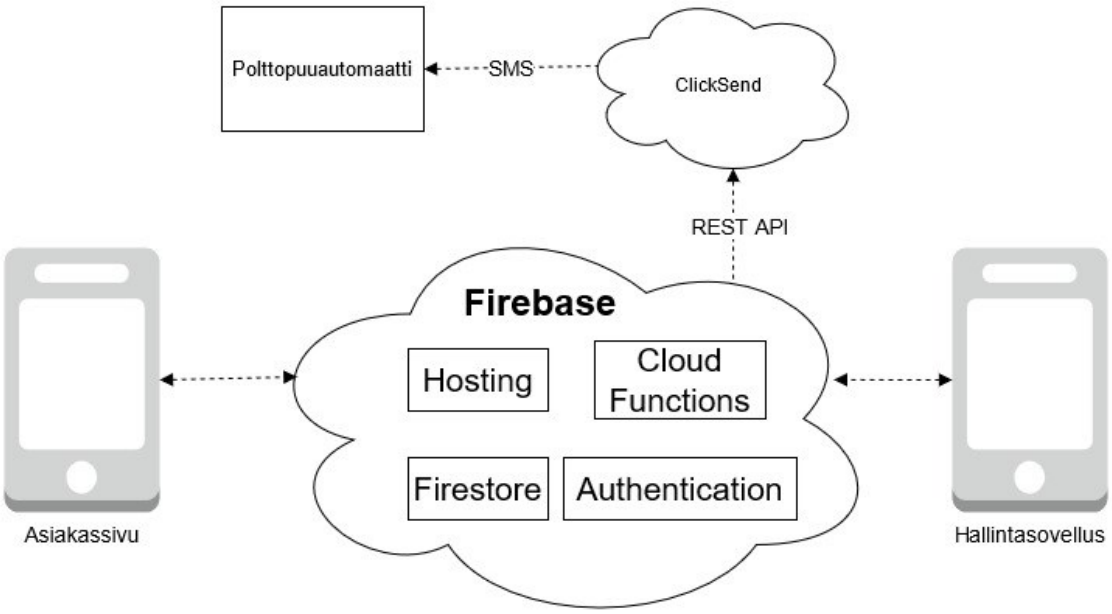
Opinnäytetyö tehtiin keväällä 2021 Yritetään yhdessä ry:n Kestävän kehityksen keskukselle. Työ on osa Halkonen-polttopuuautomaattihanketta. Ennen opinnäytetyön aloittamista tein harjoittelun merkeissä hankkeeseen tilaussivun (kuva 1), josta polttopuita voidaan tilata, sekä tähän liittyvät Firebase taustajärjestelmät lokeron avaamiseen ja maksujen käsittelyyn.



KUVA 1. Harjoittelutyönä tehty tilaussivu

Kestävän kehityksen keskus ylläpitää osaa Oulun alueen nuotiopaikoista. Viime vuosina nuotiopaikoilta on useamman kerran varastettu polttopuita, joskus isojakin määriä (1). Lisäksi retkeilijät ovat saattaneet käyttää puita tarvettaan enemmän. Tästä on koitunut lisäkustannuksia nuotiopaikkojen ylläpitoon ja harmia retkeilijöille puiden loppuessa. Halkonen on polttopuuautomaattijärjestelmä, joka tarjoaa retkeilijöille mahdollisuuden ostaa polttopuita suoraan nuotiopaikan automaatista. Järjestelmän toivotaan estävän puiden varastamista ja liikakäyttöä. Se mahdollistaisi myös polttopuiden käytön seuraamista, jolloin polttopuita voitaisiin tuoda nuotiopaikoille tarpeen vaatiessa.

Järjestelmä koostuu automaatista sekä maksu- ja hallintasivuista (kuva 2). Sivut kommunikoivat automaatin kanssa SMS-viestejä käyttäen. Viestien välittämiseen käytetään ClickSend-viestimispalvelua, jonka REST-rajapintaa käytetään viestien lähettämiseen.



KUVA 2. Järjestelmäkuvaus

Tässä opinnäytetyössä tehtiin järjestelmälle ylläpitosovellus, jonka avulla automaatin hoitaja voi hallita automaatteja. Ylläpitäjät voivat lisätä sovelluksella järjestelmään uusia käyttäjiä ja antaa heille erilaisia oikeuksia sovellukseen. Käyttäjä voi roolinsa oikeuksien perusteella merkata automaatin täytetyksi, tarkistaa automaatin täyttötilanteen, lisätä uuden automaatin tai muokata automaattia. Tarkoituksena oli saada aikaan sovellus, jolla pystytään testaamaan ja kehittämään järjestelmän toimintaa. Jos järjestelmä päätetään siirtää tuotantovaiheeseen, voidaan sovellusta jatkokehittää monipuolisemmaksi.

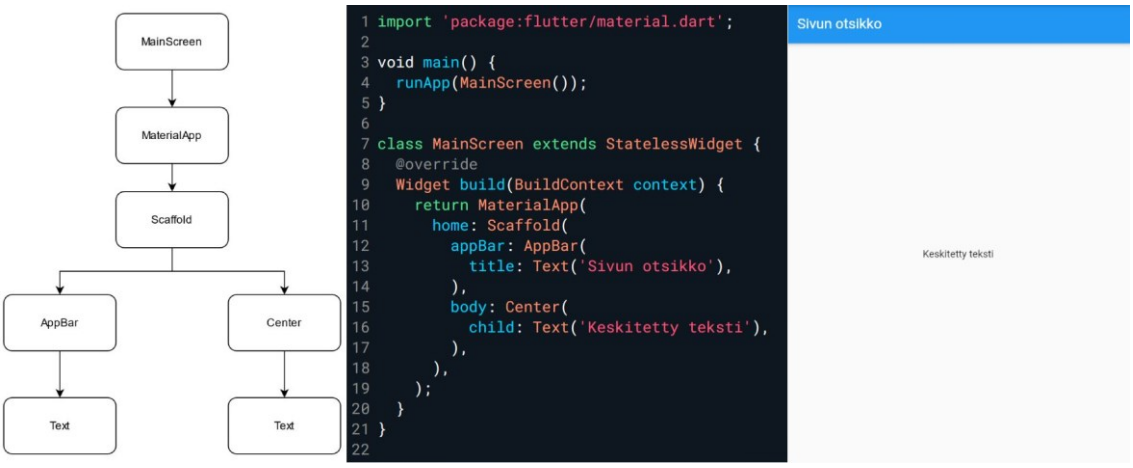
2 KÄYTETYT TEKNOLOGIAT JA TYÖKALUT

Ohjelmoinnissa käytettiin Android Studio- ja Visual Code-ohjelmointiympäristöjä, joihin saa virallisten lisäosien avulla tuen Flutterille ja Firebaselle. Käyttöliittymän suunnitteluun käytettiin Pencil- käyttöliittymän suunnittelutyökalua ja tietokannan suunnittelussa käytettiin apuna draw.io-kaavionpiirto-ohjelmaa. Projektinhallinnassa sovellettiin ketterien menetelmien periaatteita ja käytettiin Trello-projektinhallintatyökalua. Versiohallintaan käytettiin Git-versionhallintaohjelmaa, jolla lähdekoodi tallennettiin aina muutosten jälkeen GitHub-palveluun.

