Marko Lehtola

OmaFillari
Android mobiilisovellus

Opinnäytetyö
Tieto- ja viestintätekniikka

2020

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	**Tutkinto**	**Aika**
Marko Lehtola	Insinööri (AMK)	Syyskuu 2020
Opinnäytetyön nimi OmaFillari Android-mobiilisovellus		40 sivua
Toimeksiantaja Kaakkois-Suomen amattikorkeakoulu, Gamelab		
Ohjaaja Marko Oras		

Tiivistelmä

Tämän opinnäytetyön tavoitteena oli dokumentoida mobiilisovelluksen sekä sen taustajärjestelmän suunnittelu ja toteutus Android-alustalle. Sovelluksen nimi on OmaFillari ja sen tarkoituksena on helpottaa käyttäjää löytämään hänen polkupyöränsä, jos se on kadonnut. Sovellukseen lisätään tarkka kuvaus polkupyörän tiedoista sekä valokuvia, joista sen tunnistaa. Näillä tiedoilla heitteille jätetyn polkupyörän löytäjän on mahdollista selvittää polkupyörän omistaja ja lähettää hänelle löytymisilmoitus, johon liitetään löytyneen polkupyörän sijainti.

Opinnäytetyössä käytettiin toteutuspainotettua lähestymistapaa. Tavoitteena oli luoda Android-mobiilisovellus, sekä pilvipohjainen taustajärjestelmä, johon sovellus tallentaa käyttäjän syöttämät tiedot. Taustajärjestelmän toteutuksessa käytettiin hyväksi Google Firebase -pilvipalvelua, johon luotiin pilvipohjainen reaaliaikainen tietokanta sovelluksen tarvitsemille tiedoille. Sovelluksen käyttöliittymä suunniteltiin mahdollisimman yksinkertaiseksi ja intuitiiviseksi.

Tuloksena syntyi toimintakykyinen ja nykyaikainen mobiilisovellus, jolla voidaan käsitellä taustajärjestelmän tietoa ja joka on valmis julkaistavaksi Google-sovelluskaupassa hyvin pienillä muutoksilla.

Asiasanat

mobiilisovellus, Android, Firebase, pilvipalvelin

Author (authors)	Degree	Time
Marko Lehtola	Bachelor of Engineering	September 2020
Thesis Title OmaFillari Mobile Application		40 pages
Commissioned by South-Eastern Finland University of Applied Sciences, Gamelab		
Supervisor Marko Oras		

Abstract

The aim of this thesis was to document the design and implementation of a mobile application, as well as its background system for the Android platform. The app was called OmaFillari and it was designed to make it easier for the user to find his or her bike if it is lost. An accurate description of the bicycle information, as well as photos of bicycle can be added to the application. With this information the finder of the discarded bicycle can find the owner of the bicycle and send him a discovery message. Location of the found bicycle can be added to the message.

An implementation-oriented approach was used in the thesis. The goal was to create an Android mobile app with a cloud-based background system in which the app would store user-entered data. The back-end system was implemented using the Google Firebase cloud service. A cloud-based real-time database of information was created for the application. The interface of the application was designed to be as simple and intuitive as possible.

The result was a functional and modern mobile application that can handle background information and was ready to be published in the Google Apps store with very few changes.

Keywords

mobile application, Android, Firebase, cloud server

SISÄLLYS

1 JOHDANTO

Tässä opinnäytetyössä päätavoitteena on dokumentoida OmaFillari-mobiilisovelluksen suunnittelun ja toteutuksen vaiheet, sekä sovelluksen testaus. Sovelluksen tarkoituksena on auttaa kadonneen polkupyörän palautumista sen oikealle omistajalle. Opinnäytetyö tehdään toimeksiantajalle, joka on Kaakkois-Suomen ammattikorkeakoulun Kotkan kampuksella toimiva Gamelab.

Työssä hyödynnetään Google Cloud Firestore -tietokantapalvelua, jota sovellus käyttää taustapalvelunaan. Kehitysympäristönä toimii Android Studio. Ohjelmointikielenä käytetään Java-kieltä.

Opinnäytetyö koostuu kolmesta osasta. Ensimmäisessä osassa käsitellään mobiilisovelluksen suunnittelu- ja teoriaosuus. Toisessa osassa käsitellään mobiilisovelluksen toteutus. Kolmannessa ja viimeisessä osiossa käydään läpi sovelluksen testausvaiheessa saadut tulokset, sekä analysoidaan sovelluksen toteutuksen laatua ja saavutettiinko opinnäytetyön asetetut tavoitteet. Lisäksi osiossa arvioidaan onnistumista omasta näkökulmasta. Lopuksi osiossa pohditaan sovelluksen jatkokehitystä.

2 PEREHTYMINEN

Tarkoituksena on tehdä mobiilisovellus, jolla asiakas voi lisätä polkupyöränsä tuntomerkit tietokantaan pilvipalvelimelle. Sovelluksella on mahdollista ilmoittaa polkupyörä kadonneeksi. Sovelluksen nimeksi valittiin ”OmaFillari”. Edellä mainittu nimi valittiin, koska nimi itsessään kertoo sovelluksen käyttötarkoituksen liittyvän polkupyöriin. Harkinnan alla olivat myös nimet ”OmaPyörä” ja ”Oma Polkupyörä”, mutta nimessä päätettiin käyttää vain ASCII-merkkejä.

Sovelluksen tarkoituksena on, että polkupyörän kadotessa tai polkupyörävarkauden tapahtuessa sovelluksella on mahdollista ilmoittaa polkupyörä kadonneeksi. Selvästi heitteille jätetyn polkupyörän löytyessä pyörän löytäjällä on mahdollisuus etsiä polkupyörän tietoja sovelluksen avulla. Sovellus palauttaa polkupyörän ja omistajan tiedot tietokannasta, jos kyseisen

polkupyörän omistaja on ilmoittanut polkupyöränsä kadonneeksi. Jos polkupyörää ei ole ilmoitettu kadonneeksi, käyttäjän yhteystietoja ei näytetä, mutta omistajalle voi lähettää viestin ohjelman kautta. Siinäkin tapauksessa, että polkupyörä ei löydy lainkaan tietokannasta, löytymisilmoitus voidaan tehdä. Tässä tapauksessa, jos omistaja lisää pyöränsä myöhemmin tietokantaan, on hänen mahdollista lukea löytymisilmoitukset jälkikäteen.

Ohjelma on tarkoitettu julkaistavaksi vain Suomessa. Rajaamalla sovelluksen julkaisumaa Suomeen viestitetään asiakkaalle, että sovelluksella löytää suomalaisia polkupyöriä Suomesta. Tällä luodaan mielikuva siitä, että kyseessä on kotimaassamme toimiva tuote, jolloin sillä on suurempi mahdollisuus saada aikaan onnistunut hakutulos kotimaassamme. Sovelluksen ensimmäinen versio ei tue ulkomaille vietyjen polkupyörien etsimistä, eikä löytymisilmoituksen tekemistä ulkomailla.

Tarkoituksena on, että sovellukseen on mahdollista lisätä yksi polkupyörä ilman veloitusta. Toisen polkupyörän lisäämisestä peritään käyttäjältä maksu. Tällöin ohjelma on kaikkien potentiaalisten käyttäjien saatavilla ilmaiseksi. Maksavana asiakasryhmänä nähdään polkupyöräharrastajat, joilla polkupyöriä on todennäköisemmin useampia. Ilmaisen ensipyörän lisäysmahdollisuuden on tarkoitus lisätä ohjelman leviämistä työ- ja koulumatkapyöräilijöiden keskuudessa. Tämän toivotaan auttavan sovellusta leviämään nopeammin laajemman käyttäjäkunnan tietoisuuteen, jolloin se tavoittaa myös maksavat asiakkaat.

2.1 Työkalut ja teknologiat

Tässä opinnäytetyössä käytetään Cloud Firestore -tietokantaa. Se on osa Googlen Firebase-työkalujen kokoelmaa, johon kuuluu tietokantapalvelun lisäksi mm. todennus-, koneoppimis- ja ohjelmistotestauspalvelut, pilvitallennustila, sekä käyttäjätietojen analysointityökalut. Google on kerännyt useita työkaluja samaan palveluun helpottaakseen niiden hallintaa. Kaikkia näitä voidaan hallita saman web-konsolin kautta.

Cloud Firestore on tietokantapalvelin edellä mainitusta kokoelmasta. Sen perustana on NoSQL. NoSQL tulee englanninkielen sanoista *Not only SQL.* Se on käsite, jolla kuvataan perinteisestä relaatiomallista poikkeavaa tietokantaa. (Edlich 2020.)

Cloud Firestore on valittu tässä opinnäytetyössä käytettäväksi tietokannaksi, koska sille on sisäänrakennettu tuki Android-kehitysympäristössä ja Google on dokumentoinut sen käyttöönoton yksityiskohtaisesti. Tästä on etua, koska tarkoituksena on käyttää Android-kehitysympäristöä mobiilisovelluksen kehittämiseen. Cloud Firestore on suhteellisen uusi tietokanta. Se on julkaistu vuoden 2019 tammikuussa (Condon 2019).

Kehitysympäristöksi valittiin Android Studio, koska Android-mobiilialusta on tällä hetkellä maailman käytetyin mobiilialusta (StatCounter 2020). Valitsemalla Android-mobiilialusta saavutetaan kerralla suurin mahdollinen yleisö.

Ohjelmointikielenä projektissa käytetään Java-kieltä, koska Android-kehitysympäristössä Java on ollut kauemmin käytössä kuin Kotlin. Täten mahdollisissa ongelmatilanteissa apua verkosta löytyy helpommin Java-kieltä käytettäessä.

Sovelluksen versionhallintaan käytetään Git-versionhallintapalvelinta. Android Studio tukee Git-versionhallintaa, joten se on luonteva valinta projektiin. Eri versionhallintapalveluiden välillä vertailua ei tehty, koska projektin onnistumisen kannalta sen merkitys on vähäinen.

2.2 Vastaavanlaiset tai verrattavissa olevat tuotteet

Samankaltaisia sovelluksia etsiessäni en löytänyt yhtään samankaltaista mobiilisovellusta. Sen sijaan vastaavalla sovellusidealla löysin muutamia verkkosivustoja. Näistä osa oli kansallisia ja osa kansainvälisiä.

Ensimmäisenä vertailukohtana on BikeRegister, joka on suunniteltu Iso-Britannian kansalaisille (BikeRegister 2004). Sivustolle on mahdollista myös

rekisteröidä polkupyörä mistä tahansa maasta, mutta sivulla vieraillessaan saa käsityksen, että se on tarkoitettu pääasiassa Iso-Britannian kansalaisille. Sivustolla on mahdollista lisätä oma polkupyörä tietokantaan ja etsiä polkupyöriä runkonumerolla. Lisäksi sivustolta on mahdollista ostaa tarrasarja, jonka liimaamalla polkupyörään, sen tunnistus helpottuu. Sivusto tekee yhteistyötä Iso-Britannian poliisivoimien kanssa.

Seuraavana vertailukohtana on Bike Index, joka on yhdysvaltalainen sivusto (Bike Index 2013). Sivusto yrittää vakuuttaa vierailijat siitä, että se on kansainvälinen sivusto, mutta valittavina kielinä ovat ainoastaan englanti ja hollanti. Sivustolla on mahdollista rekisteröidä polkupyörien lisäksi myös pyörän lukkoja. Avaimen hävitessä sivustolta voi tarkistaa avaimen lovitusnumeron ja tarvittaessa tilata uuden avaimen. Sivustolla on myös mahdollista lisätä tarkempaa tietoa polkupyörästään, kuten yksittäisten osien merkit ja mallit, rungon väri, renkaiden koko tai QR-kooditarran.

