OULUN AMMATTIKORKEAKOULU

Tommi Hirvonen

OVIKU-KULUNHALLINTAJÄRJESTELMÄN TOTEUTUS JA TESTAUS

OVIKU-KULUNHALLINTAJÄRJESTELMÄN TOTEUTUS JA TESTAUS

Tommi Hirvonen
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, Ohjelmistokehitys

Tekijä: Tommi Hirvonen
Opinnäytetyön nimi: Oviku-kulunhallintajärjestelmän toteutus ja testaus
Työn ohjaajat: Veijo Väisänen, Lasse Haverinen, Aki Marttila
Työn valmistumislukukausi ja -vuosi: syksy 2021 Sivumäärä: 31

Opinnäytetyön tilaajana oli Oviku Oy. Opinnäytetyön tarkoituksena oli luoda toimiva kulunhallintajärjestelmä, joka helpottaa suurien Oviku Nero -lukkoryhmien hallintaa. Oviku Nero on lisäosa, joka tekee oven lukosta älylukon.

Kulunhallintajärjestelmään kuuluu kaksi sovellusta, web-sovellus sekä Android-sovellus. Web-sovellus luotiin käyttäen HTML-kuvauskieltä ja JavaScript suorittaa sovelluksen toiminnallisuudet. Järjestelmän mobiilisovellus toteutettiin Android Studiolla Android-laitteille. Ohjelmointikielenä mobiilisovelluksessa käytettiin Javaa.

Kulunhallinta kulunhallintajärjestelmässä tapahtuu web-sovelluksella. Web-sovelluksen käyttäjät valitsevat hallintoalueista, mihin, paikkaan avain halutaan jakaa. Avaimen jako suoritetaan joko valmiiksi määritetylle kohdehenkilölle tai käyttäen vastaanottajan sähköpostia. Avaimen vastaanottaja saa avaimen mobiilisovellukseen, jonka jälkeen on mahdollista avata lukko Bluetooth-yhteydellä.

Opinnäytetyössä kohdattiin ongelmia, joita ei ehditty ratkaista. Tästä huolimatta kulunhallintajärjestelmä toimii, vaikkei kaikkiin tavoitteisiin päästy. Lopputuloksena kulunhallintajärjestelmällä on mahdollista jakaa avain kohdehenkilölle tiettyyn Oviku Nero älylukkoon. Avaimen vastaanottamisen jälkeen käyttäjän on mahdollista ottaa yhteys lukkoon mobiililaitteella ja avata ovi.

Asiasanat: hallintajärjestelmät, web-sovellukset, mobiilisovellukset, Bluetooth, älylukot

ABSTRACT

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

---

Author: Tommi Hirvonen
Title of thesis: Oviku access control system implementation and testing
Supervisors: Veijo Väisänen, Lasse Haverinen, Aki Marttila
Term and year when the thesis was submitted: Autumn 2021
Number of pages: 31

The client of the thesis was Oviku Oy. The purpose of the thesis was to create a functional access control system that facilitates the management of large Oviku Nero lock groups. Oviku Nero is an accessory that turns the door lock into a smart lock.

The access management system includes two applications. Web application as well as Android application. The web application was created using the HTML markup language and JavaScript executes the application's functionalities. The system's mobile app was implemented with Android Studio for Android devices. Java was used as the programming language in the mobile application.

Access management in the system is done with a web application. Web application users select the administration areas to which the key is to be shared. Key sharing is performed, either to a predefined target person or using the recipient's email. The recipient of the key receives the key to the mobile application, after which it is possible to unlock it via Bluetooth.

There were problems in the thesis that could not be solved. Nevertheless, the access control system works, although not all goals were achieved. As a result, the access control system makes it possible to distribute the key to the target person in a specific Oviku Nero smart lock. After receiving the key, it is possible for the user to connect to the lock on the mobile device and open the door.

Keywords: management systems, web applications, mobile applications, Bluetooth, smart locks

ALKUSANAT

Haluan kiittää opinnäytetyön tilaajaa Oviku Oy:tä ja sen koko henkilökuntaa saamastani tuesta, projektin aikana. Erityiskiitokset yrityksessä toimineelle esimiehelleni Aki Marttilalle, joka mahdollisti työsuhteeni tämän projektin parissa. Kiitän myös opinnäytetyön ohjaavia opettajia, jotka auttoivat opinnäytetyön loppuun viemisessä.

Olen kiitollinen Ovikulla kertyneestä kokemuksesta, joka toimii perustana alkavalleni uralle. Työskentely projektin parissa opetti minulle uusia teknologiota ja toimintaperiaatteita, joista varmasti on hyötyä myöhemmin.

Oulussa 7.10.2021

Tommi Hirvonen

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tilaajana oli Oviku Oy. Oviku on vuonna 2013 perustettu yritys, joka tuottaa kodin turvallisuustuotteita (1). Yrityksen lippulaivatuote on Oviku Nero, joka tuotiin markkinoille 2018 marraskuussa. Oviku Nero on jälkiasennettava lisälaite kodin oveen, joka tekee nykyisestä kodin lukosta älylukon ja mobiililaitteesta avaimen. Älylukkoa ja niiden avaimia hallitaan Oviku-sovelluksella, joka on ladattavissa Play-kaupasta sekä Apple Storesta. Applikaatiossa pääkäyttäjäksi määritellyt käyttäjät voivat seurata lukon lokitietoja sekä jakaa uusia avaimia. Avaimen jako internetin välityksellä on mahdollista kenelle tahansa halutulle henkilölle.

Projektin idea sai alkunsa tarpeesta laajentaa Oviku Neron myyntiä kerrostaloasuntoihin. Projekti toteutettiin ja suunniteltiin isännöitsijän ja huoltohenkilön esimerkkitilanteen ympärille ja sitä tullaan käyttämään tässä opinnäytetyössä. Opinnäytetyössä käydään läpi, kuinka kulunhallintajärjestelmä on rakennettu ja mitkä ovat sen pääpiirteet ja toiminnallisuudet. Opinnäytetyön on myös tarkoitus antaa lukijalleen valmiudet kulunhallintajärjestelmän käyttöön.

Projektin tarkoituksena oli rakentaa toimiva kulunhallintajärjestelmä, joka helpottaa suurien Oviku Nero -lukkoryhmien hallintaa. Hallintajärjestelmä helpottaa kerrostaloasunnoissa tapahtuvaa liikehdintää. Esimerkki käyttötapaan viitaten: Isännöitsijän on mahdollista jakaa rajoitetun käyttöoikeuden sisältävä avain huoltohenkilölle. Huoltohenkilö saa kyseisen avaimen mobiilisovellukseen. Kun isännöitsijä on jakanut avaimen, huoltohenkilöllä on mahdollisuus käydä suorittamassa huoltotöitä asuinnoissa, joihin hänellä on pääsy. Järjestelmä helpottaa kaikkia osapuolia kerrostaloasunnon huoltotilanteessa. Isännöitsijän sekä huoltohenkilön kummankaan ei tarvitsisi huolehtia konkreettisesta avaimesta, vaan kyseessä olisi ainoastaan digitaalinen avain, joka jaetaan mobiilisovellukseen. Asukas voi seurata lukon lokitietoja ja nähdä, milloin ovi on avattu. Tämä hyödyttää myös asukasta, sillä hän on tietoinen, milloin asunnossa on käyty.