2.1 Flutter

Flutter on Googlen kehittämä ja ylläpitämä avoimen lähdekoodin ohjelmointirajapinta (SDK), jolla voidaan tehdä alustariippumattomia sovelluksia Android-, iOS-, Windows-, macOS- ja Linux-käyttöjärjestelmille sekä web-sovelluksia verkkoselaimille. Flutterin on tarkoitus nopeuttaa ja helpottaa sovelluskehittämistä usealle alustalle ilman merkittävää suorituskyvyn heikentymistä verrattuna natiiveihin ohjelmointialustoihin. Ohjelmointikielenä se käyttää Googlen kehittämää Dart-kieltä. (2.)

Widgetit ovat Flutterissa käyttöliittymän rakennukseen käytettäviä luokkia. Käyttöliittymän ulkonäkö rakennetaan sisällyttämällä widgettejä toisiinsa, jolloin ne muodostavat hierarkkisen rakennelman. (Kuva 3.) Usein widgetit rakentuvat useasta pienemmästä widgetistä, joita pystytään muokkaamaan sovelluksen tarpeisiin ja ulkonäköön sopiviksi. Ne on tehty tiettyyn tarkoitukseen ja niiden funktio on tehdä vain annettu tehtävänsä mahdollisimman hyvin. Esimerkiksi sen sijaan, että kaikilla käyttöliittymä widgeteillä olisi omat kohdistamis- ja pehmustusominaisuudet, on Flutterissa omat Padding- ja Center-widgetit, joilla voidaan ympäröidä widget, joka halutaan ympäröidä pehmusteella tai keskittää. (2.)



KUVA 3. Vasemmalla widget-hierarkia, keskellä koodi, oikealla sovellus

Yksi Flutterin ja Dart-kielen parhaita ominaisuuksia on kehittäjien käytettävissä oleva laaja valikoima erilaisia kirjastoja. Osa näistä on Googlen itsensä kirjoittamia, mutta suurin osa on muiden Flutter-kehittäjien tekemiä. Kirjastoja voi selata selaimella osoitteessa pub.dev. Asentamiseen riittää kirjaston lisääminen projektin riippuvuuksiin ja ”pub get” -komennon ajaminen. Sovelluskehittäjät voivat luottaa käytettyjen kirjastojen olevan aina saatavilla, sillä kirjastojen kehittäjät eivät voi ilman erittäin painavaa syytä poistaa kirjastojaan. (3.)

Flutterin vahvimpina vaihtoehtoina olisivat olleet Facebookin kehittämä React Native tai Googlen ylläpitämä Angular. Alustoilla on suorituskyvyn ja dokumentoinnin suhteen vain pieniä eroja. Angular ja React ovat Flutteria suositumpia, joten ohjeita ja apua löytyy niitä käytettäessä internetistä hieman enemmän. (4.) Flutter on puolestaan vähän suorituskykyisempi (5). Kaikki olisivat olleet sovelluksen tarkoitukseen hyviä valintoja. Flutter valittiin, koska siitä löytyi jo aikaisempaa kokemusta, joten itse työhön jäi enemmän aikaa kokonaan uuden alustan opiskelemisen sijaan.

2.2 Firebase

Firebase on Googlen mobiili- ja webkehittäjille suunnattu alusta, joka tarjoaa sovelluskehityksessä usein tarvittavia palvelinratkaisuja helposti käytettävässä muodossa. Alustan tuotteisiin kuuluvat mm. Cloud Firestore- ja Realtime Database -tietokannat, käyttäjän todennuspalvelu Firebase Authentication, hostingpalvelu Firebase Hosting, useita analytiikka- ja testauspalveluita sekä Cloud

Functions -pilvipalvelu, jossa voidaan ajaa palvelinpuolen koodia. Halkonen-järjestelmässä käytetään Firestore-, Authentication-, Hosting- ja Cloud Functions -tuotteita. Kehitysvaiheessa sovelluksen testaukseen käytettiin alustaan kuuluvia emulaattoreita. (6.)

Firebase valittiin, koska se tarjoaa tarvittavat taustajärjestelmät, jotka on helppo yhdistää Flutter-sovellukseen. Lisäksi siitä löytyi jo aiempaa kokemusta. Vaihtoehto Firebaselle olisi Amazonin AWS Amplify, joka on Firebasen tyylinen sovelluskehittäjille suunnattu alusta. Amplifyn Dart-kirjaston vakaa versio julkaistiin alkuvuodesta 2021. Firebase-kirjasto puolestaan on ollut olemassa käytännössä Flutterin ensimmäisestä julkaisusta asti, joten sen käytöstä löytyy huomattavasti enemmän ohjeita ja esimerkkejä. (7.) Firebasen eri tuotteita voidaan myös käyttää yhdessä muiden ratkaisujen kanssa. Työtä suunnitellessa kuitenkin todettiin, että sen toteutus halutaan pitää mahdollisimman yksinkertaisena, joten Firebasen kokonaisuus osoittautui hyväksi ratkaisuksi.

2.2.1 Firebase Firestore

Firestore on pilvipohjainen NoSQL-tietokanta, jonka käyttöön löytyy kirjastot muun muassa natiivi Android- ja iOS-järjestelmille sekä Node.js-, Python-, C++- ja Go-ohjelmointikielille. Data tallennetaan dokumentteihin avain-arvopareihin ja dokumentit tallennetaan kokoelmiin. Dokumentit voivat myös sisältää omia kokoelmiaan, jolloin voidaan rakentaa hierarkkisia datarakennelmia. (8.)

2.2.2 Firebase Authentication

Firebase Authentication mahdollistaa helposti käyttäjien turvallisen rekisteröitymisen, kirjautumisen ja tietojen tallennuksen. Se tukee sähköpostilla ja salasanalla kirjautumisen lisäksi kirjautumista kolmansien osapuolien tunnuksilla, kuten Facebook, Twitter tai Github. (9.) Palvelu toimii yhdessä muiden Firebase-palveluiden kanssa, mikä mahdollistaa esimerkiksi käyttäjien oikeuksien rajoittamisen tietokantaan. Authentication tukee maksullisena palveluna myös monivaiheista tunnistautumista SMS-viesteillä. (10.)