Eurooppalaisella Bike-ID-sivustolla pyörän rekisteröintimahdollisuuden ja runkonumerohaun lisäksi ei ole muita ominaisuuksia (Bike-ID 2015). Pyörästä ei myöskään voi lisätä perustietojen lisäksi muita tietoja, eikä erillisiä QR-tarrasarjoja voi tilata.

Suomalainen Polkupyörärekisteri on sivustoista ainoa, joka on ensisijaisesti tarkoitettu suomalaisille (Polkupyörärekisteri 2017). Polkupyörän rekisteröinti sivulle maksaa 7,90 €, ja se on voimassa 5 vuotta. Tähän hintaan kuuluu tarrasarja, joka kiinnitetään polkupyörään. Sivustolla mainitaan: ”Polkupyörärekisterissä olevan polkupyörän runkonumerolla voidaan aina selvittää, vaikka huollon yhteydessä onko polkupyörä varastettu.”, mutta siitä ei ole varmuutta, onko tämä käytäntö polkupyöräkorjaamoissa yleinen ja vakiintunut toimenpide, vaikka se olisikin mahdollista.

Samankaltaisten sovellusten vertailu osoitti, että mobiilialustalle ei löytynyt vastaavaa tuotetta. Verkkosivustoja oli muutama, joista vain yksi oli suomalainen. Tällä sivustolla ei kuitenkaan ollut mahdollisuutta ilmaiseen kokeiluun ja

hinnoittelultaan se oli vertailun kallein. Vertailun perusteella voidaan olettaa, että paremmin hinnoitellulla mobiililaitteissa toimivalla sovelluksella on mahdollisesti kysyntää Suomessa.

3 SUUNNITTELU

Sovellustuotetta kehitettäessä on tunnettava kohderyhmä, jolle tuote tehdään. Tämä tarkoittaa, että tiedetään, mikä on käyttäjän tavoite, mitä käyttäjät varsinaisesti ovat tekemässä sovellusta käyttäessään, missä sovellusta käytetään ja minkälaisia vaatimuksia nämä asettavat sovelluksen käytettävyydelle. Tyypilliset sovelluksen käytettävyysvaatimukset ovat: opittavuus, käyttötehokkuus ja käytön miellyttävyys. Itsestään selvää on, että sovelluksen tulee sopia tehtävään, jota sen avulla suoritetaan. (Sinkkonen, Kuoppala, Parkkinen & Vastamäki 2006, 15.)

Tässä tapauksessa käyttäjäryhmä koostuu käyttäjistä, joilla on huoli polkupyörän katoamisesta. Käyttäjän tavoitteena on estää polkupyörän katoaminen tai löytää pyörä, jos vahinko on jo tapahtunut. Käyttäjä voi käyttää tuotetta taltioimaan polkupyöränsä tiedot, jotta ne ovat tarvittaessa helposti löydettävissä. Polkupyörän tiedot, kuten runkonumero, on hyvä olla helposti saatavilla, jos pyörän katoamisesta joutuu ilmoittamaan viranomaisille. Lisäksi polkupyörän kadotessa sen voi ilmoittaa kadonneeksi, jolloin se on mahdollista löytää sovelluksen avulla. Sovelluksen opittavuuden tulisi olla mahdollisimman nopea ja yksinkertainen, jotta se on helposti lähestyttävä. Sovellus on ns. ”käytä kerran ja unohda” -tyyppinen sovellus, joten sen oppimiseen ei tulisi kulua liikaa aikaa. Tätä tavoitetta edistetään pitämällä sovelluksen ensimmäinen näkymä mahdollisimman yksinkertaisena, jotta käyttäjälle ei tule ns. ”infoähkyä” heti sen käynnistyttyä. Tärkeimpiä ominaisuuksia korostetaan ja edistyneemmät ominaisuudet piilotetaan pois näkyviltä.

OmaFillari-sovellus toteutetaan huomioiden seuraavat suunnitteluperiaatteet:

- tiedon etsimistapa
- tiedon määrä: liika on liikaa ja liian vähän ei ole riittävästi
- järjestys: looginen ja peräkkäinen, selkeä aloituskohta
- hierarkiat
- rytmitys, käyttäjän katseen ohjaaminen ja ikkunan sisäinen navigointi
- estetiikka: tyhjä tila, ikkunan tasapaino, ryhmittely
- mikä ikkunassa on tärkeintä?
- asioiden hahmottuminen
- asioiden näkyvyys, tarvitaanko erityisiä visuaalisia vihjeitä?

(Sinkkonen ym. 2006, 110.)

Kuva 1. Käyttötapauskaavio

Yllä olevaa käyttötapauskaaviota (Kuva 1) tarkasteltaessa huomataan, että sovelluksen käyttäjät (asiakas) toimivat kahdessa eri roolissa. He toimivat pääasiassa polkupyörän omistajan roolissa, mutta löytäessään kadotetun kulkuneuvon, he toimivatkin polkupyörän löytäjän roolissa. On kuitenkin hyvin mahdollista, että osa käyttäjistä toimii ainoastaan polkupyörän löytäjän roolissa. Esimerkiksi ostoskeskuksen järjestyksenvalvoja voi ladata sovelluksen puhelimelleen selvittääkseen pitkään pyörätelineessä seisseen polkupyörän omistajan tai lähettääkseen tälle sovelluksen kautta viestin.

3.1 Käyttöliittymän rakenne ja toiminta

Android Studio suunnittelussa käyttöliittymä koostuu ”Activity”-objekteista, joista tekstissä käytetään jatkossa nimitystä ”näkymä”. Käyttöliittymä koostuu siis useasta näkymästä, joiden hahmotelmat esitellään seuraavaksi.

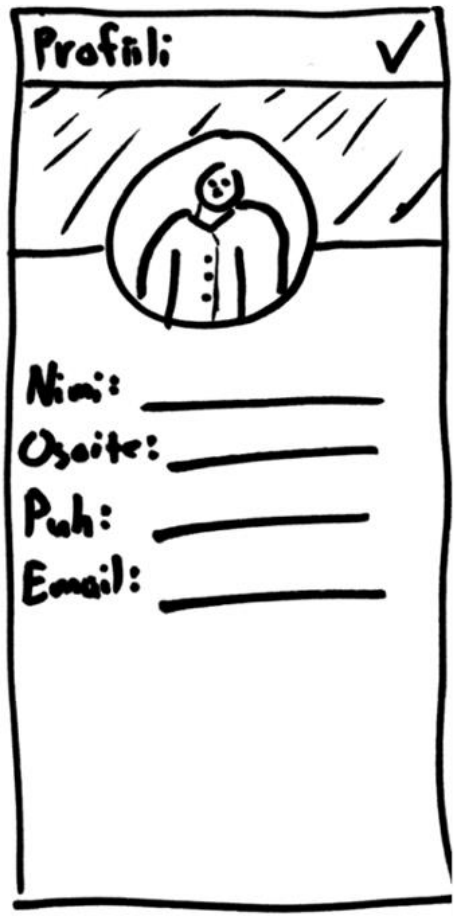
Kuva 2. Hahmotelma aloitusnäkymästä

Aloitusnäkymä pidetään yksinkertaisena (Kuva 2). Yksinkertainen aloitusnäkymä vähentää käyttäjän kokemaa stressiä uuden käyttöliittymän oppimistilanteessa,

kun valittavien painikkeiden määrä pidetään minimissä. Uutta tuotetta opetellessaan käyttäjä pohtii tietoisesti valintojen välillä. Näyttämällä käyttäjälle aloitusnäkymässä ainoastaan valikkopainike ja polkupyörän lisäyspainike valinnasta painikkeiden välillä saadaan yksinkertainen. (Sinkkonen ym. 2006, 217 – 218.)

Jos polkupyöriä ei ole vielä lisätty, näkymän keskellä on plusmerkki, jota painamalla aloitetaan ensimmäisen polkupyörän lisääminen. Jos polkupyörä on jo lisätty, näkyy sen tiedot ”laatikon” sisällä, joka sisältää vasemmalla puolella kuvan polkupyörästä, sekä oikealla tekstinä merkin, mallin ja runkonumeron. ”Laatikkoa” painamalla päästään polkupyöränäkymään. Oikeassa yläkulmassa on valikkopainike, jonka alta löytyy kaikki muut ominaisuudet. Valikko sisältää seuraavat elementit:

- Löysin hylätyn polkupyörän
- Etsi polkupyörää runkonumerolla
- Lisää polkupyörä
- Käyttäjäprofiili



Kuva 3. Hahmotelma profiilinäkymästä

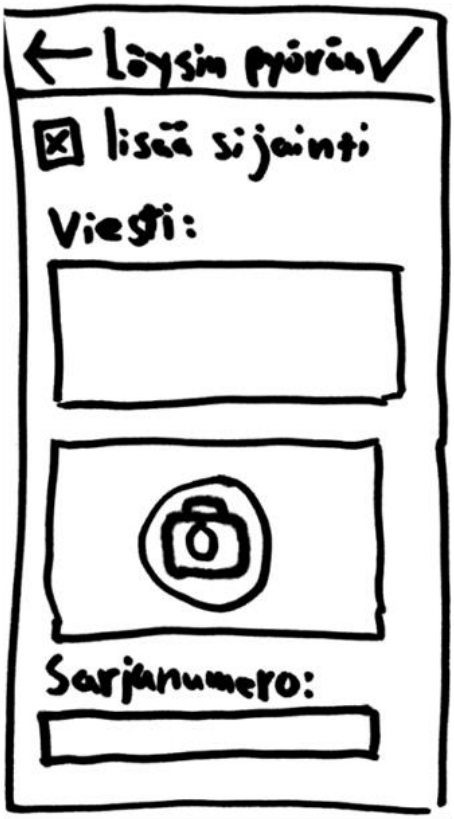
Profiilinäkymässä (Kuva 3) ensimmäisenä on käyttäjän valokuva. Se on pyöreän kehyksen sisällä. Seuraavaksi sen alla ovat tekstikentät nimelle, osoitteelle, puhelinnumerolle ja sähköpostiosoitteelle. Profiilinäkymän oikeassa yläkulmassa on ✓-symboli, joka tallentaa muutokset ja poistuu profiilinäkymästä.

Kuva 4. Hahmotelma etsintänäkymästä

Etsintänäkymä (Kuva 4) jaetaan kahteen osaan. Yläosa on hakuosio, mikä sisältää tekstinsyöttökentän, johon polkupyörän runkonumero voidaan kirjoittaa. Tekstikentän oikealla puolella on hakupainike. Hakutulokset ilmestyvät alaosioon. Hakutulokset näkyvät samanlaisina ”laatikoina” kuin aloitusnäytöllä. Hakutulosta painamalla päästään polkupyöränäkymään.