2 PROJEKTIN TAVOITTEET JA VAATIMUKSET

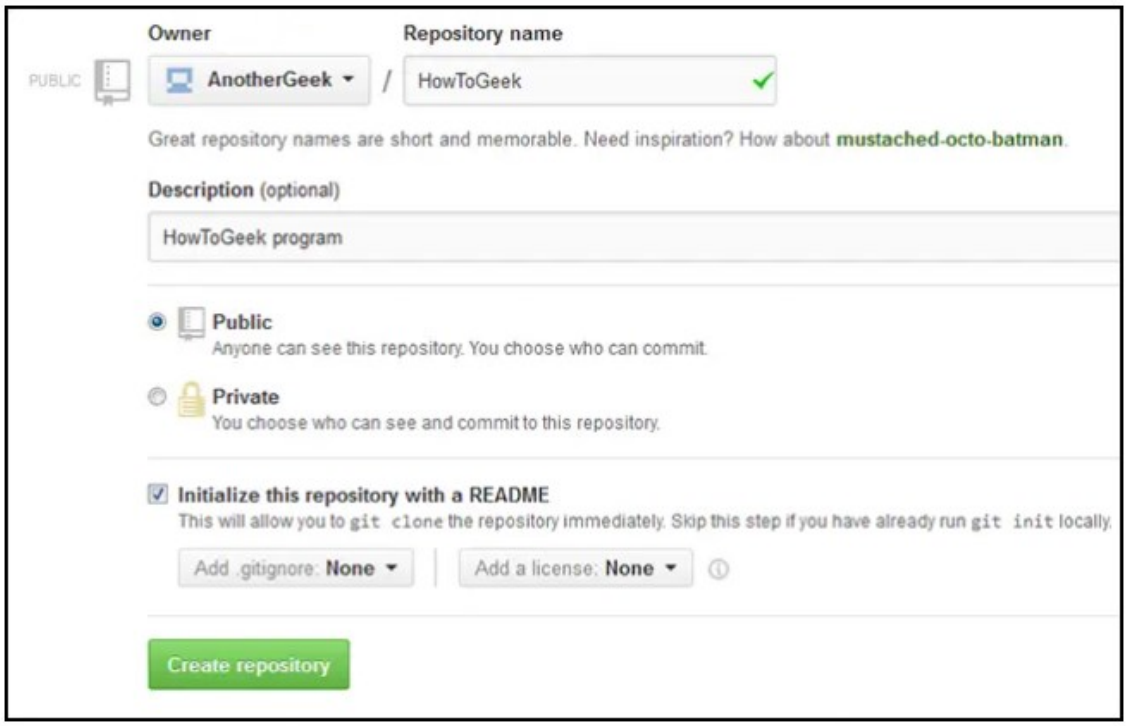
Aloitin työskentelyn Oviku Oy:llä maaliskuussa 2020 yrityslähtöisten tuotekehitysprojektien merkeissä. Näiden yrityslähtöisten tuotekehitysprojektien aikana aloitin suunnittelemaan ja toteuttamaan Oviku-kulunhallintajärjestelmää. Oviku-kulunhallintajärjestelmä toteutettiin ja saatettiin loppuun opinnäytetyönä.

Kulunhallintajärjestelmän luomisesta tuli ajankohtaista, jotta Oviku Nero-älylukon myyntiä voidaan laajentaa kerrostaloasumiseen. Ovikulla oli valmiiksi toimiva Android-sovellus, jolla yksityiset henkilöt pystyivät hallitsemaan omaa älylukkoaan. Projektin tavoitteet pystyi jakamaan kolmeen eri osaan. Ensiksi luodaan kokonaan uusi web-sovellus isännöinti tarkoituksiin. Johdetaan rinnakkaissovellus huoltohenkilöiden käyttöön, Ovikun toimivasta Android-sovelluksesta. Mukautetaan olemassa olevaa tietokantaa vastaamaan uuden web-sovelluksen ja mobiilisovelluksen tarpeisiin.

Web-sovelluksen vaatimuksena oli toteuttaa toiminnallisuudet masterkäyttäjille ja isännöitsijäkäyttäjille. Masterkäyttäjät tarjoavat hallinto-oikeudet isännöitsijäkäyttäjille. Isännöitsijäkäyttäjät jakavat avaimia haluttuihin kerrostaloasuntoihin kohdehenkilölle. Mobiilisovelluksen vaatimuksena oli johtaa helppokäyttöinen rinnakkaissovellus olemassa olevasta mobiilisovelluksesta. Mobiilisovelluksen täytyi vastaanottaa web-sovelluksesta jaettu avain, jonka jälkeen huoltohenkilö pääsee avaimella huoltokohteeseen. Vaatimuksena tietokannalle oli laajentaa ja mukauttaa se toimimaan web-sovelluksen ja Android-sovelluksen käyttötarkoitukseen.

3 VERSIONHALLINTA

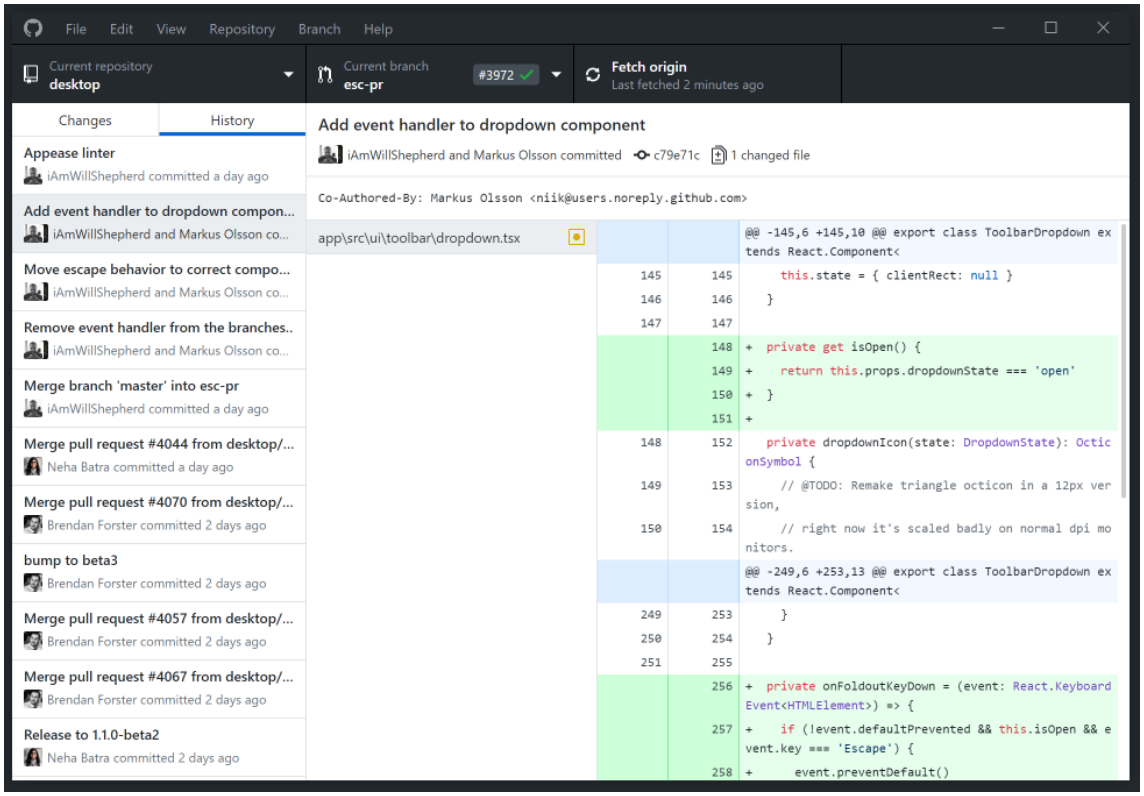
Kulunhallintajärjestelmän versionhallinta tapahtui GitHubissa. GitHub on graafinen käyttöliittymä Git-versionhallintajärjestelmälle. GitHub on pilvipalvelu ja työkalu, jonka avulla ohjelmistokehittäjät voivat ylläpitää ohjelmistokehitysprojekteja (2). Projektin alussa Oviku GitHub-käyttäjällä luotiin GitHub repositoryt järjestelmän web- ja mobiilisovellukselle (kuva 1). Repositoryt sisältävät sovelluksien lähdekoodin. Repositoryn luonnin jälkeen on mahdollista määrittää, ketkä kuuluvat kyseiseen projektiin ja mitkä ovat osallistujien oikeudet.



KUVA 1. Esimerkki GitHub repository luonnin näkymästä (3)

GitHub mahdollistaa usean kehittäjän yhtäaikaisen versionhallinnan samaan projektiin. Projektin versionhallinnassa käytin GitHub desktop-sovellusta (kuva 2), joka mahdollistaa helpon käytön ja graafisenkäyttöliittymän Git-komennoille. Versionhallinta GitHub desktopissa tapahtuu *pull*- ja *commit*-komennoilla. Kun projektiin on tehty muutoksia, jotka on todettu toimiviksi ja on syytä tallentaa, on myös hyvä tehdä commit-komento, jolla uusi ja toimiva koodi tallennetaan Gitiin. Kun commit suoritetaan hyväksytysti, tallentuu uusin projektiversio Gitiin. Pull-

komennolla taas puolestaan tuodaan uusi versio Gitistä. Pull-komennolla henkilöt saavat käyttöön uudet muutokset, joita projektiin on suoritettu commit-komennoilla.



