OULUN AMMATTIKORKEAKOULU

SIIRTOAUTOJEN MOBIILISOVELLUS: KÄYTTÄJÄYSTÄVÄLLINEN RATKAISU AUTOJEN HALLINTAAN JA VARAAMISEEN

Osman Akbaba & Tuukka Huru
Opinnäytetyö AMK

Kevät 2025
Tieto- ja viestintätekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistokehitys

Tekijät: Osman Akbaba & Tuukka Huru
Opinnäytetyön otsikko: Siirtoautojen mobiilisovellus: Käyttäjäystävällinen ratkaisu autojen varaamiseen ja hallintaan
Työn ohjaaja: Jari Kiiskinen
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 61

Tässä opinnäytetyössä kehitettiin mobiilisovellus siirtoautojen hallintaan ja varaamiseen. Työn tavoitteena oli luoda käyttäjäystävällinen, sekä tehokas prototyyppi, joka yhdistää siirtoautojen etsimisen, varaamisen ja hallinnan yhdeksi kokonaisuudeksi. Sovellus kehitettiin käyttäen React Nativea, Expoa ja Firebasea. Näiden teknologioiden yhdistelmä mahdollistaa sovelluksen kehittämisen eri alustoille, kuten iOS:lle ja Androidille, yhdellä koodilla. Firebase tarjoaa skaalautuvan ja tehokkaan tietokannan, joka tukee sovelluksen kasvua ja mahdollistaa reaaliaikaisen tiedonhallinnan. Firebase Authentication takasi turvallisen käyttäjähallinnan, ja Firestore tarjosi tehokkaan ja joustavan tietokantaratkaisun.

Sovelluksen päätoiminnot keskittyivät autojen etsimiseen, varaamiseen ja hallintaan, mutta myös autojen lisääminen ja karttapalveluiden käyttö olivat keskeisiä osia. Käyttäjät voivat hakea autoja hakukriteerien avulla ja varata ne. Karttaominaisuudet mahdollistavat ajoneuvojen sijainnin tarkastelun sekä reitin näyttämisen valittuun autoon. Sovellus tukee sekä henkilöasiakkaita että yritysasiakkaita, joissa henkilöasiakkaat voivat varata autoja, ja yritykset voivat lisätä ajoneuvojaan sovellukseen ja hallita niitä. Projektissa hyödynnettiin ketteriä menetelmiä, kuten Scrumia, ja suunnittelussa käytettiin Figmaa ja vuokaavioita, jotka auttoivat hahmottamaan sovelluksen logiikkaa ja parantamaan käyttäjäkokemusta.

Sovelluksen kehittämisessä saavutettiin asetetut tavoitteet, ja lopputuloksena saatiin toimiva prototyyppi, joka täyttää sekä henkilöasiakkaiden että yritysasiakkaiden tarpeet. Kehitystyössä opittiin sovelluksen teknisestä toteutuksesta, projektinhallinnasta ja tiimityöskentelystä. Tulevaisuudessa sovellusta on mahdollista laajentaa uusilla toiminnoilla ja liiketoimintamahdollisuuksilla.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information and Communication Technology
Option of Software Development

Authors: Osman Akbaba & Tuukka Huru
Title of thesis: Mobile Application for Vehicle Relocation: A User-Friendly Solution for Vehicle Reservation and Management
Supervisor: Jari Kiiskinen
Term and year when the thesis was submitted: Spring 2025
Number of pages: 61

This thesis focuses on the development of a mobile application for managing and booking vehicles, specifically aimed at vehicle relocation. The primary objective was to design a functional and user-friendly prototype that enables users to easily search, book, and manage vehicles. The application was built using React Native, Expo, and Firebase, offering cross-platform functionality and scalability. Firebase was used to handle user authentication and store data in real time, allowing seamless data synchronization across devices.