Kuva 5. Hahmotelma polkukpyöränäkymästä

Polkupyöränäkymässä (Kuva 5) ensimmäisenä on kuva polkupyörästä. Tämän alapuolella on kolme tekstikenttää. Jos polkupyörä on käyttäjän oma, tekstikenttiin voidaan syöttää polkupyörän merkki, malli ja runkonumero. Oikeassa yläkulmassa on hyväksymispainike, jota painamalla muuttuneet tiedot tallennetaan ja sivu suljetaan. Sen vasemmalla puolella on karttapainike, josta näkee muiden käyttäjien lähettämät viestit koskien kyseistä pyörää sijaintitietojen kera. Nämä tiedot näytetään sijaintinäytöllä. Jos taas polkupyörä ei ole käyttäjän oma, vaan saatu hakutuloksena etsiessä runkonumeron avulla, karttapainike piilotetaan ja tekstikenttiin ei voida syöttää uutta tietoa. Lisäksi tekstikenttien alle tulee näkyviin polkupyörän omistajan yhteystiedot, mikäli hän on ilmoittanut kyseisen polkupyörän kadonneeksi. Polkupyörän omistajalle ei näytetä hänen omia tietojaan, vaan sen sijaan näytetään liukukytkin, jolla polkupyörän voi ilmoittaa kadonneeksi. Polkupyörän poistopainike sijoitetaan karttapainikkeen vasemmalle puolelle.



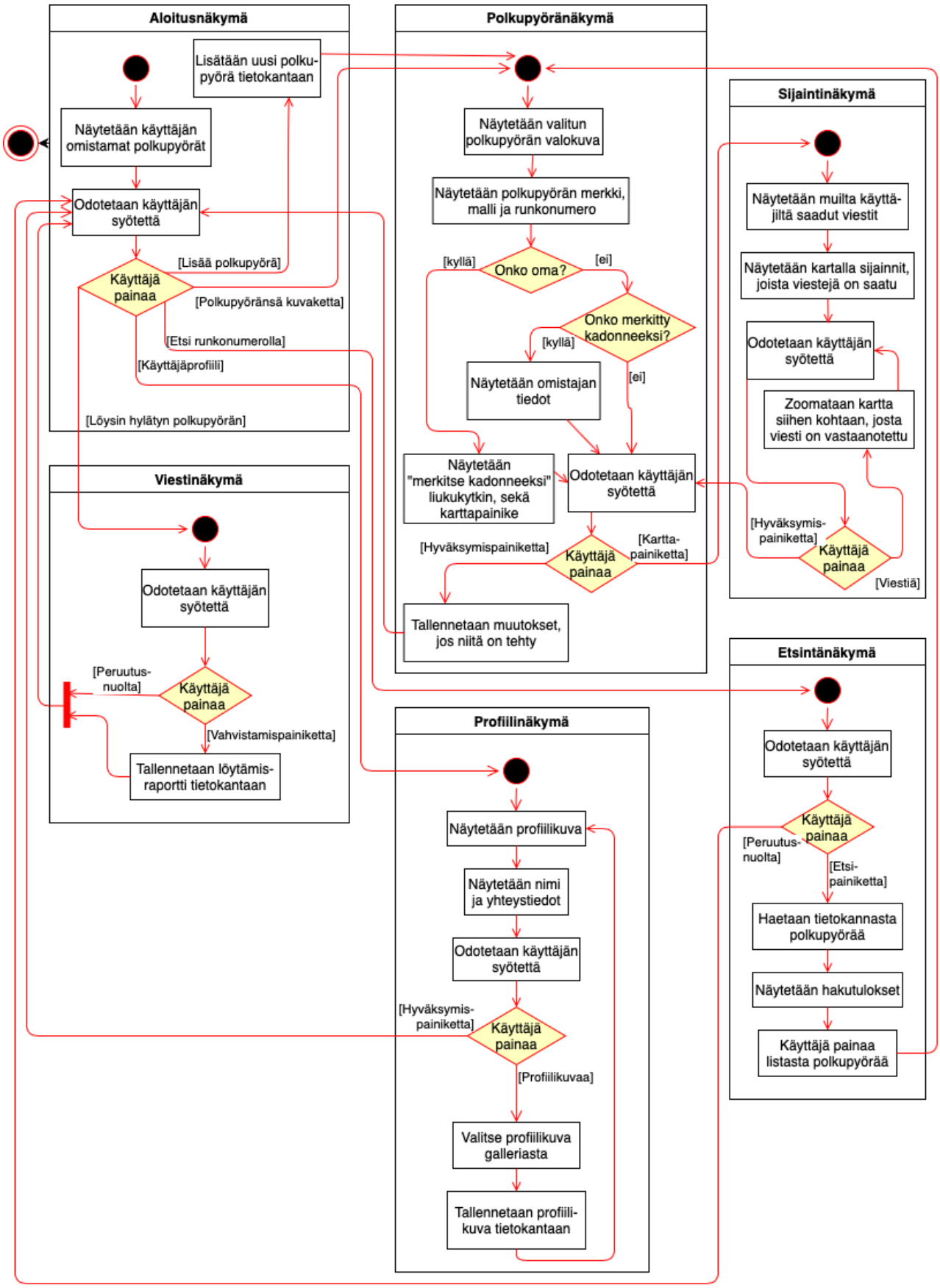
Kuva 6. Hahmotelma viestinäkymästä

Viestinäkymä (Kuva 6) ylimpänä on mahdollisuus lähettää nykyinen sijainti viestin vastaanottajalle laittamalla rasti kohtaan ”lisää sijainti”. Sen alapuolella on viestikenttä, johon voi kirjoittaa vastaanottajalle viestin. Halutessaan mukaan voi liittää valokuvan. Seuraavaksi syötetään löydetyn polkupyörän runkonumero. Lopuksi oikeasta yläkulmasta valitaan hyväksymispainike, jolloin viesti lähetetään.

Kuva 7. Hahmotelma sijaintinäkymästä

Sijaintinäkymässä (Kuva 7) nähdään vastaanotetut viestit muilta käyttäjiltä. Ylimpänä näytöllä on kartta, johon on merkitty kaikki sijainnit, joista viestejä on vastaanotettu. Tämän alla on lista vastaanotetuista viesteistä. Viestikenttä sisältää ajan, jolloin viesti on lähetetty ja toisen käyttäjän lähettämän viestin. Painamalla viestiä kartta lähentää sijaintiin, josta se on lähetetty.

Käyttöliittymähahmotelmien pohjalta on koottu aktiviteettikaavio (Kuva 8), joka selventää käyttöliittymän toimintaa ja eri näkymien välisiä yhteyksiä.



Kuva 8. Aktiviteettikaavio

Aktiviteetti kaaviosta nähdään, millaisissa tapauksissa siirrytään näkymästä toiseen ja milloin palataan alkunäkymään. Kaaviossa on vain yksi loppupiste, koska periaatteessa jokainen näkymä on Android-ympäristössä oma instanssinsa, jolla ei ole loppua. Ohjelmaa ei siis käytännössä koskaan lopeteta, vaan se jää käyntiin taustalle pysäytystilassa. Käyttöjärjestelmä sulkee sovelluksen lopulta muistin loppuessa kesken. Perinteistä aktiviteetin loppua oli hieman hankala sijoittaa kaavioon, mutta se on kaaviossa sijoitettu aloitusnäkymä luokan osaksi. Tämän on tarkoitus ilmaista sitä, että sovellus sulkeutuu joskus, jos käyttöjärjestelmä niin päättää.