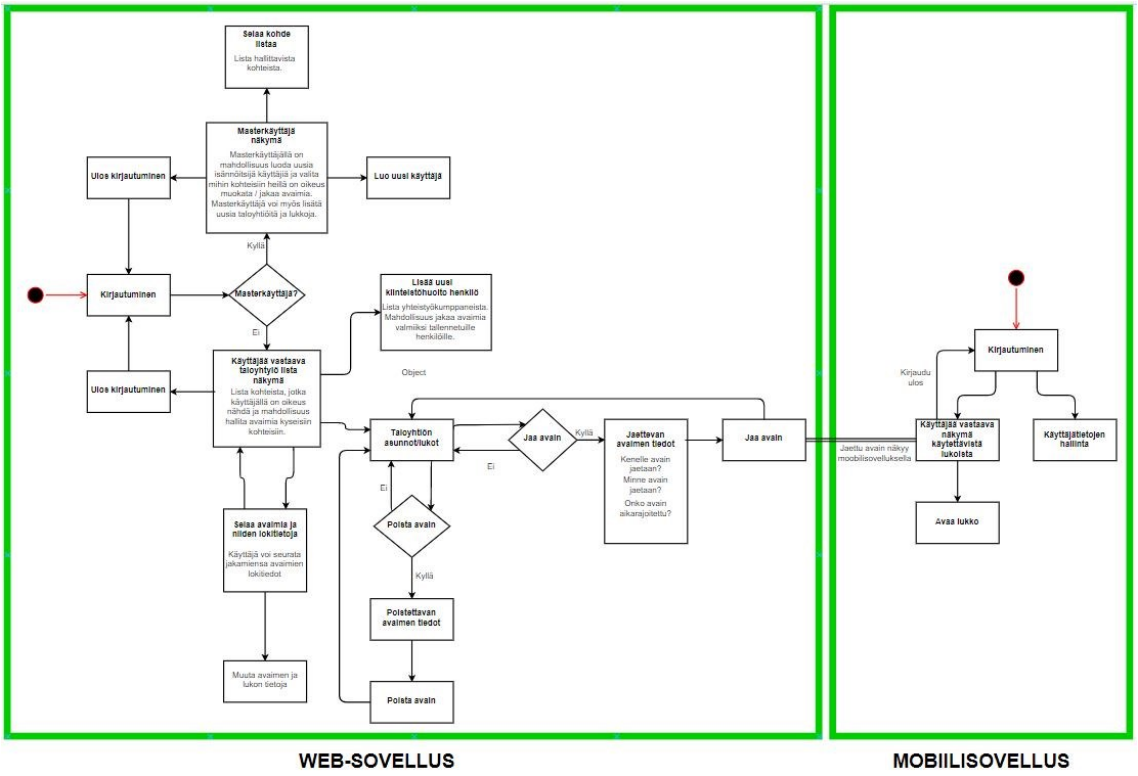
KUVA 2. GitHub desktop sovelluksen käyttöliittymä (4)

4 JÄRJESTELMÄKOKONAISUUS

Tämä luku jaetaan kahteen kappaleeseen, järjestelmärakenteeseen sekä järjestelmäkommunikaatioon. Järjestelmärakenne pitää sisällään käyttöliittymät ja niiden toiminnallisuudet. Järjestelmäkommunikaatiossa esitellään tietokanta- ja palvelintoteutusta.

4.1 Järjestelmärakenne

Kulunhallintajärjestelmä koostuu kahdesta sovelluksesta (kuva 3) sekä tietokanta- ja palvelintoteutuksesta. Projektin web-sovellus luotiin helpottamaan avainten jakoa suuriin Oviku Nero -lukkoryhmiin (kuva 4). Web-sovelluksen tehtävä on toteuttaa mahdollisemmin yksinkertainen avaintenhallinta isännöitäviin kohteisiin. Web-sovelluksesta isännöitsijät jakavat avaimia suoraan mobiilisovellukseen. Mobiilisovellus toimii avaimena, kunhan käyttäjä suorittaa kirjautumisen. Kirjautumisen jälkeen käyttäjällä on pääsy omiin käyttäjätietoihin sekä mahdollisuus avata niiden asuntojen ovi, joihin hänelle on jaettu voimassa oleva avain. Avaimen voimassaoloa voidaan rajoittaa esimerkiksi päivämäärä- ja kellonaika-asetuksilla.



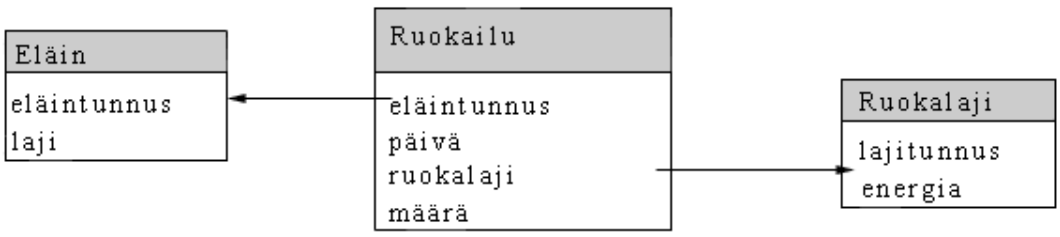
KUVA 3. Oviku-kulunhallintajärjestelmä vuokaavio (5)

KUVA 4. Oviku Nero -älylukko (1)

4.2 Järjestelmäkommunikaatio

Molemmat sovellukset rakennettiin toimimaan samassa tietokanta- ja palvelintoteutuksessa. Ovikun tietokanta- ja palvelintoteutus sijaitsee Google Cloud Platformissa. Googlen Cloud Platform on Googlen tarjoama pilvipalvelu. Google Cloud Platformissa on Compute Engine -toiminnallisuus, jolla voidaan luoda virtuaalisia ympäristöjä. Ovikun tietokanta- ja palvelintoteutus on toteutettu toiminnallisuutta käyttäen. Kulunhallintajärjestelmä toimii Ovikun testitietokannassa ja testipalvelimella. Testiympäristö on kopio olemassa olevista release-toteutuksista, joissa toimii myös Oviku-applikaatio. Testiympäristössä työskentely minimoi riskit ja antoi vapauden kokeilla riskialttiimpia toimintoja.

Ovikun tietokantana toimii PostgreSQL, joka on relaatiotietokanta. Relaatiotietokannalla tarkoitetaan tietokantaa, jonka sisältämän tiedon välillä on yhteyksiä. Relaatiotietokannassa on tauluja ja näiden taulujen sisällä kenttiä. Taulut määrittelevät kokonaisuuden, mihin kyseisen taulun kentät kuuluvat ja kentät sisältävät arvoja, jotka niille määritellään. Kuvassa 5 havainnollistetaan yksinkertaisen relaatiotietokannan välisiä yhteyksiä.



KUVA 5. Havainnollistava kuva relaatiotietokannan välisistä yhteyksistä (6)

Ovikun palvelin suorittaa funktioita, jotka on kirjoitettu Python-ohjelmointikielellä. Funktiot suorittavat SQL-komentoja, joilla tietokannasta haetaan haluttu tieto. SQL-komennot ovat komentoja SQL-tietokannalle. SQL noudattaa tiettyä syntaksia. Ohessa esimerkki yksinkertaisesta SQL-komennosta, jolla haetaan kaikki users-taulun sisältämät tiedot käyttäjistä, joiden asuinkaupunki on Oulu:

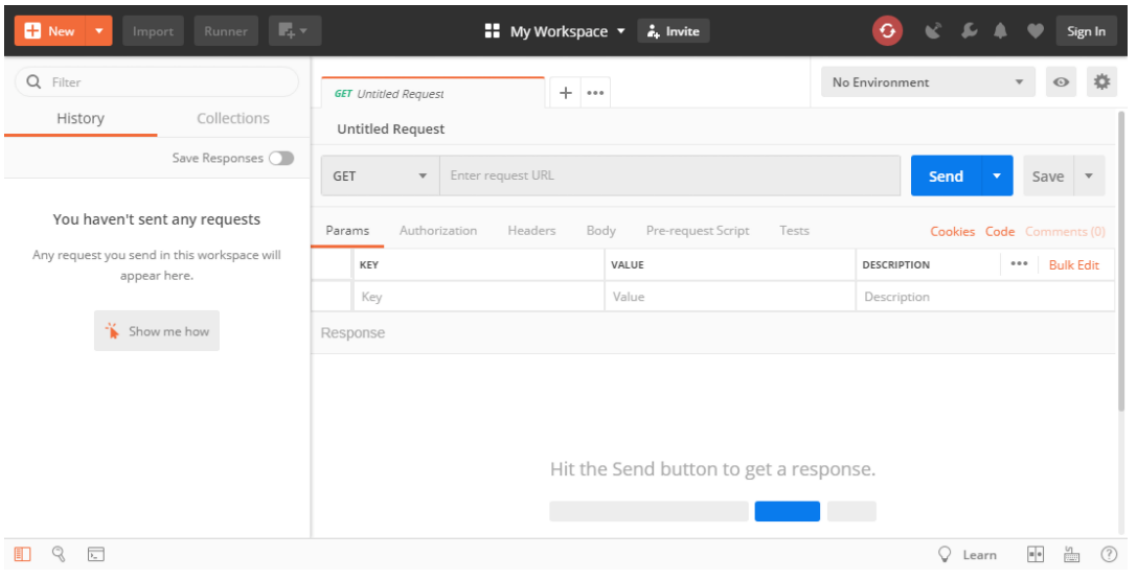
```
SELECT * FROM users WHERE city='oulu';
```