2.2.3 Cloud Functions

Pilvifunktioilla voidaan ajaa serveripuolen koodia pilvessä ilman tarvetta varsinaisille palvelimille. Javascript- tai Typescript -kielillä kirjoitetut funktiot voivat vastata HTTP-kutsuihin tai ne voivat reagoida erilaisiin tapahtumiin Firebasessa, esimerkiksi tietokantaan kirjoittaessa tai uuden käyttäjän rekisteröityessä. Funktiot skaalautuvat automaattisesti käyttäjämäärän mukaan. (11.)

2.2.4 Firebase Hosting

Firebase Hosting on pilvipohjainen hostingpalvelu, joka on kehitetty erityisesti web-sovellusten ja mikropalveluiden tarpeisiin. Palvelussa voidaan staattisten www-sivujen tai web-sovellusten lisäksi isännöidä myös dynaamista sisältöä yhdistämällä se pilvifunktioiden kanssa. Tällöin sitä voidaan käyttää myös esimerkiksi Express.js:llä rakennettuihin sovelluksiin ja palveluihin. (12.)

3 SUUNNITTELU

Vaatimuksia ylläpitosovellukselle oli mietitty jo ennen opinnäytetyötä harjoittelujakson aikana. Tärkeimpiä ominaisuuksia sovellukselle ovat automaattien luominen ja muokkaus, automaatin täytetyksi merkitseminen sekä automaatin täyttöasteen seuraaminen. Lisäksi sovellukseen haluttiin käyttäjienhallinta, jossa voidaan asettaa erilaisille käyttäjille eri oikeuksia. Koska järjestelmää ollaan mahdollisesti myymässä muille nuotiopaikkojen ylläpitäjille, haluttiin sovellukseen tuki myös usealle eri organisaatiolle, joilla on omat automaattinsa ja käyttäjät.

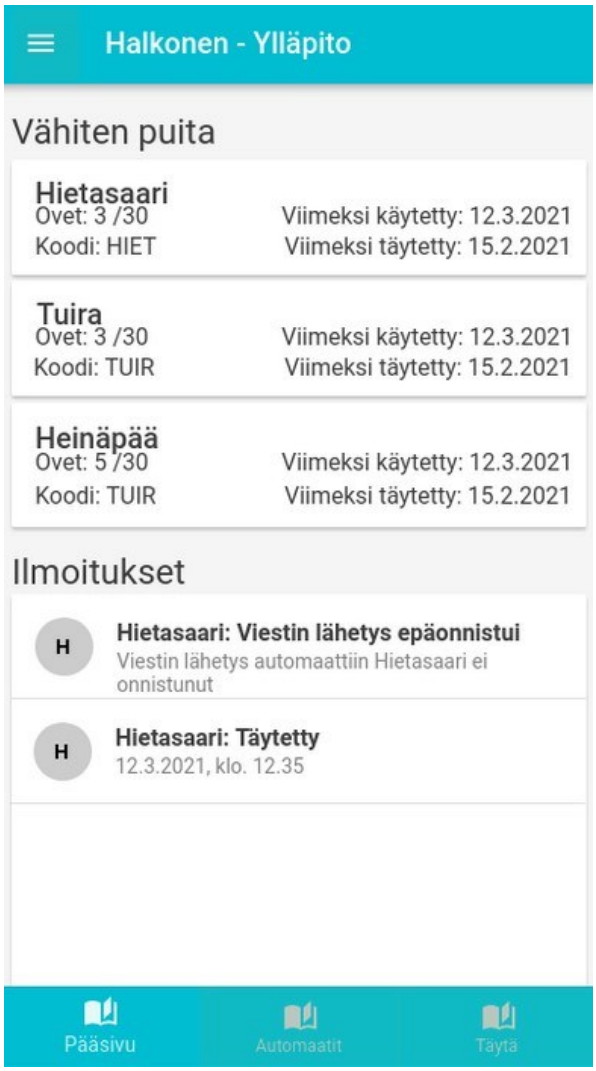
Sovelluksen taustajärjestelmä Firebaseen oli jo pitkälti valmis. Myös automaattien tietokanta oli suunniteltu harjoittelujakson aikana.

3.1 Tietokanta

Tietokannan suunnittelussa pääajatuksena oli, ettei tietokantakyselyitä tehtäisi turhaan, sillä Firestore laskuttaa asiakasta luettujen ja kirjoitettujen dokumenttien perusteella. Tästä syystä dataa toisinnetaan eri dokumenteissa pilvifunktioiden avulla. (13.)

3.2 Käyttöliittymä

Käyttöliittymän suunnittelussa käytettiin ilmaista Pencil-sovellusta. Tavoitteena oli tehdä käyttöliittymästä selkeä, looginen ja helppokäyttöinen. Esimerkiksi sovelluksessa käytetään alapalkkia navigointiin pääsivun, automaattisivun ja täyttösivun välillä, koska sovellusta käytetään pääsääntöisesti puhelimella. Näin käyttäjä pääsee helposti peukalolla käytetyimpiin ominaisuuksiin. (Kuva 4.)



KUVA 4. Pääsivun käyttöliittymäsuunnitelma

Koska käyttöliittymän uniikki ulkonäkö ei ole ylläpitosovelluksessa tärkeässä asemassa, päätettiin suunnittelussa käyttää valmiita Material Design -komponentteja. Lisäksi Material Design on käyttäjille tuttu muista sovelluksista. Se on myös Flutterissa oletuksena käytössä käyttöliittymäkomponenteissa, joten käyttöliittymän visuaaliseen suunnitteluun ei tarvinnut käyttää juurikaan aikaa. Käyttöliittymän suunnitelmaa pystyttiin hyödyntämään myös tietokannan suunnittelussa, sillä Firestoressa hyvänä lähtökohtana on, että sovelluksen yhdellä sivulla käytetään korkeintaan yhden kokoelman tietoja. (13.)

3.3 Käyttäjien roolit

Sovelluksella on useita erilaisia käyttäjiä, joilla on erilaiset oikeudet järjestelmään. Oikeudet on jaettu käyttäjien roolien mukaan taulukon 1 mukaisesti.