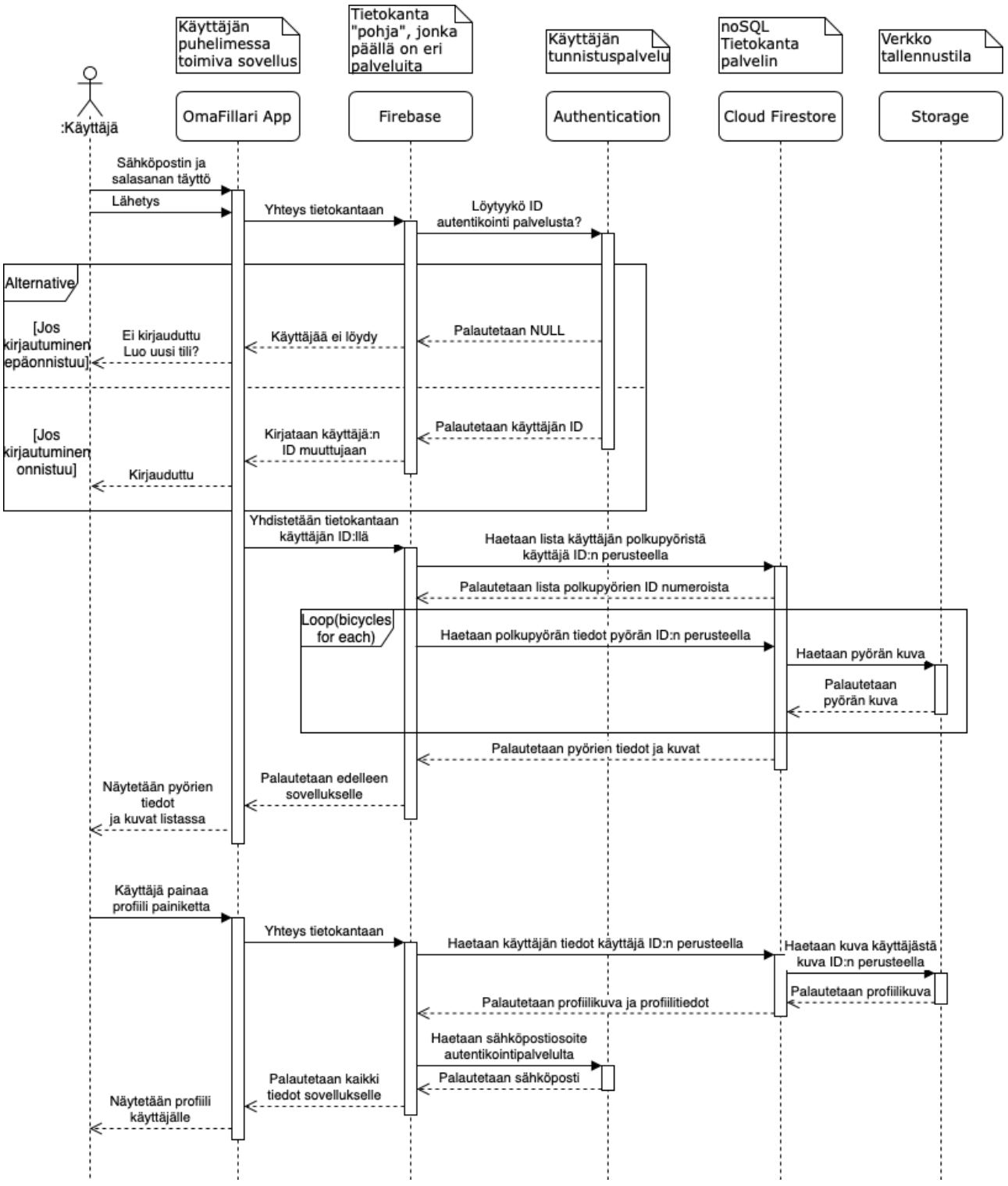
3.2 Taustapalvelujen toiminta

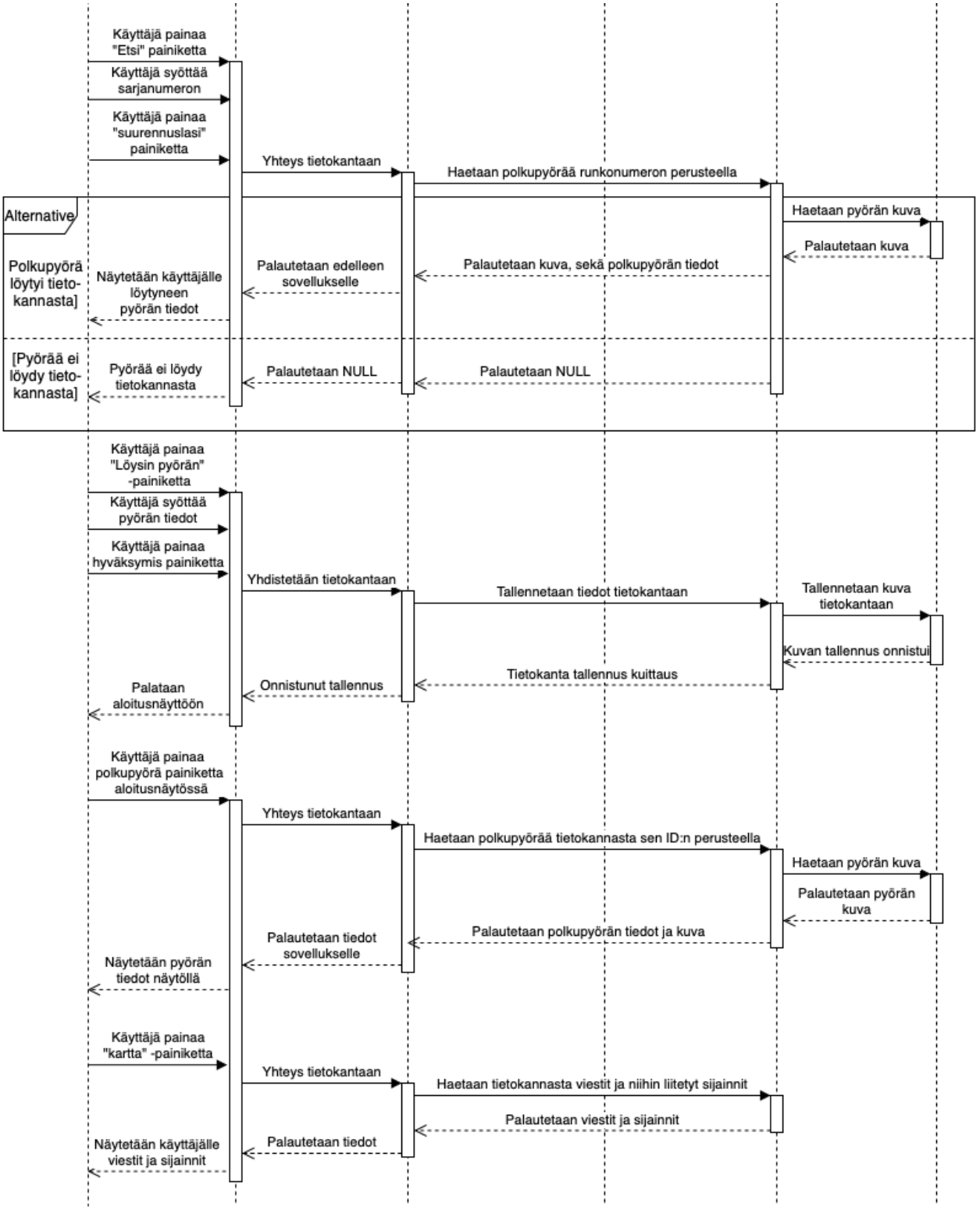
Cloud Firestore on tietokanta, jossa kaikki tekstimuodossa esitettävä tieto polkupyöristä sijaitsee. OmaFillari-mobiilisovellus toimii välikätenä ja graafisena käyttöliittymänä Cloud Firestore -tietokannan ja käyttäjän välillä.

Firebase Storage on pilvitallennuspalvelin, johon on tarkoitus tallentaa kaikki käyttäjien lisäämät polkupyöräkuvat. Ilmaista tilaa on käytettävissä 5 Gt, joten kuvien tarkkuutta ja tiedostokokoa joudutaan optimoimaan tilansäästön kannalta.

Firebase Authentication on tunnistautumispalvelin, joka todentaa käyttäjän hänen annettuaan käyttäjätunnuksensa ja salasanansa. Tunnistautumista tarvitaan sovelluksen käynnistyksen yhteydessä.

Alla olevasta sekvenssikaaviosta (Kuva 9) nähdään, milloin käyttäjän ja tietokannan välinen yhteys muodostuu ja millainen käyttäjän, sovelluksen ja palvelimen välinen tiedonkulku on yleisimmissä käyttötapauksissa. Sekvenssikaaviossa ei ole tarkoitus ottaa huomioon kaikkia mahdollisia häiriötilanteita. Kaavion tarkoituksena on selventää työnkulku niissä tilanteissa, joissa sovellus toimii tarkoitetulla tavalla.


Käyttäjän puhelimessa toimiva sovellus
Tietokanta "pohja", jonka päällä on eri palveluita
Käyttäjän tunnistuspalvelu
noSQL Tietokanta palvelin
Verkko tallennustila
:Käyttäjä
OmaFillari App
Firebase
Authentication
Cloud Firestore
Storage
Sähköpostin ja salasanan täyttö
Lähetys
Yhteys tietokantaan
Löytyykö ID autentikointi palvelusta?
Alternative
[Jos kirjautuminen epäonnistuu]
Ei kirjauduttu Luo uusi tili?
Käyttäjää ei löydy
Palautetaan NULL
[Jos kirjautuminen onnistuu]
Kirjataan käyttäjä:n ID muuttujaan
Palautetaan käyttäjän ID
Kirjauduttu
Yhdistetään tietokantaan käyttäjän ID:llä
Haetaan lista käyttäjän polkupyöristä käyttäjä ID:n perusteella
Palautetaan lista polkupyörien ID numeroista
Loop(bicycles for each)
Haetaan polkupyörän tiedot pyörän ID:n perusteella
Haetaan pyörän kuva
Palautetaan pyörän kuva
Palautetaan pyörien tiedot ja kuvat
Näytetään pyörien tiedot ja kuvat listassa
Palautetaan edelleen sovellukselle
Käyttäjä painaa profiili painiketta
Yhteys tietokantaan
Haetaan käyttäjän tiedot käyttäjä ID:n perusteella
Haetaan kuva käyttäjästä kuva ID:n perusteella
Palautetaan profiilikuva
Palautetaan profiilikuva ja profiilitiedot
Haetaan sähköpostiosoite autentikointipalvelulta
Palautetaan sähköposti
Palautetaan kaikki tiedot sovellukselle
Näytetään profiili käyttäjälle



Kuva 9. Sekvenssikaavio

Sekvenssikaavio osoittaa, että kaikki tietokantatoiminta tapahtuu Firebase-alustan kautta, jonka jälkeen yhteys muodostetaan tiettyyn alipalveluun. Käyttäjän tunnistus tapahtuu vain avattaessa sovellus ja tarkistettaessa käyttäjän sähköpostiosoite profiilinäkymässä. Myös Storage-palvelua käytetään silloin, kun

Cloud Firestore -tietokannassa on merkintä siitä, että valokuva on tallennettu palvelimelle. Lähes jokaiseen toimintoon liittyy yhteydenotto Cloud Firestore -tietokantaan.

3.3 Käyttöliittymän tyyli

OmaFillarin käyttöliittymän väreillä on tarkoitus viestiä käyttäjälle luotettavasta ympäristöstä ja vaikuttaa käyttäjän tunnetilaan. Väreillä halutaan lisäksi viestiä rauhallisuutta, rehellisyyttä, suomalaisuutta ja selkeyttä. Värimaailma koostuu sen vuoksi lähinnä väreistä sininen ja valkoinen.

Sininen väri viestii käyttäjälle mm. seuraavia tunteita: tosi, rauhallisuus ja viattomuus. Tummansininen on auktoriteettien väri ja se ilmaisee myös luotettavuutta, voimaa ja suoritusta. Yhdistettynä valkoiseen väriin saadaan: Suomi ja isänmaallisuus. (Sinkkonen ym. 2006,130.)

Suunnittelussa päädyin seuraaviin väreihin (Kuva 10):

Vaalea pääväri	#3995B3	
Tumma pääväri	#276478	
Tehoste väri	#A9EAFF	
Otsikko tekstien väri	#276478	
Normaali tekstin väri	#292929	
Painikkeiden taustaväri	#276478	
Painikkeiden teksti	#FFFFFF	

Kuva 10. Sovelluksen väripaletti

Apuna värisuunnitteluun on saatu Color Hunt -verkkosivuilla olevista valmiista väripaleteista, joiden pohjalta väripaletti on tehty (Color Hunt 2015). Valmiiseen väripalettiin on tehty pieniä muutoksia sovelluksen luettavuuden parantamiseksi.

4 TOTEUTUS

Toteutus aloitettiin suunnittelun jälkeen 14.2.2020 asentamalla ensimmäiseksi ohjelmointiympäristö tietokoneelle. Toteutusvaiheessa tavoitteena oli saada

sovelluksesta valmiiksi ensimmäinen toimiva versio, jossa on mahdollisuus suorittaa kaikki suunnitteluvaiheessa kuvaillut toiminnot. Tavoitteena oli myös samalla testata sovellusta jatkuvasti niin, että vältyttäisiin pahimmilta ohjelmointivirheiltä kehitysvaiheessa. Toteutusvaiheen periaatteena oli että, tarvittaessa työnkulku voi poiketa suunnitelmista, jos suunnittelussa havaitaan selkeä virhe ja poikkeaminen suunnitelmasta nähdään tarpeelliseksi. Tarkoitus oli kuitenkin välttää uusien ideoiden käyttöönottoa ja sen sijaan pysyä suunnitteludokumentoinnin rajoissa. Uudet ideat kuitenkin kirjattiin ylös myöhempää kehityssuunnittelua varten.

Kun ensimmäinen ohjelmistoversio valmistuu, voidaan käydä läpi ideat, joita sovelluskehityksen aikana on keksitty ja pohtia, voisiko niistä tulla osa kokonaisuutta. Lisäksi käydään läpi suunnitelma ja tutkitaan, mitkä ominaisuudet eivät onnistuneet suunnitelmien mukaan ja mitkä syyt johtivat päätökseen muuttaa niitä.

4.1 Avainominaisuuksien toteutustapa

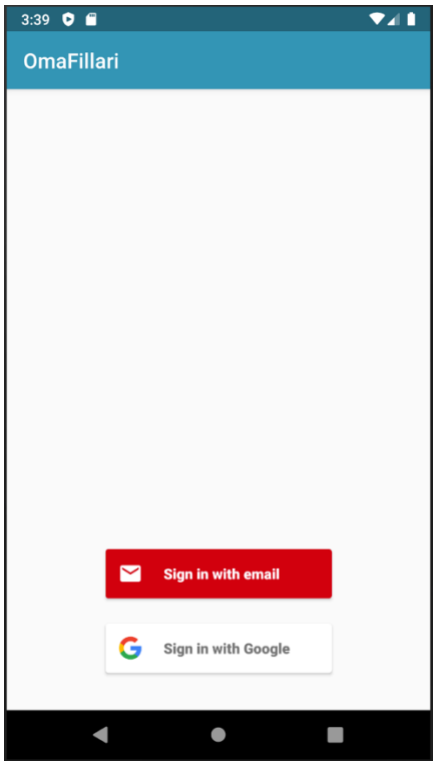
Tässä luvussa käsitellään avainominaisuuksien toteutustapoja kokonaisuus kerrallaan. Jokaisen kokonaisuuden muodostaa näkymä ja sen taustalla oleva ohjelmakoodi.