Palvelimen funktioiden suoritus tapahtuu Hypertext Transfer Protocol (HTTP) -pyynnöillä (7). HTTP on tiedonsiirtoprotokolla, jota WWW-selaimet sekä WWW-palvelimet käyttävät. HTTPS on HTTP:n salattu versio (7). HTTP-pyyntö sisältää aina URL-osoitteen, johon pyyntö lähetetään. Esimerkiksi joka kerta, kun käyttäjä avaa Googlen, hän lähettää HTTP-pyynnön osoitteeseen https://www.google.fi/.

Projektin web-sovelluksen HTTP-pyyntöjä ohjataan JavaScript-ohjelmointikielellä, kun taas mobiilisovelluksessa näitä pyyntöjä suoritetaan Java-ohjelmointikielellä. Järjestelmä pitää sisällään useita erilaisia pyyntöjä. Kunkin tiedon vastaanottamiseen vaaditaan tilanteeseen räätälöity HTTP-pyyntö. HTTP-pyyntö sisältää metodin sekä datan, mitä palvelimelle lähetetään. Järjestelmän käyttäjä lähettää pääosin HTTP-pyyntöjä palvelimelle käyttäen POST-metodia. POST-metodilla lähetetään dataa palvelimelle, jolla voidaan luoda tai päivittää resurssia (8). POST-metodin data lähetetään JSON (JavaScript Object Notation) -muodossa. Esimerkki yksinkertaisesta henkilö muuttujan JSON datasta:

```
var person = { name: "Tommi", age: 25, city: "Oulu" };
```

HTTP-pyyntöjä testattiin Postman sovelluksella (kuva 6). Postmanilla oli yksinkertaista testata ja todeta HTTP-pyyntö toimivaksi, ennen kuin se lisätään koodiin. Uusi tiedonhakuprosessi aloitettiin luomalla palvelimelle funktio, joka käsittelee HTTP-pyynnöllä lähetetyn datan ja hakee sen tietokannasta. Tämän jälkeen Postmaniin luotiin uusi HTTP-pyyntö. Pyynnölle annettiin URL-osoite, metodi ja body eli JSON-muodossa oleva data. Näiden argumenttien avulla Postmanista on mahdollista lähettää HTTP-pyyntö. Kun pyyntö lähetetään, palvelimella olevan funktion suoritus aloitetaan. Suoritusprosessi käsittelee vastaanotetun JSON-datan, hakee halutut tiedot tietokannasta, jonka jälkeen palauttaa HTTP-pyynnön lähettäjälle tiedot takaisin JSON-muodossa.



KUVA 6. Postman-sovelluksen käyttöliittymä (9)

Projektin aikana Postmanilla lähetettävä testipyyntö ei aina kuitenkaan toiminut. Postman antoi virheilmoituksen, jos jokin oli vialla palvelimella. Virheilmoituksen saatua oli helppo suunnata palvelimen virhelokitietoihin, joista vastaus ongelmiin lopulta löytyi. Kun Postman testipyyntö vastasi haluttua dataa, pystyttiin aloittamaan HTTP-pyynnön integrointi sovellukseen.

Web- ja mobiilisovelluksen välillä ei tapahdu suoraa kommunikaatiota. Molemmat sovellukset suorittavat tietokanta toimintoja, joilla tarvittavat operaatiot suoritetaan. Ensimmäisen kerran mobiilisovelluksen avaamisen yhteydessä, käyttäjä saa *clientId*-arvon, jonka avulla avain kohdistetaan oikealle henkilölle.

5 WEB-SOVELLUKSEN TOTEUTUS

Web-sovelluksen toteutus oli suoraviivainen. Suunnitteluvaiheen jälkeen oli selkeä visio, mitä toiminnallisuuksia sovellus tarvitsee ja kuinka ne toteutetaan. Sovellusta luodessa pidettiin koko ajan mielessä, kuinka luoda mahdollisimman toimiva ratkaisu avainten jakamiseen suuriin lukkoryhmiin.

5.1 Web-sovelluksessa käytetyt tekniikat

Web-sovelluksen näkymät luotiin käyttäen HTML-merkintäkieltä. HTML-toiminnallisuus perustuu tageihin. Eri nimisillä tageilla määritellään, mikä sovelluksen elementti on kyseessä. Seuraavanlaisesti voidaan luoda yksinkertainen näkymä websivusta HTML-tageilla:

```
<!DOCTYPE html>
<html>
<head>
<title>Web sivun nimi</title>
</head>
<body>
<h1>Otsikko 1</h1>
<p>Otsikon alapuolella oleva teksti.</p>
</body>
</html>
```

Projektissa käytettiin HTML-merkintäkielen lisäksi CSS-tyyliohjetta. CSS-tiedostoissa määritellään, miltä mikäkin HTML-tagin sisältämä asia näyttää ja mikä niiden tyyli on. CSS-tyyliohjetta käyttäen on helpompi ylläpitää sama tyyli läpi projektin. Seuraavanlaisesti voidaan antaa otsikolle 1 fonttityyli sekä pistekoko:

```
h1 {
  font-family: arial;
  font-size: 18px;
}
```

Web-sovelluksen toiminnallisuudet toteutettiin käyttäen JavaScript-ohjelmointikieltä. JavaScript-tiedostoissa käsitellään jokainen toimenpide, mitä sovelluksen käyttäjä suorittaa. Kyseessä voi olla esimerkiksi kirjautuminen, napin painallus tai datan vastaanottaminen. Suurin osuus JavaScript-toiminnoilla on datan käsittelyssä, siinä, kuinka se haetaan tietokannasta ja kuinka se esitetään käyttäjälle. Web-sovelluksen luonti vaati laajennuksen Ovikun olemassa olevaan tietokantaan ja palvelin toteutukseen. Tietokantaan lisättiin useampi uusi taulu ja taululle vaadittavat kentät. Muutoksiin sisältyi esimerkiksi taulu käyttäjien tietojen tallentamiseksi sekä taulu hallintoalueista, jotka kuuluvat kullekin käyttäjälle. Ovikun palvelin vaati myös muutoksia. Palvelimelle luotiin uusia funktioita, joita kutsutaan HTTP-pyynnöillä. JavaScript tiedostoista lähetetään HTTP-pyyntöjä palvelimelle, jonka jälkeen palvelimella oleva funktio käsittelee saadun datan ja hakee tietokannasta halutut tiedot. Kun tiedot saadaan tietokannasta, ne palautetaan JSON-muodossa, jonka jälkeen JavaScript-ohjelmat voivat käsitellä kyseisen datan ja esittää ne käyttäjille.