The key functionalities of the application include vehicle search, booking, and management, with map integration to view vehicle locations and routes. The development process also incorporated agile methodologies, such as Scrum.

The project successfully met its goals, resulting in a working prototype that serves both individual and business users. The development process provided valuable insights into mobile application development, project management, and teamwork. In the future, the application can be expanded with additional features and business models to increase its functionality and market potential.

SISÄLLYS

1 JOHDANTO

Siirtoautojen käyttö on lisääntynyt merkittävästi viime vuosina, sillä ne tarjoavat kustannustehokkaan ja joustavan tavan kuljettaa ajoneuvoja paikasta toiseen. Esimerkiksi autoliikkeet siirtävät myytyjä autoja asiakkaiden sijaintiin tai toisiin toimipisteisiin, ja vuokrausyritykset siirtävät ajoneuvoja kysynnän mukaan eri toimipaikkojen välillä. Digitaaliset ratkaisut ovat kuitenkin jääneet jälkeen alalla, jossa siirtoautojen hallinta ja varaaminen tapahtuvat yhä suurelta osin manuaalisesti tai hajanaisten järjestelmien kautta. Tämä opinnäytetyö keskittyy ongelman ratkaisuun kehittämällä mobiilisovelluksen, joka yhdistää siirtoautojen hakemisen, varaamisen ja autojen hallinnan yhdeksi saumattomaksi kokonaisuudeksi. Mobiilisovellus valittiin ratkaisuksi, koska se mahdollistaa varausten tekemisen ajasta ja paikasta riippumatta, tarjoaa paremman käyttökokemuksen kuin perinteiset verkkosivut.

Tämä opinnäytetyö toteutetaan ryhmäprojektina ja perustuu nykyaikaisiin ohjelmistokehitysmenetelmiin ja teknologioihin. Projektin tavoitteena on luoda prototyyppi mobiilisovelluksesta, joka mahdollistaa käyttäjille siirtoautojen selaamisen ja varaamisen yksinkertaisella tavalla. Lisäksi sovellus tarjoaa yritysasiakkaille mahdollisuuden lisätä ja hallita palveluun liittämiään ajoneuvoja, mikä tehostaa heidän toimintaansa ja helpottaa palvelun ylläpitoa. Projekti tarjoaa myös mahdollisuuden kartoittaa alaan liittyviä haasteita ja kehittää ratkaisuja, jotka parantavat toimintaprosesseja sekä lisäävät asiakastyytyväisyyttä.

1.1 Työn tausta

Siirtoautojen hallinta on jäänyt digitalisaation kehityksessä jälkeen, mikä näkyy erityisesti manuaalisina varausprosesseina ja hajanaisina tietokantaratkaisuina. Nämä vanhentuneet ratkaisut, kuten Facebook-ryhmien kautta toimivat varaukset, ovat tehottomia ja epäkäytännöllisiä. Käyttäjän täytyy etsiä itse sopivia autoja, neuvotella ehdoista ja varmistaa saatavuus manuaalisesti. Tämä tekee prosessista hitaan ja epävarman erityisesti silloin, kun autoja tarvitaan nopeasti.

Tämä osoittaa selkeän tarpeen modernisoida alaa hyödyntämällä nykyaikaista teknologiaa.

Tämän työn lähtökohtana on kehittää mobiilisovellus, joka yhdistää siirtoautojen hallintaan liittyvät keskeiset toiminnot yhteen paikkaan. Sovellus pyrkii tarjoamaan käyttäjille helpon ja tehokkaan tavan varata autoja ja yritysasiakkaille mahdollisuuden hallita ajoneuvojaan järjestelmällisesti. Samalla sovellus ratkaisi nykyisen järjestelmän hajanaisuudesta johtuvat ongelmat. Näin työ pyrkii vastaamaan sekä yksittäisten käyttäjien että yritysten tarpeisiin.