Toteutus aloitettiin kirjautumalla Firebase-konsoliin, joka on verkkosivun muodossa. Tämän jälkeen luotiin Firebase-projekti ja valittiin tietokannaksi Cloud Firebase -tietokanta. Verkkosivusto palauttaa tässä vaiheessa käyttäjälle yhteyden muodostamiseen tarvittavat kirjautumistunnukset. Nämä Firebase-kirjautumistunnukset syötettiin ohjatulla asennuksella Android Studio -ohjelmointiympäristöön. Ohjelmointiympäristön asennus ja sen liittäminen tietokantaan on Android-alustalla hyvin yksinkertainen toimenpide ohjatun asennuksen ansiosta.

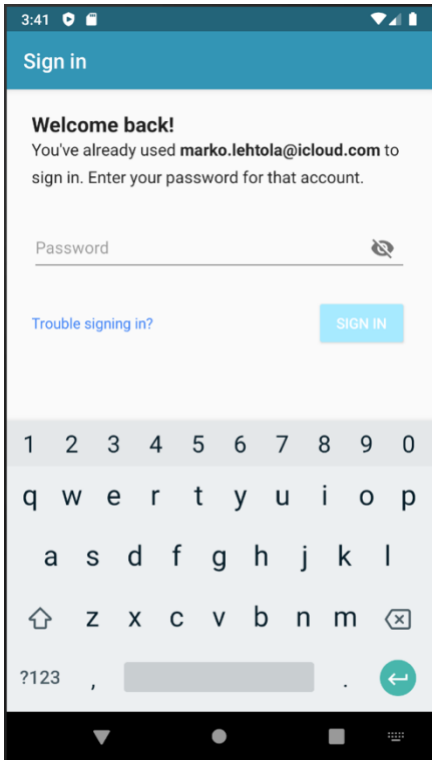
4.1.1 Kirjautumisnäkymä

Käynnistettäessä sovelluksen ensimmäinen näkymä on kirjautumisnäkymä. Jotta käyttäjä voi tallentaa tietoja tietokantaan ja tiedot pystytään yhdistämään käyttäjään, tarvitsee käyttäjällä olla tunnistenumero. Tämän vuoksi kirjautumisnäkymä toteutettiin ensimmäisenä. Kirjautuminen toteutettiin aluksi itse ohjelmoidulla sähköpostiosoite ja salasana yhdistelmällä ja sitä käytettiin sovelluskehityksen alkuvaiheessa ainoana kirjautumistapana.

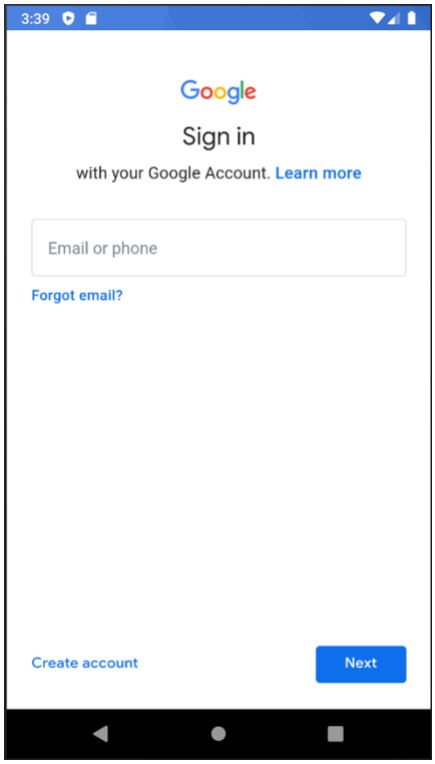
Myöhemmin sovellukseen lisättiin mahdollisuus kirjautua Google-tilillä. Tässä vaiheessa havaittiin, että Google tarjoaa valmiin kirjautumislomakkeen, jolla voitiin toteuttaa kirjautuminen sekä Google-tilillä että sähköpostiosoitteella. Alkuperäinen itseohjelmoitu kirjautumisnäkymä päätettiin hylätä ja sen tilalle otettiin valmis kirjautumislomake. Kirjautumisnäkymästä (Kuva 11) valitaan kirjautumistapa ja seuraava näkymä määräytyy valinnan mukaan (Kuva 12 ja Kuva 13).



Kuva 11. Kirjautumisnäkymä



Kuva 12. Sähköposti kirjautuminen

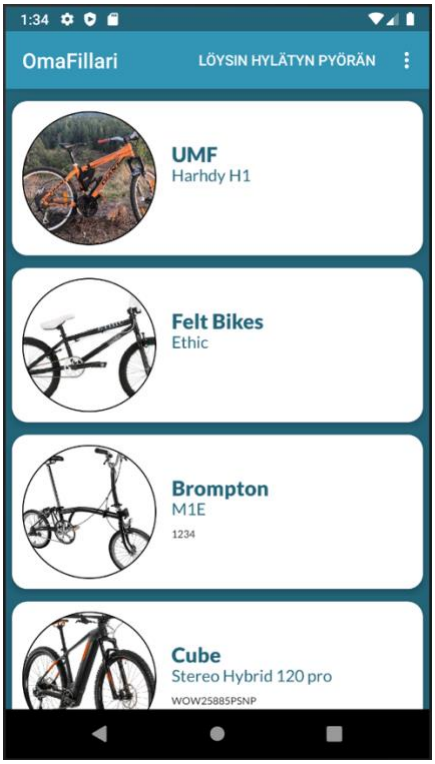


Kuva 13. Google kirjautuminen

Kirjautumislomakkeen kieli on sama kuin puhelimen käyttöjärjestelmän kieli. Kirjautumisen jälkeen siirrytään aloitusnäkymään.

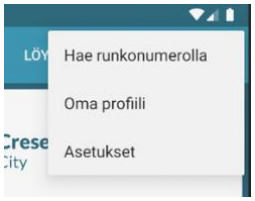
4.1.2 Aloitusnäkymä

Kirjautumisen onnistuttua avautuu näkyville aloitusnäkymä (Kuva 14). Aloitusnäkymään haetaan tietokannasta käyttäjän polkupyörät ja niihin liitetyt kuvat. Polkupyörät esitetään listana, jota voi vierittää ylös ja alas. Oikeassa yläkulmassa on valikkopainike. Sen vasemmalla puollella on pikapainike, jolla voi lähettää viestin pyörän omistajalle runkonumeron perusteella.



Kuva 14. Aloitusnäkymä

Ensimmäisenä koodiin lisättiin tarkistus, mikä varmistaa, että sovellus on kirjautunut tietokantaan. Jos sovellus ei ole kirjautunut, kirjaudutaan uudestaan tietokantaan. Tämä tarkastus tehdään siksi, että sovelluksen käyttäjä on voinut jättää sovelluksen puhelimen tausta-ajoon pitkäksi ajaksi ja palatessaan takaisin sovellukseen, ovat kirjautumistiedot ehtineet vanhentua. Tällöin kirjautuminen suoritetaan uudestaan. Tämän jälkeen lisättiin asetusvalikko. Valikkoon lisättiin valinnat: ”Hae runkonumerolla”, ”Oma profiili” ja ”Asetukset”. Asetusvalikon (Kuva 15) alavalikko ”Asetukset” sisältää ainoastaan mahdollisuuden kirjautua ulos sovelluksesta.



Kuva 15. Asetus valikko

Ensin polkupyörien näyttäminen polkupyörälistassa oli toteutettu niin, että tiedot jokaisesta polkupyörästä haettiin tietokannasta listoihin. Muuttuja-listat merkeistä, malleista, kuvien osoitteista ja sarjanumeroista lähetettiin ListView-luokan elementille erillisinä listoina.

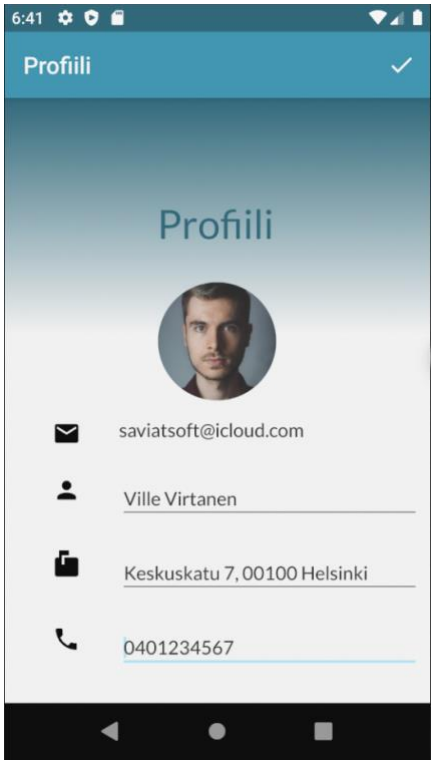
Tämä toteutus muutettiin myöhemmin niin, että luotiin Bicycle-luokka, joka poisti tarpeen käsitellä samaan aikaan useaa erillistä listaa tiedoista. Tietokannasta haetaan suoraan kaikki polkupyörää koskevat tiedot Bicycle-luokan olioon (Kuva 16, rivi 194) ja näistä olioista tehdään lista (Kuva 16, rivi 199), joka lähetetään ListView-luokan elementille (Kuva 16, rivi 206). Eli neljän erillisen listan sijaan lähetetään vain yksi lista. Näin ohjelman toiminta selkeytyy. Jatkossa jos polkupyörä olioon halutaan lisätä muuttujia, esimerkiksi polkupyörän vuosimalli, ei polkupyöriä tietokannasta noutavaan funktioon tarvitse tehdä muutoksia. Uuden muuttujan lisääminen luokkaan riittää.

```
180  @    public void getBicyclesByUser(final FirebaseUser _user){
181
182           bicycles.clear();
183
184           db.collection( collectionPath: "bicycles") CollectionReference
185                   .whereEqualTo( field: "owner", _user.getUid()) Query
186                   .get() Task<QuerySnapshot>
187  O                .addOnCompleteListener((task) → {
190                           if (task.isSuccessful()) {
191                               for (QueryDocumentSnapshot document : task.getResult()) {
192                                   Log.d(TAG,  msg: document.getId() + " => " + document.getData());
193                                   //haetaan pyörään tiedot tietokannasta
194                                   Bicycle bicycle = document.toObject(Bicycle.class);
195                                   //lisätään documentin id numero polkupyörän id:ksi,
196                                   //koska sitä ei välttämättä ole dokumentissa tai se voi olla väärä
197                                   bicycle.setId(document.getId());
198                                   //lisätään pyörä listaan
199                                   bicycles.add(bicycle);
200                               }
201                           } else {
202                               Log.i(TAG,  msg: "Error getting documents: ", task.getException());
203                               //poikkeus, yhtään polkupyörää ei löytynyt..
204                               Toast.makeText( context: MainActivity.this,  text: "Yhtään pyörää ei löytynyt ...", Toast.LENGTH_LONG).show();
205                           }
206                           showBicycles(bicycles);
207                   });
209       }
```

Kuva 16. Polkupyörän haku tietokannasta.