5.2 Käyttäjänäkymät

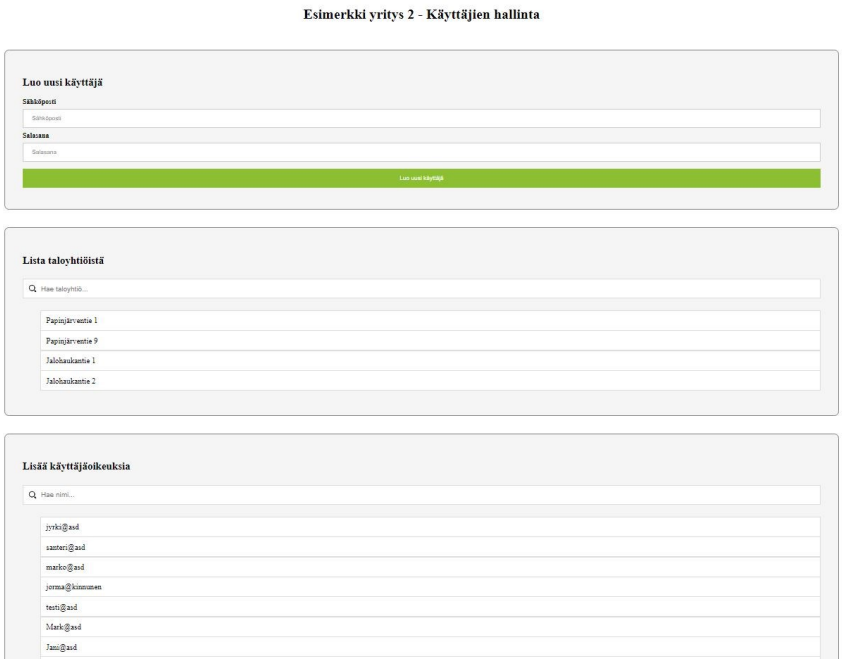
Tässä luvussa esitellään web-sovelluksessa olevat kaksi käyttäjänäkymää, masterkäyttäjänäkymä sekä isännöitsijäkäyttäjänäkymä. Molemmat näkymät luotiin omiin tarkoituksiinsa, selkeyttämään käyttäjien ja avainten hallintaa.

KUVA 7. Web-sovelluksen kirjautumissivu

Web-sovelluksen avautuessa käyttäjät ohjataan kuvan 7 mukaiselle kirjautumissivulle. Käyttäjän syöttäessä kirjautumistiedot hänet ohjataan käyttäjätietoja vastaavaan näkymään.

5.2.1 Masterkäyttäjänäkymä

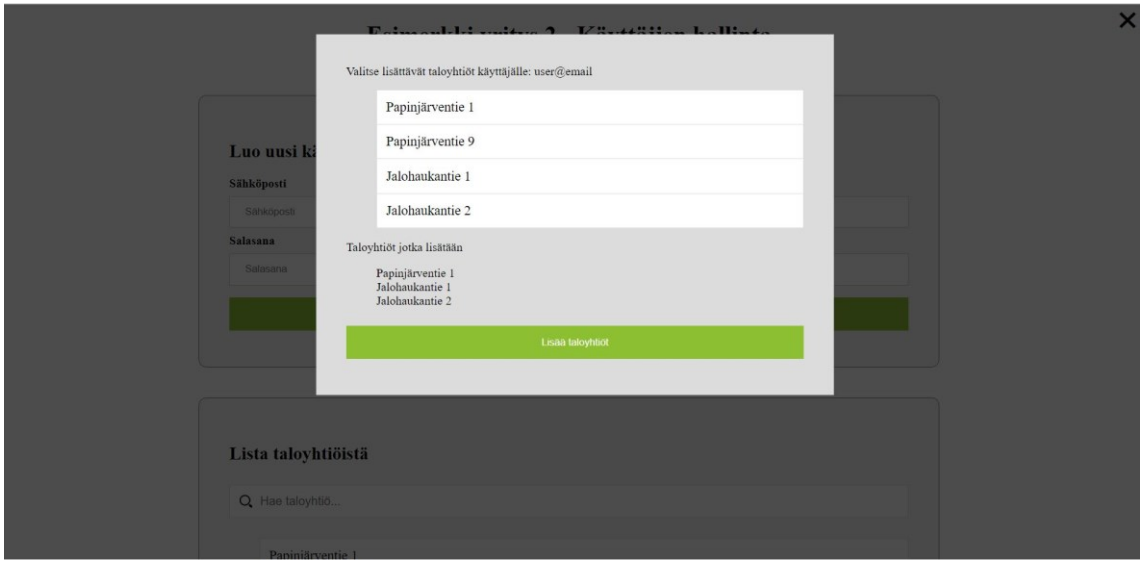
Masterkäyttäjät ovat oman yrityksensä käyttäjien hallinto-oikeuksista vastaavia käyttäjiä. Oviku luo masterkäyttäjätunnukset. Tunnuksen luonnin jälkeen käyttäjillä on mahdollista vaihtaa kirjautumistunnuksia (kuva 8).



KUVA 8. Masterkäyttäjä päänäkymä

Masterkäyttäjän kirjauduttua sisään hänelle avautuu kuvan 8 mukainen päänäkymä. Päänäkymä sisältää kolme erilaista toimintoa. Masterkäyttäjillä on mahdollisuus luoda uusia käyttäjiä, lisätä hallinto-oikeuksia käyttäjille tiettyihin taloyhtiöihin sekä lisätä uusia taloyhtiöitä. Kun masterkäyttäjä luo uuden käyttäjän, tietokannan käyttäjätietoja sisältävän taulun kenttään *master* lisätään oletusarvona *false.* Ovikun luoma masterkäyttäjä puolestaan omaa kyseisen arvon *true.* Tällä tavoin masterkäyttäjät sekä isännöitsijäkäyttäjät saadaan eriteltyä ja mahdollistettua heille pääsy eri näkymiin ja toiminnallisuuksiin.

Kun uusi käyttäjä luodaan, hänen tietonsa tulevat käyttäjälistaan, jonka jälkeen Masterkäyttäjällä on mahdollista jakaa hallinto-oikeuksia valitulle henkilölle tiettyihin taloyhtiöihin. Kuvassa 9 näkyy näkymä, jossa käyttäjälle jaetaan hallinto-oikeuksia taloyhtiöihin.



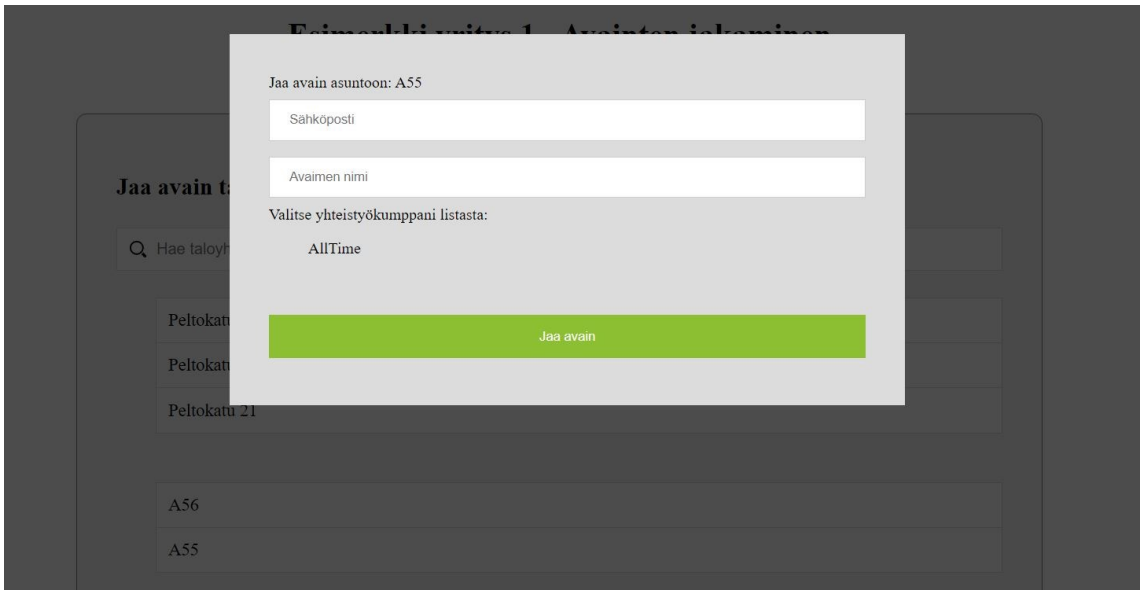
KUVA 9. Näkymä hallinto-oikeuksien lisäämisestä

5.2.2 Isännöitsijäkäyttäjänäkymä

Isännöitsijäkäyttäjät ovat vastuussa avainten hallinnasta. Kun henkilö kirjautuu isännöitsijäkäyttäjälle, hän saa kuvan 10 mukaisen päänäkymän. Päänäkymä sisältää listan taloyhtiöistä, joihin isännöitsijäkäyttäjä voi jakaa avaimia. Lista muodostuu masterkäyttäjän valitsemista hallinto-oikeuksista. Lista on yksilöllinen, jonka masterkäyttäjä määrittää kullekin isännöitsijäkäyttäjälle erikseen.