TAULUKKO 1. Käyttäjien oikeudet on luokiteltu roolien mukaan

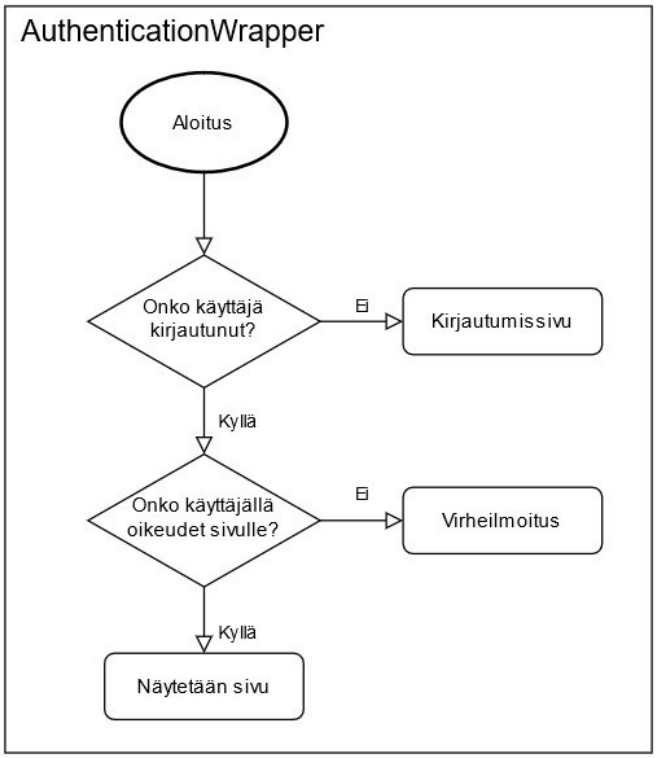
ROOLI	OIKEUDET
Master	• Voi luoda uuden organisaation
Owner	• Voi tarkastella maksutapahtumia • Voi antaa käyttäjille owner-oikeudet
Admin	• Voi luoda uusia käyttäjiä • Voi antaa käyttäjille admin- tai filler-oikeudet • Pystyy lisäämään ja muokkaamaan automaatteja
Filler	• Voi merkitä automaatti täytetyksi • Voi nähdä automaatin perustiedot

4 TOTEUTUS

4.1 Käyttäjien todennus

Käyttäjien todennukseen käytetään Firebase Authentication -palvelua. Palvelun käyttöönottoon Flutterissa riittää Firebase-kehittäjien firebase_core- ja firebase_auth-kirjastojen lisääminen projektiin. Palvelussa on mahdollisuus todentaa käyttäjä myös kolmansien osapuolien palvelujen kautta (mm. Facebook, Twitter ja GitHub), mutta sovelluksessa käytetään vain sähköpostilla ja salasanalla kirjautumista. Käyttäjän tiedot tallennetaan myös tietokantaan.

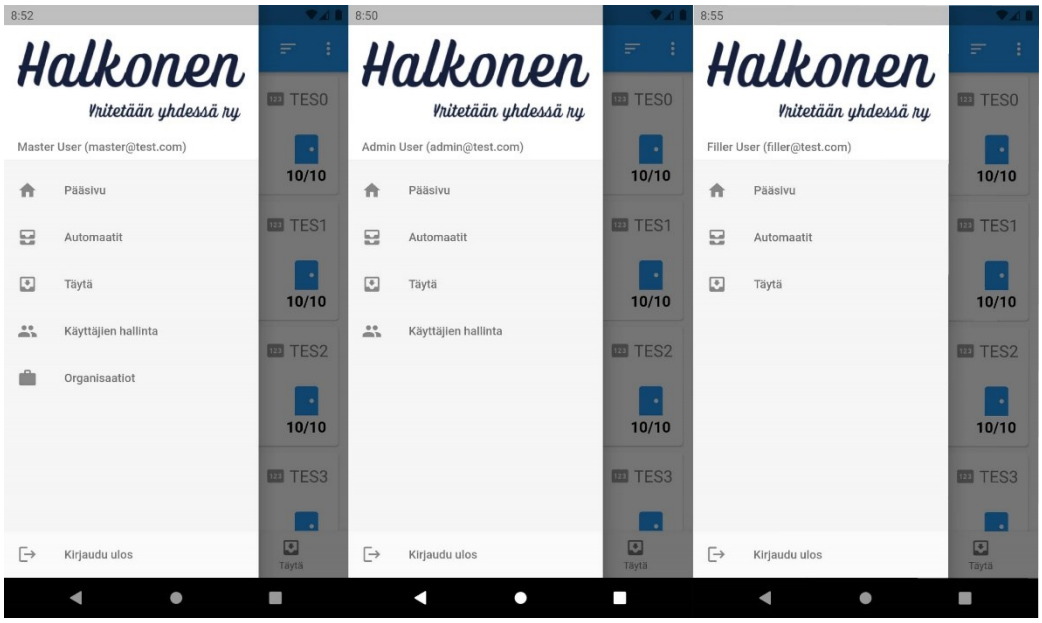
Käyttäjän kirjautumistilaa seurataan StreamProviderin avulla, joka ilmoittaa muutoksista käyttäjän kirjautumistietoihin. Jokainen sovelluksen sivu on ympäröity AuthenticationWrapper-luokalla. (Kuva 5.) Käyttäjän avatessa sovelluksen ensimmäisen kerran näytetään kirjautumissivu, koska StreamProviderilta saatu käyttäjä on tyhjä. Sisään kirjautuessa StreamProviderin tila muuttuu. Se ilmoittaa muutoksesta tilaajille ja rakentaa uudestaan widgetit, joissa se vaikuttaa. Koska käyttäjä on nyt olemassa, voidaan näyttää oikea sivu.



KUVA 5. AuthenticationWrapper-luokan toiminta

Käyttäjän tiedot ladataan kirjautumisen yhteydessä tietokannasta ja tallennetaan Flutter-tiimin kehittämän shared_preferences-kirjaston avulla laitteelle. Kirjastolla voidaan tallentaa yksinkertaisia data-arvoja tai esimerkiksi pieniä olioita JSON-merkkijonoina, kuten tässä tapauksessa käyttäjän tiedot tallennetaan.