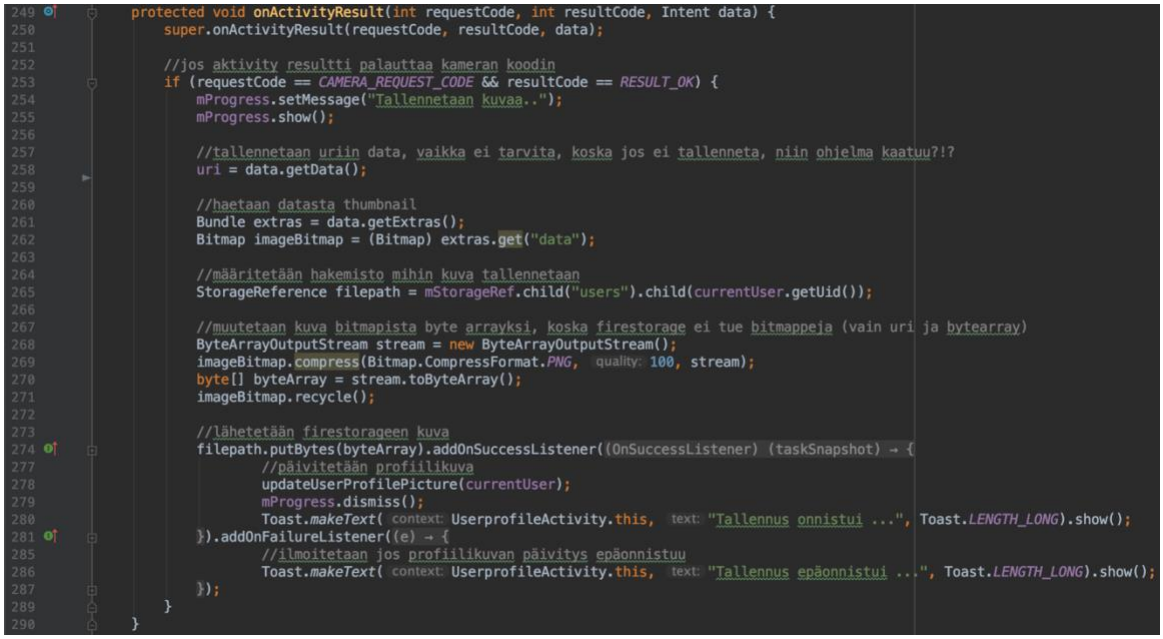
4.1.3 Profiilinäkymä

Käyttäjällä on mahdollisuus lisätä profiilisivulle profiilikuva ja yhteystiedot polkupyörän katoamisen varalta. Tietojen lisäyksen jälkeen muutokset tallennetaan painamalla oikeassa yläkulmassa (Kuva 17) olevaa hyväksymispainiketta.



Kuva 17. Profiilinäkymä

Profiilinäkymää tehdessä kohdattiin ongelma, joka kaatoi sovelluksen. Ylälaidan navigointipalkkiin ei voinut tehdä painiketta, joka kutsuu ohjelmakoodissa olevaa funktiota. Tämä johtui siitä, että toisin kuin profiilinäkymä, sen navigointipalkki ei ole linkitetty suoraan taustalla toimivaan ohjelmakoodiin, jolloin painike ei tiedä mistä tiedostosta sen tulisi kutsua ohjelmakoodia. Ongelma kierrettiin tekemällä ohjelmakoodiin kuuntelija, joka vuorostaan kuuntelee navigointipalkin painalluksia. Näin saatiin tietoliikenne koodin ja navigointipalkin välillä käännettyä toisinpäin ja ongelma ratkaistua.



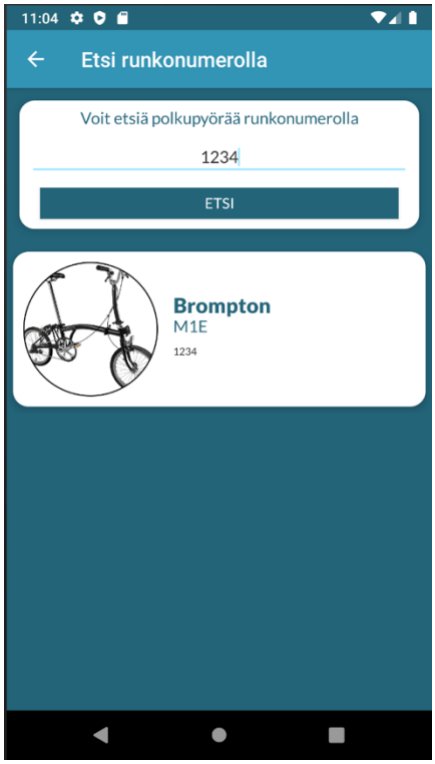
```
249    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
250        super.onActivityResult(requestCode, resultCode, data);
251
252        //jos aktivity resultti palauttaa kameran koodin
253        if (requestCode == CAMERA_REQUEST_CODE && resultCode == RESULT_OK) {
254            mProgress.setMessage("Tallennetaan kuvaa..");
255            mProgress.show();
256
257            //tallennetaan uriin data, vaikka ei tarvita, koska jos ei tallenneta, niin ohjelma kaatuu?!?
258            uri = data.getData();
259
260            //haetaan datasta thumbnail
261            Bundle extras = data.getExtras();
262            Bitmap imageBitmap = (Bitmap) extras.get("data");
263
264            //määritetään hakemisto mihin kuva tallennetaan
265            StorageReference filepath = mStorageRef.child("users").child(currentUser.getUid());
266
267            //muutetaan kuva bitmapista byte arrayksi, koska firestorage ei tue bitmappeja (vain uri ja bytearray)
268            ByteArrayOutputStream stream = new ByteArrayOutputStream();
269            imageBitmap.compress(Bitmap.CompressFormat.PNG,  quality: 100, stream);
270            byte[] byteArray = stream.toByteArray();
271            imageBitmap.recycle();
272
273            //lähetetään firestorageen kuva
274            filepath.putBytes(byteArray).addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
277                    //päivitetään profiilikuva
278                    updateUserProfilePicture(currentUser);
279                    mProgress.dismiss();
280                    Toast.makeText( context: UserprofileActivity.this,  text: "Tallennus onnistui ...", Toast.LENGTH_LONG).show();
281            }).addOnFailureListener((e) → {
285                    //ilmoitetaan jos profiilikuvan päivitys epäonnistuu
286                    Toast.makeText( context: UserprofileActivity.this,  text: "Tallennus epäonnistui ...", Toast.LENGTH_LONG).show();
287            });
289        }
290    }
```

Kuva 18. Valokuvan tallennus tietokantaan

Profiilikuvan tallennustilanteessa puhelimen kamera kuvaa paljon tallennustilaa kuluttavia valokuvia. Valmiissa valokuvausfunktiossa on ominaisuus, joka tekee jokaisesta valokuvasta myös pienikokoisen esikatselukuvan, jota voidaan käyttää esimerkiksi galleriasovelluksen esikatselukuvana. Esikatselukuvan tarkkuus riitti hyvin profiilikuvan tarkkuudeksi, joten sitä päätettiin käyttää profiilikuvana (Kuva 18, rivi 262). Esikatselukuvan käyttö profiilikuvana kuitenkin kaatoi sovelluksen. Ongelman syyksi löytyi se, että todellista valokuvadataa ei käytetty ohjelmakoodissa. Jotta sovellus ei kaadu kuvatessa, täytyy valokuvattu data tallentaa muuttujaan (Kuva 18, rivi 258) vaikka sitä ei käytetä hyödyksi missään vaiheessa.

4.1.4 Etsintänäkymä

Etsintänäkymässä (Kuva 19) on ensimmäisenä muokattava tekstikenttä runkonumeron syöttöä varten. Tämän alla on ”ETSI”-painike. Suunnitelmassa painike oli piirretty tekstikentän jälkeen, mutta se siirrettiin tekstikentän alle, jotta siitä voisi tehdä suuremman. Muutos tehtiin siksi, että suurempaa painiketta on helpompi painaa ja se parantaa käytettävyyttä.

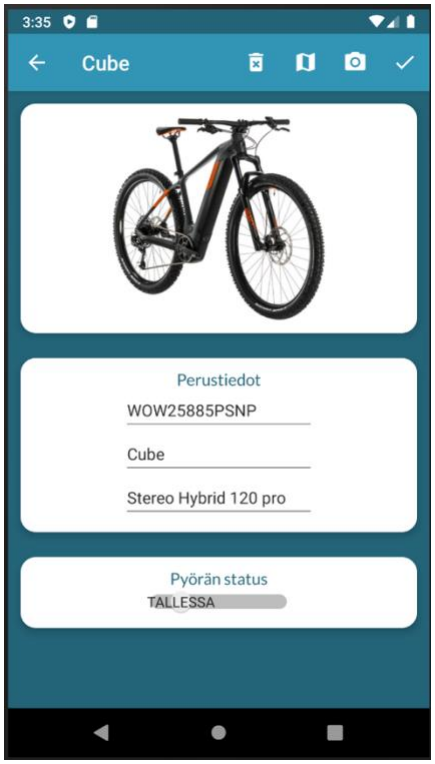


Kuva 19. Etsintänäkymä

Etsintänäkymässä käytetty ohjelmakoodi on identtinen aloitusnäkymässä käytetyn ohjelmakoodin kanssa, yhtä muutosta lukuun ottamatta: polkupyörälistaa ei haeta tietokannasta käyttäjätunnisteen perusteella, vaan haku tehdään runkonumeron perusteella. Hakutulosta painettaessa ohjelma siirtyy polkupyöränäkymään aivan kuten aloitusnäkymässäkin.

4.1.5 Polkupyöränäkymä

Suunnitteluvaiheen polkupyöränäkymän (Kuva 20) hahmotelmasta puuttui liukukytkin, jolla valitaan, onko polkupyörä kadonnut. Se lisättiin lopulliseen versioon, koska se on tärkeä elementti ohjelman käytettävyyden kannalta. Muilta osin edettiin suunnitteluvaiheessa tehdyn hahmotelman mukaan.



Kuva 20. Polkupyöränäkymä

Alla esitellään polkupyöränäkymän tärkein funktio, eli polkupyörän tietojen tallennusfunktio (Kuva 21).

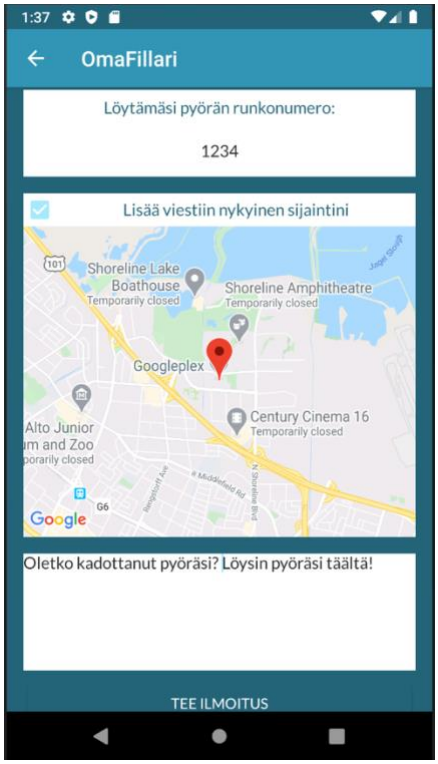
```
314     private void saveButtonPressed(){
315         //Luodaan bicycle olio
316         Bicycle bicycleToSave = new Bicycle();
317         bicycleToSave.setBrand(tvBrand.getText().toString());
318         bicycleToSave.setModel(tvModel.getText().toString());
319         bicycleToSave.setSerial(tvSerial.getText().toString());
320         bicycleToSave.setStolen(sStolen.isChecked());
321         bicycleToSave.setId(activeBicycle.getId());
322         bicycleToSave.setOwnerByString(activeBicycle.getOwner());
323         bicycleToSave.setPicture(activeBicycle.getPicture());
324
325         //Lähetetään ja tarvittaessa ylikirjoitetaan käyttäjän tiedot
326         db.collection( collectionPath: "bicycles").document(bicycleToSave.getId()).set(bicycleToSave, SetOptions.merge());
327
328         //suljetaan aktiviteetti
329         finish();
330     }
```

Kuva 21. Funktio polkupyörän lisäämiseksi

Aluksi luodaan polkupyöräobjekti (Kuva 21, rivi 316). Tämän jälkeen lisätään polkupyörän tiedot tekstikentistä objektin ominaisuuksiin (Kuva 21, rivit 317 – 323). Tässä vaiheessa objekti on valmis lähetettäväksi tietokantaan. Kun objekti on lähetetty tietokantaan (Kuva 21, rivi 326), polkupyöränäkymä suljetaan (Kuva 21, rivi 329).