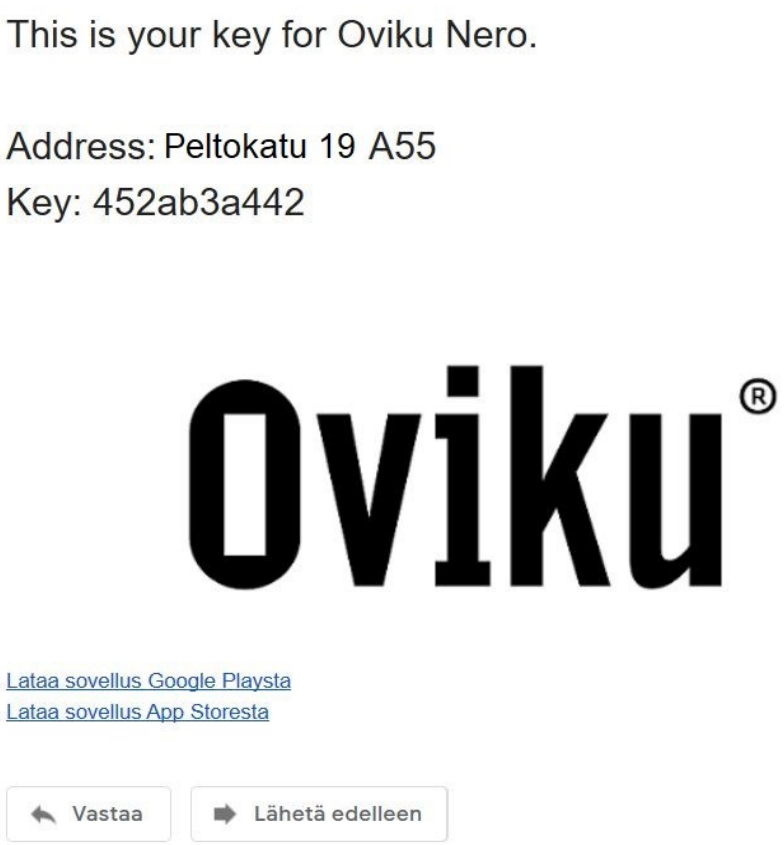
KUVA 10. Isännöitsijäkäyttäjän päänäkymä

Avainten hallinta on isännöitsijäkäyttäjien ainut toiminnallisuus. Isännöitsijäkäyttäjä klikkaa näkymässä olevaa taloyhtiötä halutessaan jakaa avaimen. Taloyhtiötä klikatessa aukeaa lista kyseisen taloyhtiön huoneistoista, joissa on Oviku Nero -älylukko, joihin isännöitsijäkäyttäjän on mahdollista jakaa avain. Huoneistoa klikattaessa aukeaa kuvan 11 mukainen näkymä.



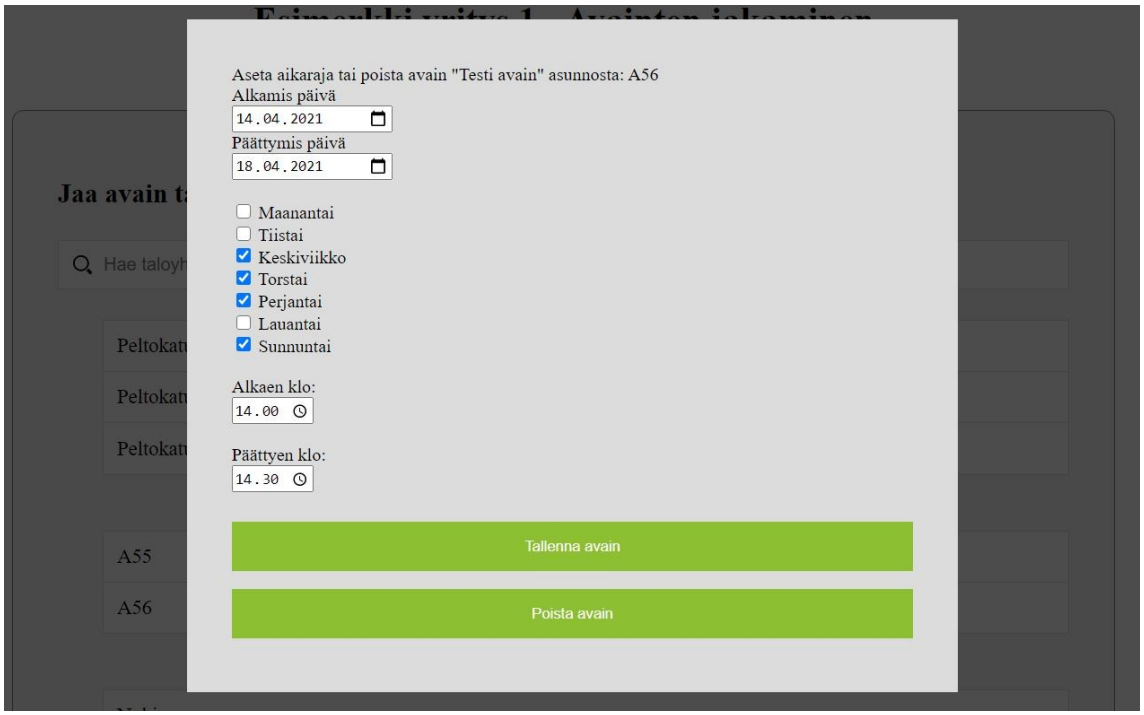
KUVA 11. Näkymä avainten jakamistoiminnosta

Avaimen voi jakaa joko sähköpostitse tai valmiiksi määritellylle kohdehenkilölle. Toiminnot poikkeavat toisistaan merkittävästi. Kun avain jaetaan käyttäjälle sähköpostiin, isännöitsijäkäyttäjän on kirjoitettava vastaanottavan henkilön sähköpostiosoite ja annettava jaettavalle avaimelle nimi. Tämän jälkeen kohdehenkilö saa kuvan 12 mukaisen sähköposti viestin, jossa on avaimen tiedot ja osoite, johon avain käy. Jos kohdehenkilöä ei ole valmiiksi määritelty ja hänelle jaetaan avain sähköpostitse, hän joutuu tekemään manuaalisen avaimen lisäämisen mobiilisovelluksessa. Kun valitaan listasta valmiiksi järjestelmään tallennettu kohdehenkilö, hän saa avaimen suoraan mobiililaitteeseen mobiilisovelluksen sisältämän *clientId:n* avulla. Avaimen saatuaan kohdehenkilö voi muodostaa mobiililaitteella yhteyden älylukkoon ja avata oven.