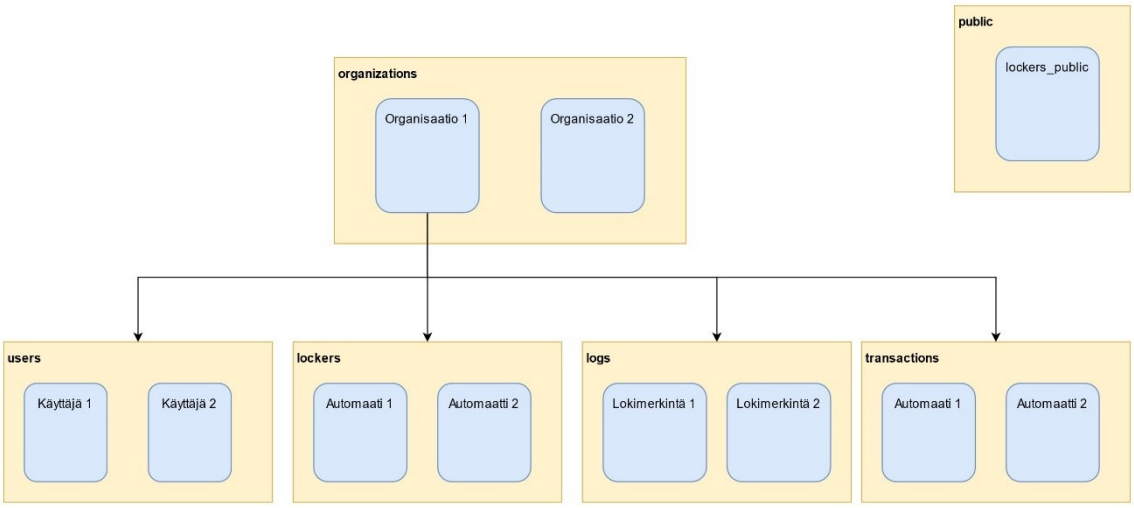
Sovelluksessa halutaan antaa erilaisia käyttöoikeuksia riippuen käyttäjän roolista ja työtehtävistä automaatin ylläpidossa. Roolien määrittelyyn käytetään Firebase Authenticationin Custom Claims -ominaisuutta, jolla voidaan määrittää käyttäjille erilaisia ominaisuuksia nimi-arvoparien avulla. Käyttäjien roolit määritellään sovelluksessa totuusarvomuuttujilla master, owner, admin ja filler. Roolien avulla määritetään esimerkiksi, mitä käyttöliittymän elementtejä käyttäjä näkee sovelluksessa (kuva 6) ja mihin tietokantadokumentteihin käyttäjällä on oikeudet.



KUVA 6. Eri käyttäjien näkymiä sivuvalikosta.

4.2 Tietokanta

Firestore on NoSQL-tietokanta, joten se ei seuraa ennalta määriteltyä taulukkoskeemaa. Tietokannassa on kokoelma, joka sisältää dokumentit eri organisaatioille. Nämä sisältävät organisaatioiden perustietojen lisäksi myös omat kokoelmansa käyttäjille, automaateille, lokimerkinnöille ja maksutapahtumille. Erilliseen julkiseen kokoelmaan on tallennettu koko järjestelmän kaikkien lokeroiden julkiset tiedot. (Kuva 7.) Dokumenttia käytetään tilaussivulla asiakkaan tilatessa puita automaatista sekä varmistamaan, ettei järjestelmään lisättävän automaatin koodia ole jo käytössä.



KUVA 7. Tietokanta koostuu kokoelmista (keltaiset laatikot) ja niiden sisältämistä dokumenteista (siniset laatikot).

Firestore-tietokantaan pääsyä voidaan rajoittaa asettamalla dokumenteille sääntöjä. Firebasen suosituksen mukaisesti kaikkiin dokumentteihin on oletuksena pääsy estetty. (Kuva 8.) Yllättävän yleinen virhe etenkin uusilla kehittäjillä on asettaa kehitystarkoituksessa vapaa pääsy tietokantaan. Tällöin kuka tahansa pystyy näkemään ja muokkaamaan tietokantaa. (14.)

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
```

KUVA 8. Pääsääntönä kaikkien dokumenttien luku- ja kirjoitusoikeudet on evätty.

Tarkempia sääntöjä voidaan määrittää dokumenttien polun mukaan. Esimerkiksi automaattien tietoja pääsee hakemaan vain varmennettu käyttäjä, jolla on ylläpitäjän oikeudet. Koska Firestore-tietokannan rakennetta ei määritellä etukäteen, on joissain tilanteissa mahdollista, että tietokantaan tallennetaan rakenteeltaan virheellisiä dokumentteja. Tietokannan säännöillä voidaan kuitenkin tarkistaa tallennettavan tiedon rakenne. Esimerkiksi järjestelmän maksupalveluna toimiva VismaPay vaatii maksettavan tuotteen tiedot tietyllä rakenteella. Tietokannan

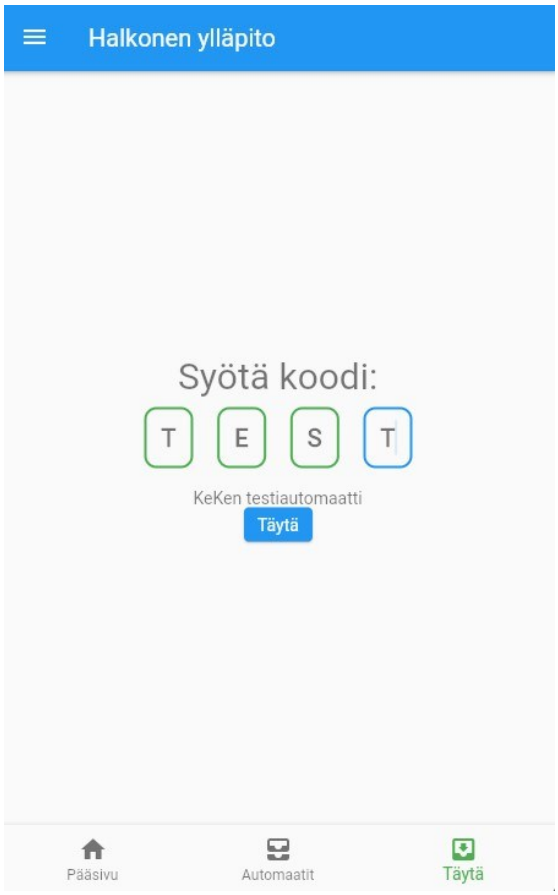
säännöissä on siksi määritetty, että automaattien tuotetiedot sisältävät aina ja vain ainoastaan tietyt kentät. (Kuva 9.)