1.2 Projektin rajaus ja aikataulu

Opinnäytetyön pääasiallisena tavoitteena on toimivan prototyypin kehittäminen, joka kattaa ydintoiminnallisuudet kuten käyttäjähallinnan, autojen hakutoiminnot, varaamisprosessin ja karttapalvelut. Projekti rajataan käsittelemään vain siirtoautojen hallintaan liittyviä keskeisiä toimintoja, ja mahdolliset jatkokehitykseen liittyvät ominaisuudet, kuten laajennettu analytiikka, jätetään prototyypin ulkopuolelle. Laajennettavuus on kuitenkin huomioitu arkkitehtuurissa, jotta uusia ominaisuuksia voidaan helposti lisätä myöhemmässä vaiheessa. Lisäksi sovelluksen käyttäjäryhmät, henkilöasiakkaat ja yritysasiakkaat, saavat eri ominaisuudet tarpeidensa mukaan. Yritysasiakkaat voivat lisätä ja hallita ajoneuvoja, kun taas henkilöasiakkaille tarjotaan yksinkertainen auton haku- ja varaustoiminto.

Projektin aikataulu seuraa selkeää sprinttimallia, jossa viikoittain asetetaan tavoitteet ja arvioidaan edistymistä sprint review -tilaisuuksissa. Kokonaisaikataulu on 12 viikkoa, ja työn on oltava valmis silloin. Aikataulun tiukka noudattaminen varmistaa, että kaikki työvaiheet saadaan toteutettua ajallaan. Näitä ovat esimerkiksi suunnittelu, toteutus ja dokumentointi.

1.3 Käytettävät menetelmät ja teknologiat

Projekti toteutetaan Scrum-menetelmiä soveltaen, mikä mahdollistaa joustavan mutta suunnitelmallisen kehitystyön kahden opiskelijan ryhmässä. Työn etenemistä tukee Kanban-taulu, jota hyödynnetään tehtävien jakamiseen, priorisointiin

ja aikatauluttamiseen. Kanban-taulu auttaa varmistamaan, että kaikki tehtävät pysyvät hallinnassa ja etenevät järjestelmällisesti.

Teknologioiksi valittiin React Native ja Expo, jotka tukevat mobiilisovelluksen monialustakehitystä. React Native valittiin, koska se mahdollistaa monialustakehityksen yhdellä koodipohjalla, mikä vähentää kehitystyön määrää verrattuna erillisiin Android- ja iOS-sovelluksiin. Expo puolestaan yksinkertaistaa sovelluksen kehitys- ja testausprosessia, jolloin kehittäjien ei tarvitse konfiguroida monimutkaisia natiiviasetuksia. Firebase valittiin backend-ratkaisuksi sen helppokäyttöisyyden ja skaalautuvuuden vuoksi. Firebase myös tarjoaa käyttäjähallinnan, tietokantapalvelut ja reaaliaikaisen datan synkronoinnin ilman tarvetta erilliselle palvelinympäristölle. Versionhallinta toteutetaan GitHubin avulla, ja kaikki kehitystyö dokumentoidaan huolellisesti. Suunnitteluvaiheessa käytettiin Figmaa, joka mahdollisti käyttöliittymäprototyyppien nopean luomisen ja tiimityön reaaliaikaisen yhteistyön. Tämä yhdistelmä takaa tehokkaan kehitysprosessin ja laadukkaan lopputuotteen.

1.4 Projektin merkitys ja tavoitteet

Siirtoautojen digitalisoitu hallinta tarjoaa merkittäviä etuja niin palveluntarjoajille kuin käyttäjillekin. Tehokkaammat prosessit voivat säästää aikaa ja rahaa, samalla kun ne mahdollistavat resurssien paremman hallinnan. Mobiilisovellus vastaa suoraan näihin tarpeisiin yksinkertaistamalla autojen varaamista ja hallintaa.