KUVA 12. Näkymä sähköpostiviestin vastaanottajalle jaetusta avaimesta

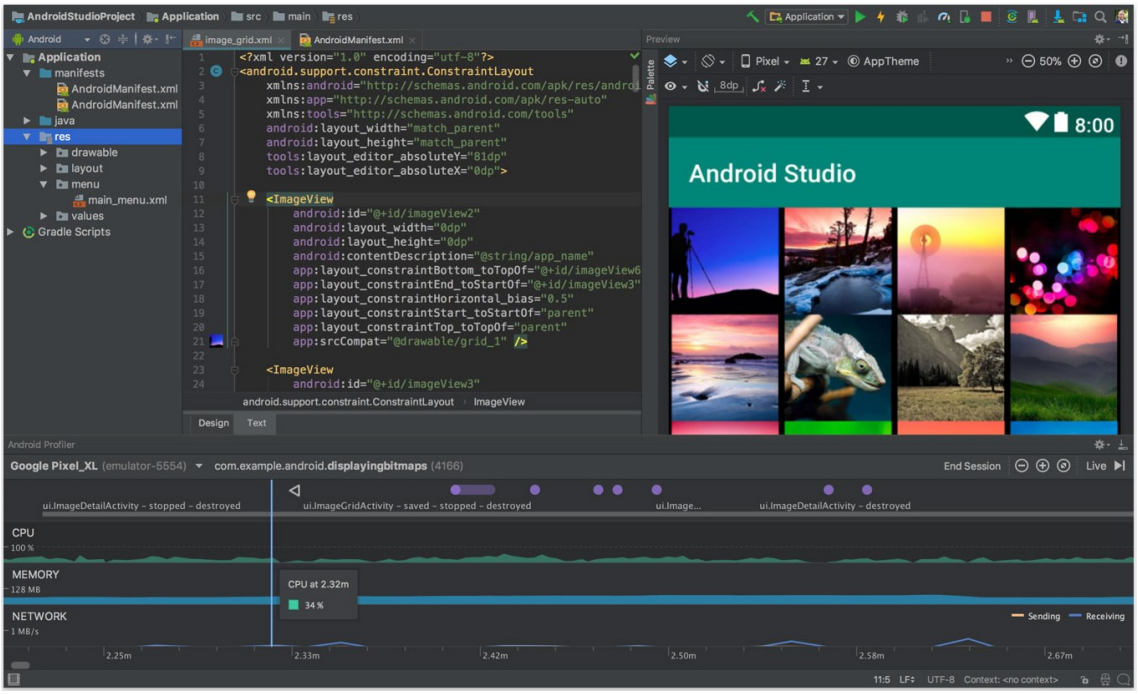
Isännöitsijäkäyttäjän on mahdollisuus poistaa kokonaan avain tai muokata sen tietoja, jotka on jaettu web-sovelluksen kautta tiettyyn asuntoon. Kun isännöitsijäkäyttäjä katsoo kohteen tietoja, hän voi jakaa uuden avaimen ja näkee listan jaetuista avaimista. Avainta klikkaamalla pääsee mahdollisiin toiminnallisuuksiin, joita avaimelle voi suorittaa (kuva 13). Toiminnallisuuksiin kuuluu uuden avaimen luominen, aikarajoituksen asettaminen ja avaimen poistaminen. Aikarajoitus avaimelle asetetaan valitsemalla ensin päivämäärät, jolloin avain on voimassa. Seuraavaksi avaimelle määritetään viikonpäivät ja kellonaika, minkä välisenä aikana avain toimii. Vaikka kalenteri ajankohta avaimelle olisi yksi viikko, mutta valittuna viikonpäivinä on vain maanantai ja perjantai, avain on voimassa vain kyseisinä viikonpäivinä. Jos haluaa luoda avaimen, joka on voimassa halutulla viikolla maanantaina ja perjantaina keskenään eri kellonaikoina, on käyttäjän jaettava kaksi eri avainta saavuttaakseen kyseisen toiminnon.



KUVA 13. Näkymä avaimelle tehtävistä toiminnoista. Avaimelle voi asettaa aikarajoituksen toimimiselle tai poistaa kokonaan.

6 MOBIILISOVELLUKSEN TOTEUTUS

Mobiilisovellus toteutettiin Android Studiolla. Android Studio on natiivi ohjelmointiympäristö Android-laitteille (kuva 14). Sovellus päätettiin johtaa muokkaamalla olemassa olevaa Oviku-applikaatiota. Tähän päädyttiin helpottaakseen sovelluksen toteuttamista. Oviku-applikaatio pitää sisällään useita toimivia komponentteja, joita mobiilisovelluksen toteutus vaati, kuten esimerkiksi lukon ja mobiililaitteen välinen kommunikaatio. Tästä syystä mobiilisovelluksen muokkaaminen oli parempi ratkaisu kuin alkaa luomaan koko järjestelmää alusta.



KUVA14. Android Studion käyttöliittymä (10)

6.1 Sovelluksen rakenne

Sovelluksen suunnitteluvaiheessa oli selvää, että rakenne pidetään lähes samana kuin Oviku-applikaatiossa. Sovelluksen näkymät tarjotaan Activityillä. Activity on luokka Android Studiossa. Activity tarjoaa näkymän, mihin sovellus piirtää

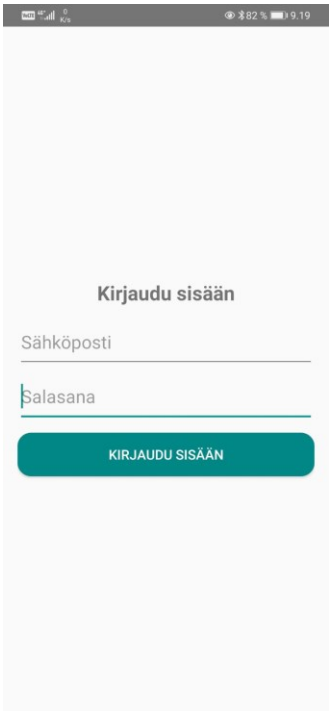
sen käyttöliittymän (11). Sovelluksessa on kaksi Activityä, Log In Activity ja Devices Activity. Nämä kaksi Activityä vastaavat koko sovelluksen näkymästä. Activityjen lisäksi sovelluksessa on Fragment-tiedostoja ja XML-tiedostoja. Fragment-tiedostot ovat näkymiä, jotka kutsutaan ja käsitellään toisessa Activityssä. Fragmentit pitävät sisällään oman näkymän ja toiminnallisuudet. Kirjautuminen eli Log In Activity ei kutsu ainuttakaan Fragment-tiedostoa. Kaikki sovelluksessa olevat näkymät käsitellään ja suoritetaan Devices Activityn kautta. Fragmentit vaikuttavat sovellukseen kokoon sekä suorituskykyyn. Fragmenttejä käyttämällä sovelluksesta saadaan tehtyä kevyt ja nopea. Sovelluksen näkymät määritellään XML-tiedostoissa. XML-tiedostot pitävät sisällään XML-kuvauskieltä. Kaikille sovelluksen Activityille ja Fragmenteille on niitä vastaava XML-tiedosto, missä määritellään, miltä kyseinen näkymä näyttää. Seuraavanlaisesti voidaan kuvata XML-tiedostossa tekstinäkymä ja nappi:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/an-
droid"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

(12).

Android Studio tarjoaa XML-tiedostojen muokkaamiseen myös graafisen käyttöliittymän. Kehittäjällä on mahdollisuus valita, haluaako hän muodostaa näkymän kirjoittamalla kuvauskieltä itse vai käyttämällä drag and drop -toimintoja graafisessa käyttöliittymässä. Graafisen käyttöliittymän kautta asetetut elementit luovat automaattisesti XML-kuvauksen, jota voi myös halutessaan muokata.

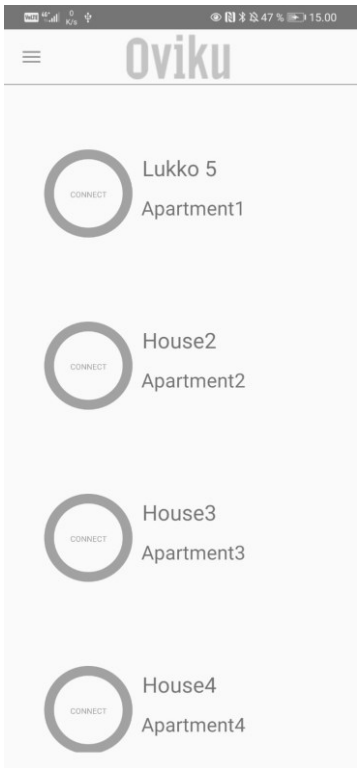
Sovelluksen kaksi Activityä tarjoavat käyttäjälle sovelluksen pääominaisuudet. Kirjautumisnäkymä (kuva 15) suorittaa kirjautumisen ja palauttaa tietokannasta kirjautuneen henkilön käyttäjätiedot sekä avaimet. Devices Activity on sovelluksen päänäkymä ja se myös sisältää sovelluksen päätoiminnallisuudet kuten lukon avaamisen. Devices Activityssä tapahtuu myös fragmenttien suoritus ja kutsut. Esimerkki fragmentteja Devices Activityssä ovat näkymä toimintoineen lukon poistamiselle sekä lukkojen lisäämiselle.



KUVA 15. Mobiilisovelluksen kirjautumisnäkymä

6.2 RecyclerView-komponentti

Mobiilisovelluksen päätoiminnallisuutena on listanäkymä, josta käyttäjä voi avata lukkoja, joihin hänellä on voimassa oleva avain. Listanäkymä toteutettiin käyttäen Android Studion komponenttia RecyclerView (13). RecyclerView tarkoittaa suomeksi kierrätysnäkymää. Nimensä mukaan se kierrättää näkymää eikä tuhoa elementtiä, kun se vieritetään ulos ruudusta, vaan varaa sen tilan seuraaville elementeille. Tällä tavoin RecyclerView takaa hyvän suorituskyvyn listanäkymälle.



KUVA 15. Mobiilisovelluksen päänäkymä. RecyclerView, jossa voimassa olevat avaimet.