```
match /organizations/{organization}/lockers/{lockerId} {
  allow read: if request.auth != null && request.auth.token.admin
  allow write: if request.auth != null && request.auth.token.admin && (request.resource.data.product.keys().hasOnly(
    ['id', 'title', 'count', 'tax', 'pretax_price', 'price', 'type']) &&
    request.resource.data.product.keys().hasAll(
    ['id', 'title', 'count', 'tax', 'pretax_price', 'price', 'type']) &&
    request.resource.data.keys().hasAll(
    ['code', 'name', 'sms_number']));
  allow delete: if request.auth != null && request.auth.token.admin
}
```

KUVA 9. Tietokannan säännöissä on määritelty, mitä kenttiä automaatin tuotetietojen tulee sisältää.

4.3 Automaatin täydeksi merkitseminen

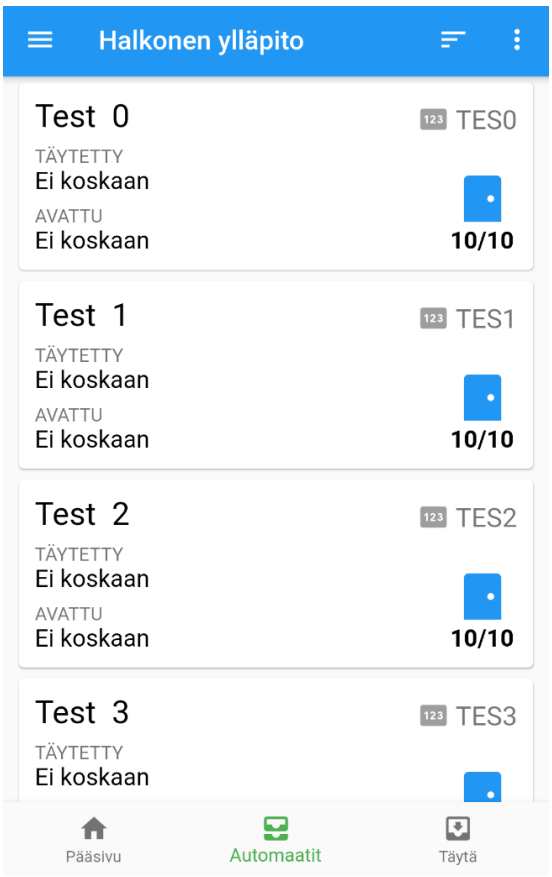
Täyttäjä merkitsee automaatin täytetyksi syöttämällä automaatin koodin kenttään ja painamalla Täytä-painiketta (kuva 10). Sovellus lataa jo sivulle siirtyessä listan kaikista automaateista ja vertailee käyttäjän kirjoittamaa koodia käyttäjän organisaation automaatteihin. Mikäli koodia vastaava automaatti löytyy, voi käyttäjä painaa Täytä-painiketta, jolloin automaatti merkitään tietokannassa täytetyksi. Lopuksi käyttäjä saa tiedon operaation onnistumisesta.



KUVA 10. Täyttö-sivu

4.4 Automaatit-sivu

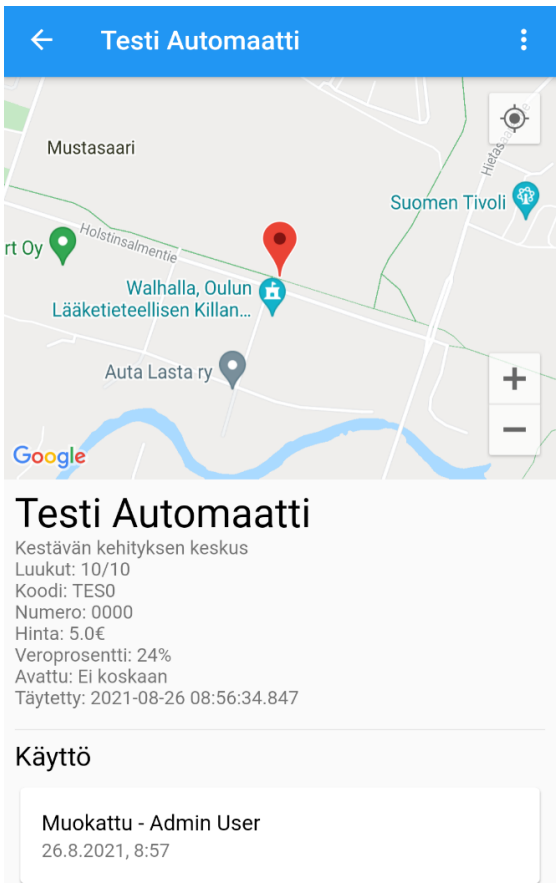
Automaatit-sivu listaa kaikki organisaation automaatit ja kertoo niiden perustiedot (kuva 11). Automaattien latauksessa käytetään StreamBuilder-widgettiä. Se kuuntelee tietokannan dokumentin tapahtumia, jossa automaatin tiedot ovat tallennettuina. Kun dokumentissa tapahtuu muutoksia, StreamBuilder rakentaa listauksen uudestaan muutetun dokumentin mukaisesti. Firestoren Dart-kirjasto myös tallentaa ladatun dokumentin välimuistiin. Täten dokumentti ladataan vain käyttäjän avatessa Automaatit-sivun ensimmäisen kerran. Tämän jälkeen tietokannasta haetaan uusi dokumentti vain, jos siihen on tullut muutos tai kun edellisestä hausta on kulunut haluttu aika, mikä on oletuksena 30 minuuttia.



KUVA 11. Automaatit-sivu

4.4.1 Automaatin infosivu

Automaatin infosivulta näkee automaatin ja siihen liittyvien tapahtumien tiedot (kuva 12). Perustietojen lisäksi sivulta näkee myös automaatin tapahtumat, kuten käyttökerrat ja täytöt. Tapahtumalokiin merkataan aika, käyttäjä ja tapahtuman tyyppi. Automaatin sijainnin näkee karttanäkymästä, joka on toteutettu Googlen kehittämällä google_maps_flutter-kirjastolla. Toinen suosittu karttakirjasto Fluttelle on Leaflet JavaScript-kirjastosta Dart-kielelle käännetty flutter_map-kirjasto, jossa voidaan karttapalveluna käyttää esimerkiksi OpenStreetMap- tai Azure-Map-aineistoa. Kirjasto ei ollut työtä tehdessä yhteensopiva sovelluksessa käytetyn Dart-version kanssa, jonka vuoksi sitä ei olisi voitu käyttää.

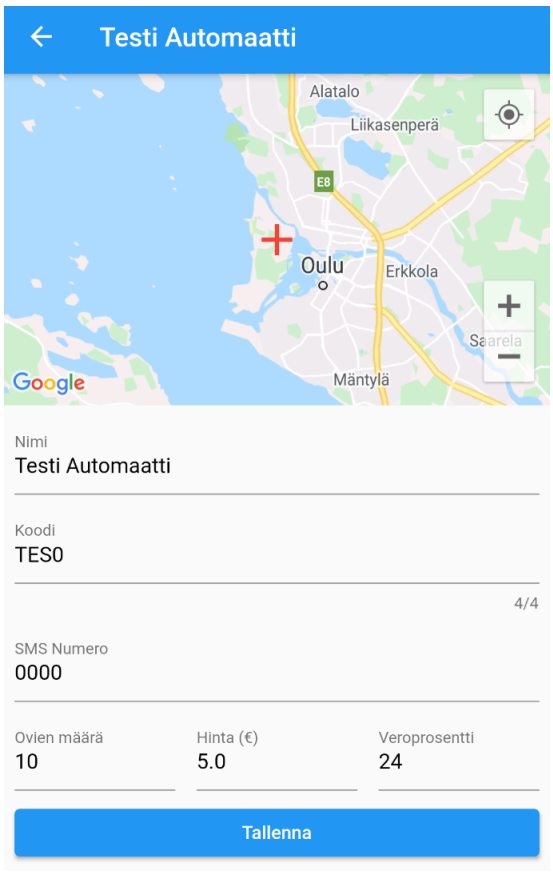


KUVA 12. Automaatin infosivu

4.4.2 Automaatin lisäys ja muokkaus

Automaatin lisäys- ja muokkaussivuina toimii sama sivu (kuva 13). Automaattia muokatessa sen tiedot täytetään valmiiksi kenttiin.

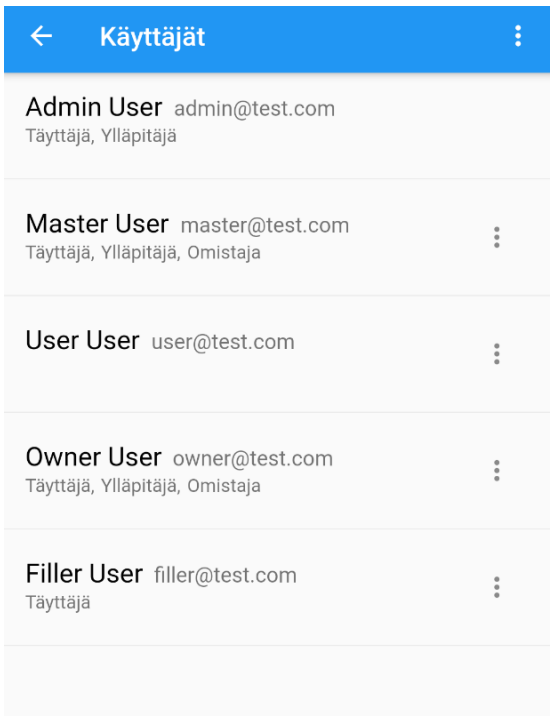
Automaatin sijainti asetetaan karttaa raahaamalla punaisen ristin kohdalle. Karttanäkymä on toteutettu Flutter-tiimin kehittämällä google_maps_flutter-kirjastolla.



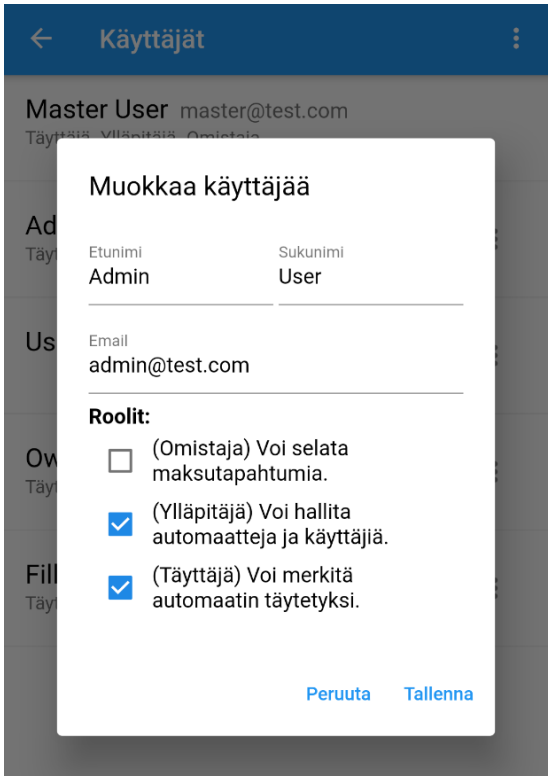
KUVA 13. Automaatin lisäyssivu

4.5 Käyttäjät-sivu