4.1.6 Viestinäkymä

Viestinäkymässä (Kuva 22) ensimmäiseksi syötetään löydetyn polkupyörän runkonumero. Seuraavaksi valitaan, liitetäänkö mukaan sijainti, josta viesti on lähetetty. Viimeisessä vaiheessa syötetään viesti, joka halutaan lähettää. Lopuksi painetaan ”TEE ILMOITUS” -painiketta, jolloin katoamisilmoitus lähetetään palvelimelle.



Kuva 22. Viestinäkymä

Viesti tallennetaan tietokantaan ja siihen liitetään viestin lähetysaika. Alla olevassa kuvassa on pätkä ohjelmakoodia, jossa lisätään kartalle nykyistä sijaintia osoittava sijaintimerkki.

```
168     @Override
169     public void onMapReady(GoogleMap _map){
170
171         GoogleMap gMap = _map;
172
173         LatLng currentLocation = new LatLng(location.getLatitude(),location.getLongitude());
174
175         MarkerOptions markerOptions = new MarkerOptions();
176         markerOptions.position(currentLocation);
177         markerOptions.title("Nykyinen sijaintisi");
178         float zoom = gMap.getMaxZoomLevel() - 8 ;
179
180         gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(currentLocation, zoom));
181         gMap.addMarker(markerOptions);
182     }
```

Kuva 23. Sijaintimerkin lisääminen karttaan

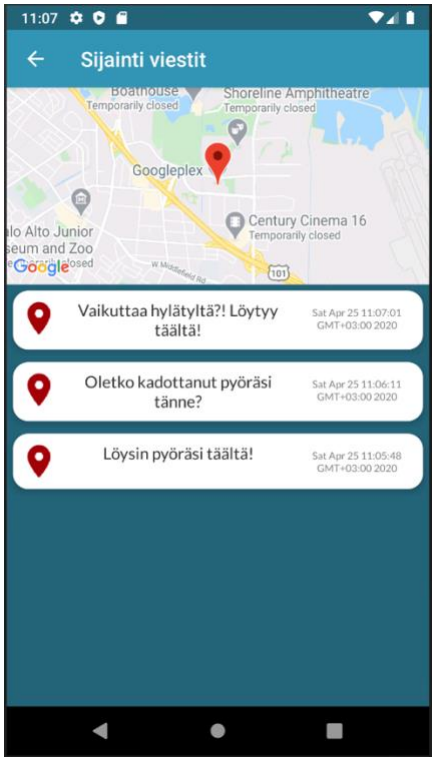
Aluksi luodaan viittaus Google-karttaan (Kuva 23, rivi 171). Seuraavaksi luodaan sijaintiobjekti, johon lisätään käyttäjän tämänhetkinen sijainti (Kuva 23, rivi 173).

Tämän jälkeen luodaan sijaintimerkin määritelmä (Kuva 23, rivi 175), johon tallennetaan äskettäin luotu sijaintiobjekti (Kuva 23, rivi 173), koska sijaintimerkki halutaan kartalle siihen kohtaan, jossa käyttäjä ilmoituksen luomishetkellä sijaitsee. Sijaintimerkin otsikoksi lisätään teksti: ”Nykyinen sijaintisi” (Kuva 23, rivi 177), joka tulee näkyviin, jos sijaintimerkkiä kosketetaan. Seuraavana koodissa luodaan lähennysobjekti (Kuva 23, rivi 178), joka sisältää tiedot lähennyksen tasosta. Lähennyksen tasoksi on asetettu maksimilähennys, josta on loitonnettu kahdeksan tasoa ulospäin.

Lopuksi siirretään kartta, animaatiota käyttäen, käyttäjän tämänhetkiseen sijaintiin samalla tarkentaen kartta oikealle etäisyydelle (Kuva 23, rivi 180) ja lisätään sijaintimerkki aiemman määritelmän mukaisesti oikealle paikalleen (Kuva 23, rivi 181).

4.1.7 Sijaintinäkymä

Sijaintinäkymässä tietokannasta haetaan viestejä polkupyörän runkonumeron perusteella. Viestit tulostetaan listaan (Kuva 24) kartan alapuolelle. Jos viestin jättäjä on lisännyt viestiin sijaintitiedot, viestiä painettaessa kartta lähentää sijaintiin.



Kuva 24. Sijaintinäkymä

Jotta karttaa pystyy käyttämään sovelluksessa, tarvitaan API-avain Google-tunnistepalvelimelta. Avaimen perusteella Google voi seurata karttapalvelun käyttömääriä ja jos käyttömäärät ylittävät ilmaisen käytön rajan, kartan käytöstä peritään maksu sovelluskehittäjältä. Tätä avainta ei saa sisällyttää ohjelmakoodiin, koska ulkopuolinen taho pystyy urkkimaan sen ohjelmakoodista. Sen sijaan se tulee tallentaa Firebase-tietokantaan, josta valtuutetuilla tunnuksilla se voidaan hakea salattujen yhteyksien välityksellä. Näin ollen API-avain voidaan myös tarvittaessa vaihtaa ilman tarvetta ohjelmistopäivitykselle.

4.2 Tietokantayhteyksien toteutustapa

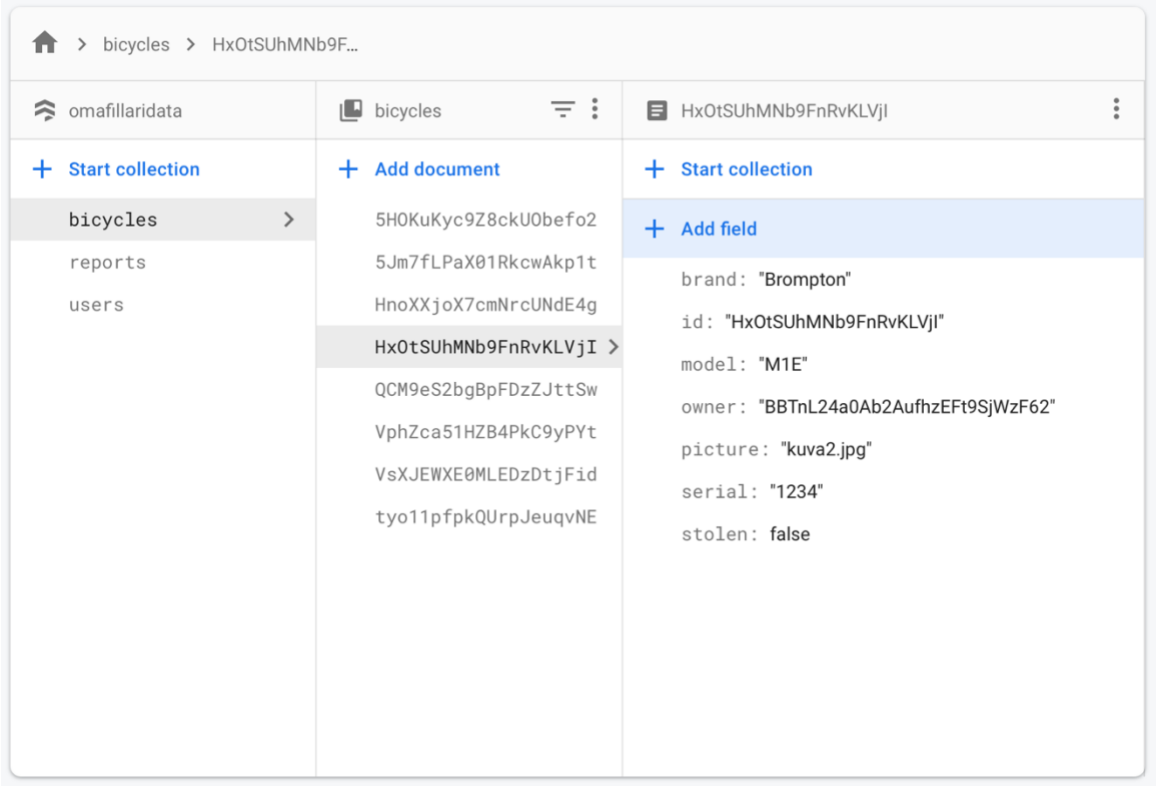
Tietokantayhteydet toteutettiin suunnittelu vaiheessa tehtyjen suunnitelmien mukaisesti luomalla Cloud Firestore -tietokanta. Cloud Firebase koostuu kokoelmista (Collection), joiden sisällä säilötään dokumentteja (Document) ja dokumentit sisältävät kenttiä (Field), joissa tieto on tallennettuna. Tietokannan rakennetta ei suunnitteluvaiheessa määritelty, joten se suunniteltiin ja toteutettiin sovelluksen kehitystyön ohessa.

Projektia varten luotiin uusi Cloud Firebase -tietokanta, johon lisättiin kolme dokumenttikokoelmaa. Kokoelmien nimet ovat: ”bicycles”, ”reports” ja ”users”. Ensimmäinen dokumenttikokoelma sisältää kaikki järjestelmään lisätyt polkupyörät. Toinen sisältää katoamisilmoitukset ja viimeinen sisältää kaikki rekisteröityneet käyttäjät.

Dokumenttikokoelma polkupyöristä sisältää täten polkupyörädokumentteja, joista yksittäinen polkupyörädokumentti (Kuva 25) sisältää kentät: merkki, id-tunniste, polkupyörän malli, polkupyörän omistaja, kuvan tiedostonimi, polkupyörän runkonumero ja tieto siitä onko polkupyörä merkitty kadonneeksi.

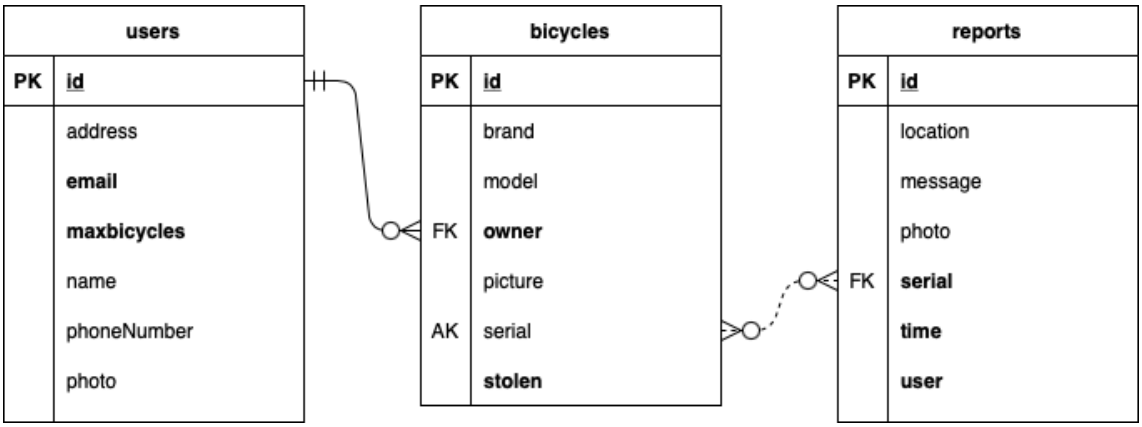
Käyttäjädokumentti sisältää kentät: osoite, sähköposti, käyttäjän ostamien polkupyöräpaikkojen määrä, nimi, puhelinnumero ja profiilikuvan tiedostonimi. Polkupyörädokumentti on linkitetty käyttäjään polkupyörän omistaja kentän avulla, jolloin voidaan tarvittaessa hakea tietokannasta kunkin käyttäjän omistamat polkupyörät.