RecyclerView-näkymä suunniteltiin toimimaan mahdollisimman käyttäjäystävällisesti. Huoltohenkilö voi yksinkertaisesti nähdä listasta kaikki lukot, joihin hänellä on voimassa oleva avain. Listan toiminnallisuuksiin suunniteltiin useita ominaisuuksia, joita ei kuitenkaan ehditty saattaa loppuun tämän raportin valmistumishetkellä.

7 JATKOKEHITYS

Sovelluskehitysprojektissa on iso työ saada se julkaisukelpoiseksi. Kulunhallintajärjestelmä toteutettiin vaatimusten mukaan, ja suurin osa vaatimuksissa määritellyistä komponenteista saatiin valmiiksi. Jatkokehitykselle jää silti vielä tilaa näin laajassa projektissa. Järjestelmää testatessa havaittiin useita ohjelmointivirheitä, jotka pitää vielä korjata. Suurin osa ohjelmointivirheistä liittyy web-sovelluksen käyttöliittymään. Kulunhallintajärjestelmän loppuun saattamiseksi olisi myös hyvä saada testiryhmä järjestelmän kohderyhmästä. Testiryhmältä voisi saada hyviä näkemyksiä, mitä järjestelmästä puuttuu ja minkä asian olisi voinut toteuttaa toisin.

7.1 Web-sovelluksen jatkokehitys

Web-sovellusta testatessa todettiin virheitä käyttöliittymän ilmentämisessä. Graafinen puoli käyttöliittymässä tarvitsee päivitystä, jotta se on käyttäjäystävällinen eikä jätä epäselvyyksiä sovelluksen käyttäjälle tämän navigoidessa eri toiminnallisuuksien välillä. Myös osa toiminnallisuuksista on vielä kesken, jotka sovellukseen on saatava. Tällaisia toimintoja ovat esimerkiksi taloyhtiön hallintaoikeuden vaihtaminen eri isännöintipalvelulle.

Projektin vaatimuksiin sisältyi lokitietojen katsaus. Ovikun olemassa oleva applikaatio mahdollistaa lokitietojen seuraamisen. Näin ollen asukas pystyisi seuraamaan applikaatiosta, milloin huoltohenkilö on käynyt asunnossa. Lokitietojen selaaminen toimintona toimisi siten, että isännöitsijäkäyttäjät voivat nähdä lokitietoja, kun asunnon ovi on avattu heidän jakamallaan avaimella. Isännöitsijäkäyttäjät eivät kuitenkaan voisi seurata asukkaan liikehdintää asunnossa.

Suunnitteluvaiheessa määriteltiin isännöitsijäkäyttäjällä olevan mahdollisuus hallita taloyhtiöitä. Taloyhtiöiden hallinnalla tarkoitetaan sitä, että jos jonkun taloyhtiöin kanssa isännöintisopimus loppuu, isännöintikohde voitaisiin poistaa isän-

nöitsijäkäyttäjän hallittavista kohteista. Tähän toimintoon kuuluisi myös taloyhtiöiden yksittäisten asuntojen hallinta. Esimerkiksi jos asukas ei olisi halukas antamaan isännöitsijällä lupaa jakaa avaimia hänen asuntoonsa. Tämä mahdollisuus olisi suotava asukkaalle ja masterkäyttäjä joutuisi poistamaan asunnon hallittavista kohteista.

7.2 Mobiilisovelluksen jatkokehitys

Mobiilisovelluksen päänäkymässä on raportin valmistumishetkellä todettu ongelmia. Näistä ongelmista on raportoitu ja tehty korjaussuunnitelmat. Listanäkymä ei toimi kuten määriteltiin. Web-käyttöliittymästä on mahdollisuus jakaa avain kohdehenkilölle, mutta kohdehenkilön on lisättävä kyseinen lukko manuaalisesti automaattisen lisäyksen sijaan. Lukon lisäämisen jälkeen kohdehenkilön on kuitenkin mahdollista avata lukko voimassa olevalla avaimella.

Mobiilisovellukselle on suunniteltu useita jatkokehitystoimintoja, joita ei ehditä toteuttamaan tämän opinnäytetyön merkeissä. Mobiilisovelluksen päänäkymässä yksi jatkokehityskohde on graafinen ilmoitus niistä lukoista, jotka ovat kuuluvuusalueella. Lukot, joihin olisi mahdollista ottaa yhteys, muuttuisivat vihreiksi ja ovi olisi mahdollista avata painamalla nappia. Mobiilisovellus ei ole kaukana julkaisuvalmiudesta, mutta julkaisun jälkeen sovellus tarvitsee ylläpidon havaittavien virheiden korjaamiseksi.

8 YHTEENVETO

Sovelluskehitysprojektin saattamisessa julkaisukelpoiseksi on valtava työ. Sovellukset ovat pääosaltaan toimivia, mutta jatkokehitykselle jää silti varaa. Opinnäytetyön tavoitteeksi määriteltiin saada kulunhallintajärjestelmä toimimaan. Tavoitteisiin päästiin niiltä osin, että kommunikaatio sovelluksien välillä toimii ja järjestelmän päätoiminnallisuus on kunnossa. Opinnäytetyössä kohtasin onnistumisia sekä ongelmia. Ongelmia saatiin selvitettyä, mutta valitettavasti osa jää jatkokehitykseen.

Mielestäni opinnäytetyön laajuus 15 opintopistettä on vähän. Sovelluskehitysprojektia rakentaessa opinnäytetyön opintopistemäärä täyttyy hetkessä ja tekemistä silti riittäisi. Opinnäytetyön aikana opin valtavan määrän uusia taitoja, joita voin varmasti hyödyntää tulevaisuudessa. Koin itselleni haastavaksi opinnäytetyön raporttiosuuden. Opinnäytetyön jäsentely siten, että se pysyy selkeänä ja lukijalle helposti ymmärrettävänä, oli hankalaa. Raporttiosuus kuitenkin mielestäni opastaa lukijaa ja antaa valmiuden käyttää kulunhallintajärjestelmää.

Aloittaessani työt Ovikulla kulunhallintajärjestelmän parissa tehtäviksi määriteltiin kahden sovelluksen luonti, tietokanta- ja palvelintoteutukset sekä dokumentaatio. Ilokseni voin todeta, että opinnäytetyön merkeissä nämä tehtävät on saatu päätökseen. Opinnäytetyön valmistumishetkellä kulunhallintajärjestelmän päätoiminnallisuudet on toteutettu ja todettu toimiviksi. Kulunhallintajärjestelmä toimii Ovikun testiympäristöissä ja on viimeisiä muutoksia vailla valmis siirrettäväksi release-ympäristöön.

LÄHTEET

1. Oviku. 2021. Saatavissa: https://oviku.com/. Hakupäivä 11.2.2021.
2. --local-branching-on-the-cheap. 2021. Git. Saatavissa: https://git-scm.com/. Hakupäivä 21.4.2021.
3. Brown, Korbin. 2019. What Is Github, and What Is It Used For?, How-To Geek. Saatavissa: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/. Hakupäivä 21.4.2021.
4. Github Desktop. 2021. Github. Saatavissa: https://desktop.github.com/. Hakupäivä 21.4.2021.
5. Draw.io. 2021. Saatavissa: https://app.diagrams.net/. Hakupäivä 15.2.2021.
6. Laine, Harri 2005. Tietokantojen perusteet. Helsingin yliopisto, tietojenkäsittelytieteen laitos. Saatavissa: https://www.cs.helsinki.fi/u/laine/tkp/relaatiomalli/rakenne.html. Hakupäivä 22.4.2021.
7. HTTP. 2020. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/HTTP. Hakupäivä 23.4.2021.
8. HTTP Request Methods. 2021. W3Schools. Saatavissa: https://www.w3schools.com/tags/ref_httpmethods.asp. Hakupäivä 23.4.2021.
9. Postman Tutorial: How to use Postman Tool for Api Testing. 2021. Guru99. Saatavissa: https://www.guru99.com/postman-tutorial.html. Hakupäivä 15.4.2021.
10. Android Studio. 2021. Android. Saatavissa: https://developer.android.com/studio. Hakupäivä: 21.4.2021.
11. Introduction to Activities. 2021. Android. Saatavissa: https://developer.android.com/guide/components/activities/intro-activities. Hakupäivä: 21.4.2021.
12. Layouts. 2021. Android. Saatavissa: https://developer.android.com/guide/topics/ui/declaring-layout. Hakupäivä 21.4.2021.
13. Create dynamic lists with RecyclerView. 2021. Android. Saatavissa: https://developer.android.com/guide/topics/ui/layout/recyclerview. Hakupäivä 21.4.2021.