Käyttäjät-sivu listaa kaikki organisaation käyttäjät (kuva 14). Ylläpitäjät ja omistajat voivat lisätä uusia käyttäjiä ja antaa heille eri oikeuksia. Kun uusi käyttäjä lisätään, luo Firebasessa oleva funktio uuden käyttäjän annetulla sähköpostiosoitteella. Salasana on kaikille uusille käyttäjille sama ja se pyydetään vaihtamaan ensimmäisellä kirjautumisella. Käyttäjän nimeä ja oikeuksia voidaan myös muuttaa (kuva 15). Käyttäjä ei pysty kuitenkaan muuttamaan omia oikeuksiaan tai antamaan muille käyttäjille itseään korkeampia oikeuksia.



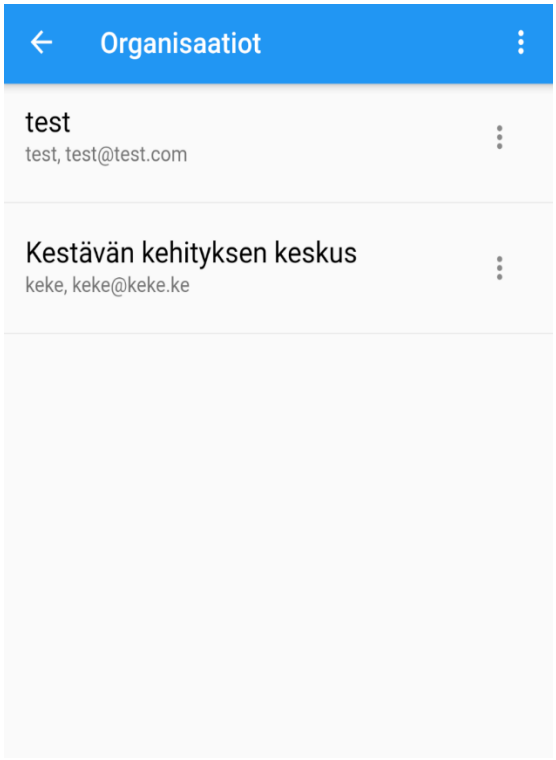
KUVA 14. Käyttäjien hallintasivu



KUVA 15. Käyttäjän muokkaus

4.6 Organisaatiot-sivu

Organisaatiot-sivulla master-käyttäjä voi luoda uuden organisaation asettamalla sen ja omistajan tiedot. Tämä luo uuden käyttäjän omistajan tiedoilla. Uudella organisaatiolla on omat käyttäjänsä ja automaatit. Eri organisaation käyttäjillä ei ole pääsyä muiden organisaatioiden käyttäjiin eikä automaatteihin. (Kuva 16.)



KUVA 16. Organisaatiot-sivu

5 JATKOKEHITYS

Polttoautomaattijärjestelmän ollessa vielä keskeneräisessä vaiheessa ylläpitosovelluksen päätarkoituksena on auttaa järjestelmän testauksessa ja suunnittelussa. Sovellukselta ei täten vielä odotettukaan täysin valmista toimintakykyä. Työn vaatimuksia määriteltäessä ja työn edetessä päätettiin jättää toteuttamatta ei-välttämättömiä toimintoja, joiden kehitystä voidaan jatkaa myöhemmin. Näitä ovat esimerkiksi maksutapahtumien selaus, salasanan vaihto sovelluksesta käsin, ilmoitukset puiden loppumisesta, tuki eri kielille ja parempi käyttöliittymä isommilla näytöillä.

Tietoturvaan tulisi kiinnittää erityistä huomiota etenkin, jos järjestelmää aletaan myöhemmin lisensoida muille toimijoille. Esimerkiksi tuki salasananhallinta-sovelluksille ja monivaiheisen tunnistautumisen lisääminen vähentäisi huomattavasti riskiä käyttäjätunnusten joutumista vääriin käsiin. Käyttäjien tekemisten tarkempi kirjaaminen lokiin puolestaan auttaisi selvittämään mahdollisia väärinkäytöksiä.

Koska Flutter on alustariippumaton alusta, voidaan sovelluksesta tehdä myös omat sovellukset esimerkiksi Android- ja iOS-alustoille. Kehitysvaiheessa sovellusta on jo testattu Android-emulaattorilla, mutta alustalle kääntäminen vaatisi vielä lisää testausta. Mobiilisovellukset mahdollistaisivat esimerkiksi ilmoitukset suoraan käyttäjän puhelimeen, kun automaatin polttopuut ovat vähissä.

Sovellusta testattiin manuaalisesti sovellusta käyttämällä. Jatkokehityksen aikana sovellusta kannattaisi testata ohjelmallisesti ainakin yksikkö- ja integraatiotestein. Näin pystyttäisiin varmistamaan ohjelman toimivuus muutoksien ja ominaisuuksien lisäyksen jälkeen.

6 POHDINTA

Työn tavoitteena oli kehittää web-sovellus, jolla voidaan hallita puuautomaattijärjestelmään kuuluvia automaatteja. Vähimmäisvaatimukset sovellukselle olivat automaattien lisääminen, muokkaus ja täytetyksi merkitseminen. Lisäksi sovelluksen käyttäjille haluttiin antaa erilaisia oikeuksia riippuen heidän työtehtävistään. Näiden tavoitteiden lisäksi järjestelmään tehtiin mahdollisuus lisätä uusia organisaatioita, joilla on hallittavanaan omat automaattinsa ja käyttäjät.

Opinnäytetyön alussa esiteltiin työssä käytettäviä Flutter- ja Firebase-alustoja. Käytännön osuudessa käytiin läpi sovelluksen suunnittelu- ja kehitysvaiheita sekä esiteltiin sovellusta sivu kerrallaan. Lopuksi pohdittiin, mihin suuntaan sovellusta voitaisiin kehittää jatkossa.