Katoamisilmoitusdokumentti sisältää kentät: sijainti, viesti, valokuvan tiedostonimi, löydetyn polkupyörän runkonumero, löytymisaika ja raportin jättäjän id-tunniste. Raportit linkitetään polkupyörään polkupyörän runkonumero-kentän avulla.



Kuva 25. Polkupyörä tietue Firebase-konsolinäkymässä

Tietokannan rakenne on melko yksinkertainen. Tietokannan rakennekaaviosta (Kuva 26) nähdään, että polkupyörällä on mahdollista olla vain yksi käyttäjä, mutta käyttäjällä voi olla useampia polkupyöriä. Polkupyörää ei voi olla olemassa ilman omistajaa.



Kuva 26. Tietokannan rakenne

Kaaviosta nähdään myös, että löytymisilmoituksella ei tarvitse olla tietokannassa vastaavaa polkupyörää, tai se voi vastata useampaa polkupyörää. Tämä johtuu siitä, että polkupyörien runkonumerot eivät ole täysin uniikkeja ja on mahdollista,

että kahdessa polkupyörässä on sama runkonumero. On myös mahdollista, että sovelluksella jätetään löytymisilmoitus polkupyörästä, jota ei ole vielä rekisteröity sovellukseen. Se tallennetaan siitä huolimatta tietokantaan. Jos polkupyörä lisätään myöhemmin tietokantaan, on siitä jo löytymisilmoitus saatavilla.

4.3 Käyttöliittymän toiminta

Käyttäjällä on ajatusmalli ohjelman toiminnasta, jota hän lähtee kokeilemaan avattuaan ohjelman ensimmäistä kertaa (Wiio 2004, 151). Tämä ajatusmalli on muodostunut aikaisemmin käytettyjen ohjelmien käyttökokemuksen myötä. OmaFillari-sovelluksen käyttöliittymän on tarkoitus myötäillä tätä aikaisemmin opittua ajatusmallia sijoittamalla painikkeet ja toiminnot paikkoihin, joista käyttäjä on tottunut niitä etsimään. Käyttäjä ei halua luottaa navigoinnissa ulkomuistiin, vaan logiikkaan, jonka vuoksi sovelluksen jokainen näkymä on pyritty toteuttamaan loogisen navigoinnin ehdoilla (Wiio 2004, 154).

Käyttöliittymä on tehty yksinkertaiseksi välttämällä liikaa syvyyttä, jotta sen käyttö on aloitteleville ja satunnaisille käyttäjille helppoa (Wiio 2004, 164). Edistyneemmät asetukset on piilotettu asetusvalikkoon, eli sovelluksen suunnittelussa on käytetty kerrostetun käyttöliittymän periaatetta (Wiio 2004, 163). Edistyneemmät ominaisuudet on haluttu pitää mukana, vaikkakin kätkettynä asetusvalikkoon, koska sovelluksesta ei tulisi tehdä käyttäjän tarpeita yksinkertaisempaa (Wiio 2004, 167). Jos käyttäjällä on aikaisempaa kokemusta mobiilisovelluksista, hän osaa etsiä edistyneempää sisältöä asetusvalikoista (Wiio 2004, 153).

4.4 Käyttöliittymän ulkoasu

Jokaisella ohjelmalla täytyy olla päänäyttö (Wiio 2004, 175) ja tässä sovelluksessa se on näyttö, joka sisältää listan polkupyöristä. Näyttöjen syvyys on rajoitettu kahteen näyttöön. Päänäytöstä pääsee yhteen alinäyttöön, joka voi vielä johtaa kolmanteen näyttöön, joka tukee tai muokkaa edellisten näytön sisältöä, vaikkakin kolmanteen näyttöön siirtymistä on vältetty sovelluksen selkeyden vuoksi. Eri näyttöjen välillä pyrittiin pitämään mahdollisimman

samanlainen ilme, koska epäyhtenäinen ilme näyttöjen välillä viestii laadun puutetta (Wiio 2004, 217). Sovelluksen elementtien väreinä käytettiin suunnitteluvaiheessa valittuja värejä.

Toteutusvaiheessa kiinnitettiin huomiota sovelluksen kirjaisintyypin luettavuuteen. Yleisesti kirjallisuudessa pidetään antikvakirjaimia luettavampana kuin groteskikirjaimia (Wiio 2004, 204). Antikvakirjaimissa on päätteviivat, jotka ohjaavat lukijan katsetta tekstin rivien mukaan, jolloin rivit näyttävät selkeämmiltä. Kuitenkin groteski toistuu paremmin näytöillä, joissa tarkkuus ei ole yhtä hyvä kuin painotuotteissa (Wiio 2004, 208). Groteskissa silmukat ja kaaret ovat väljempiä, jolloin ne ovat selkeämpiä näytöllä. Nykyaikaiset mobiililaitteiden näytöt ovat kuitenkin erittäin tarkkoja ja eroa painotuotteeseen on vaikea havaita, jonka vuoksi myös antikvakirjaimet olivat varteenotettava vaihtoehto. Kirjaisintyypin valinta antikvan ja groteskin välillä kääntyi kuitenkin groteskin puoleen, koska se on selkempi lukea, jos rivejä ei ole useampia. Yksirivisissä teksteissä rivin seuranta ei ole vaikeaa, jolloin tekstin selkeys voidaan nostaa etusijalle ja tällöin voidaan käyttää groteskia (Wiio 2004, 207). Usein myös groteskin käyttö on jopa parempi vaihtoehto, kun tekstiä on vähän, siksi sitä käytetään usein väliotsikoiden kirjaisintyyppinä (Wiio 2004, 206).

Sovelluksessa käytetään vain yhtä fonttia, koska useampien eri fonttien käyttö voi hankaloittaa lukemista (Wiio 2004, 208). Fontiksi valittiin Łukasz Dziedzicin suunnittelema Lato, joka julkaistiin vuonna 2010. Oletusfontin korvaamisella haluttiin saada sovellukselle hieman persoonallisempi ja raikkaampi ilme.

5 TULOKSET JA POHDINTA

Kehitystyön tarkoituksena oli suunnitella ja toteuttaa OmaFillari-mobiilisovellus, jonka avulla hylätyn polkupyörän löytäjä voi ilmoittaa polkupyörän omistajalle löytäneensä hänen polkupyöränsä. Lisäksi sovellukselle oli tarkoituksena luoda taustapalvelut, johon sovelluksen tarvitsemat käyttäjätiedot tallennetaan. Lopputuloksena syntyi lähes valmis tuote, joka on mahdollista julkaista Google Play -kaupassa hyvin pienillä muutoksilla. Toteutusvaiheen valmistumishetkellä sovelluksessa käytössä oleva karttakomponentti on vanhentunut ja se ei ole

yhteensopiva tämänhetkisten Android-laitteiden kanssa. Karttakomponentti pitäisi korvata uudemmalla, jotta sovellus toimisi ja sen voisi julkaista Google Play -sovelluskaupassa. Sovelluksen kehittämiseen kului aikaa 137 tuntia, eli hieman yli 17 työpäivää. Sovellus täyttää sille suunnitteluvaiheessa asetetut tavoitteet ja laatuvaatimukset. Lopputulos on hyvin lähellä suunnitelmissa asetettuja tavoitteita, koska toteutusvaiheen aikana ei lähdetty rönsyilemään suunnitteluvaiheessa tehtyjen päätösten ulkopuolelle. Toteutusvaiheessa sovelluksen kehittämiseksi keksittiin muutama uusi idea, mutta niitä ei lähdetty toteuttamaan, vaan ne kirjattiin muistiin jatkokehitystä varten.

Omasta näkökulmastani opinnäytetyö onnistui toteutukseltaan hyvin, joskin aikataulussa pysymisessä epäonnistuttiin ja kirjallisen työn valmistuminen venyi hieman suunniteltua myöhemmäksi. Syynä tähän olivat ennen opinnäytetyön valmistumista alkaneet kesätyöt ja tästä johtuva hetkellinen motivaation puute. Toteutusvaiheessa ei kohdattu suuria ongelmia ja sovelluksen kehitys sujui joutuisasti. Sovelluksen kehitystyö olikin mielestäni projektin antoisin vaihe.

Sovellusta voisi jatkokehittää lisäämällä siihen mahdollisuuden myydä ja ostaa käytettyjä polkupyöriä. Koska käyttäjän polkupyörien tiedot ovat jo valmiiksi syötetty sovellukseen, tarvitsisi käyttäjän ainoastaan painaa myyntipainiketta myytäväksi haluttavan polkupyörän kohdalta, jolloin polkupyörä siirtyisi myytävien pyörien kategoriaan. Ostajan löydyttäessä omistajatiedot voisi myös siirtää sovelluksen kautta uudelle pyörän omistajalle puhelimen näytöllä näytettävän QR-koodin avulla. Sovellukseen voisi myös lisätä NFC-tarrojen lukuominaisuuden. Tällöin käyttäjä voisi liimata polkupyöräänsä NFC tarran, joka linkitettäisiin hänen pyöränsä tietoihin sovelluksen avulla. Tällöin polkupyörän pohjasta ei tarvitsisi lukea runkonumeroa, vaan tiedot saisi esille tuomalla älypuhelin tarran lähelle.

LÄHTEET

Bike Index. 2013. Polkupyörän rekisteröintiin tarkoitettu Internetsivu. Saatavissa: https://bikeindex.org [Viitattu 24.6.2020].

Bike-ID. 2015. Polkupyörän rekisteröintiin tarkoitettu Internetsivu. Saatavissa: https://www.bike-id.eu [Viitattu 24.6.2020].

BikeRegister. 2004. Polkupyörän rekisteröintiin tarkoitettu Internetsivu. Saatavissa: https://www.bikeregister.com [Viitattu 24.6.2020].

Color Hunt. 2015. Color palettes for designers and artist. WWW-dokumentti. Saatavissa: https://colorhunt.co [Käytetty 15.3.2020].

Condon, S. 2019. Google's Cloud Firestore is now generally available, ZDNet. Verkkoartikkeli. Saatavissa: https://www.zdnet.com/article/googles-cloud-firestore-is-now-generally-available/ [Viitattu 23.6.2020].

Edlich, S. 2020. NoSQL Databases list. WWW-dokumentti. Saatavissa: https://hostingdata.co.uk/nosql-database/ [Viitattu: 23.6.2020].

Polkupyörärekisteri. 2017. WWW-dokumentti. Saatavissa: https://www.pprek.fi/ [Viitattu 24.6.2020].

Sinkkonen I., Kuoppala H., Parkkinen J. & Vastamäki R. 2006. Käytettävyyden psykologia. 3. uud. p. Helsinki: IT Press, Edita.

StatCounter. 2020. WWW-dokumentti. Saatavissa: https://gs.statcounter.com/os-market-share/mobile/worldwide [Viitattu 5.8.2020].

Wiio, A. 2004. Käyttäjäystävällisen sovelluksen suunnittelu. 1. p. Helsinki: IT Press, Edita.

Yahiaoui, H. 2017. Firebase Cookbook.

KUVALUETTELO