Työtä tehtiin pääasiassa itsenäisesti, mikä mahdollisti joustavat työskentelymenetelmät. Työn teossa sovellettiin ketteriä menetelmiä listaamalla tehtäviä ja antamalla niille tärkeysaste. Tehtäviä suoritettiin valitsemalla pari tärkeintä tehtävää kerrallaan, jotka tehtiin ja testattiin ennen uusien tehtäväien aloittamista. Menetelmä toimi pääosin hyvin, mutta vaati välillä hyvää itsekuria, jotta malttoi tehdä tehtävän loppuun saakka ennen uuden aloittamista.

Kokonaisuutena sovelluksen työstäminen oli haastava, opettava ja mielenkiintoinen projekti. Flutterin kanssa työskentely on mukavaa ja hyvä dokumentointi sekä monipuoliset kirjastot helpottavat erilaisten palvelujen, kuten Firebasen, liittämistä sovelluksiin. Paikoin koodista kasvoi vaikeasti hallittavaa, sillä Flutterissa ei tarvitse eriyttää käyttöliittymän ja sovelluksen logiikan koodia, minkä takia kooditiedostot paisuivat helposti liian isoiksi. Pilkkomalla koodia uusiin luokkiin ja funktioihin saatiin nämä ongelmat selvitettyä.

Lopputuloksena oli sovellus, joka täytti sille asetetut vaatimukset. Se käy hyvin järjestelmän kehittämiseen ja testaamiseen. Jos polttopuuautomaattijärjestelmän kehitys etenee käyttöönottovaiheeseen asti, tarvitsee sovellus joitain lisäominaisuuksia ja kattavampaa testausta.

LÄHTEET

1. Virolainen, Taina 2018. Suositulta Hietasaaren nuotiopaikalta varastetaan toistuvasti polttopuut – jokaisen ostettava itse makkaranpaistopuut. Kaleva. Hakupäivä 3.9.2021. https://www.kaleva.fi/suositulta-hietasaaren-nuotiopaikalta-varastetaan/1748608.

2. Flutter 2021. Flutter architectural overview. Hakupäivä 12.4.2021. https://flutter.dev/docs/resources/architectural-overview.

3. Dart. Publishing packages. Hakupäivä 30.4.2021. https://dart.dev/tools/pub/publishing.

4. Bose, Sudeepa 2020. Under the Lens - React Native vs Flutter vs Angular. Codemagic. Hakupäivä 4.8.2021. https://blog.codemagic.io/angular-vs-flutter-vs-react-native.

5. Demedyuk, Ihor & Tsybulskyi, Nazar 2020. Flutter vs React Native vs Native: Deep Performance Comparison. InVerita. Hakupäivä 4.8.2021. https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison.

6. Stevenson, Doug 2018. What is Firebase? The complete story, abridged. Medium. Hakupäivä 27.7.2021. https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0.

7. Petra, Tadas 2021. AWS Amplify vs Firebase for Flutter. Hakupäivä 4.8.2021. https://www.youtube.com/watch?v=qGElWORpD3c.

8. Google 2021. Cloud Firestore. Hakupäivä 26.4.2021. https://firebase.google.com/docs/firestore.

9. Google 2021. Firebase Authentication. Hakupäivä 26.4.2021. https://firebase.google.com/docs/auth.

10. Google 2021. Adding multi-factor authentication to your web app. Hakupäivä 4.8.2021. https://cloud.google.com/identity-platform/docs/web/mfa.

11. Google 2021. Cloud Functions. Hakupäivä 26.4.2021. https://firebase.google.com/docs/functions.

12. Google 2021. Firebase Hosting. Hakupäivä 26.4.2021. https://firebase.google.com/docs/hosting.

13. Kerpelman, Todd. 2019. Cloud Firestore Data Modeling (Google I/O'19). Hakupäivä 26.4.2021. https://www.youtube.com/watch?v=lW7DWV2jST0.

14. Google 2021. Fix insecure rules. Hakupäivä 10.6.2021. https://firebase.google.com/docs/firestore/security/insecure-rules.