Projektin tavoitteena on luoda sovellus, joka tarjoaa henkilöasiakkaille helpon tavan löytää ja varata sopivia autoja sekä yritysasiakkaille työkalut hallita palveluun lisättyjä ajoneuvojaan. Tämä mahdollistaa sujuvamman ja nopeamman palvelun, joka hyödyttää kaikkia osapuolia.

Projektin merkitys on erityisen suuri siirtoautojen palveluiden kehittämiselle ja digitalisoinnille. Lisäksi projekti tarjoaa ryhmälle mahdollisuuden soveltaa oppimaansa tietoa ja kehittää työelämässä tarvittavia taitoja, kuten ohjelmistokehitystä, projektinhallintaa ja yhteistyötä. Projektin oppimistavoitteet ovat myös keskeisiä. Opinnäytetyön aikana ryhmä kehittää teknistä osaamistaan moderneissa ohjelmistokehityksen menetelmissä, kuten skaalautuvan prototyypin luomisessa

ja projektinhallinnassa. Samalla työ dokumentoidaan huolellisesti osana opinnäytetyöraporttia, mikä varmistaa työn tulosten ja opitun hyödynnettävyyden myös jatkossa. Projekti edistää myös oppilaitoksen tavoitteita tuottaa käytännönläheisiä ja vaikuttavia opinnäytetöitä.

2 SUUNNITELMAT JA MENETELMÄT

Projekti suunniteltiin alusta asti huolellisesti, jotta kehitystyö pysyisi hallittuna ja käyttäjien tarpeet huomioitaisiin jo varhaisessa vaiheessa. Koska sovellus on monipuolinen ja palvelee sekä henkilö- että yritysasiakkaita, sen rakenteen ja toiminnallisuuksien tuli olla selkeitä. Suunnitteluvaiheen tavoitteena oli varmistaa, että sovellus tarjoaa sujuvan ja tehokkaan käyttökokemuksen kaikille käyttäjäryhmille.

Projektissa sovellettiin ketteriä ohjelmistokehitysmenetelmiä, jotka mahdollistivat nopeat muutokset ja jatkuvan palautteen hyödyntämisen kehitysprosessissa. Teknologioiden valinnassa painotettiin erityisesti kehityksen tehokkuutta, sovelluksen käytettävyyttä sekä skaalautuvuutta tulevia laajennuksia varten.

Käyttöliittymän suunnittelussa hyödynnettiin moderneja työkaluja, kuten Figmaa ja Microsoft Visiota, joiden avulla sovelluksen navigaatio ja visuaalinen ilme voitiin suunnitella ennen varsinaista kehitystyötä. Vuokaavioita käytettiin sovelluksen logiikan hahmottamiseen, mikä auttoi varmistamaan, että kaikki toiminnot oli suunniteltu käyttäjäystävällisesti ja tehokkaasti.

2.1 Scrum-menetelmien käyttö sovelluskehityksessä

Tässä projektissa Scrum-menetelmän käyttö on valittu kehitysmenetelmäksi, koska se soveltuu hyvin pieniin tiimeihin ja mahdollistaa jatkuvan palautteen hyödyntämisen kehitystyössä. Verrattuna perinteiseen vesiputousmalliin, jossa projekti etenee vaiheittain ja muutokset voivat aiheuttaa viivästyksiä, Scrum antaa mahdollisuuden nopeisiin muutoksiin sprinttien välillä. Scrum-prosessin keskeisiä osia ovat sprintit, sprint suunnitelmat, päivittäiset kokoontumiset sekä sprint review ja retrospektiivi. (Parsonen 2024.)

Projektissa sprinttien pituudeksi on valittu yksi viikko, jolloin ryhmä voi keskittyä lyhyen aikavälin tavoitteisiin ja mukauttaa suunnitelmia palautteen perusteella. Jokaisen sprintin alussa määritellään sprint-goals, jotka ohjaavat kehitystyötä ja

varmistavat että tehtävät etenevät suunnitellusti. Scrum-menetelmää ja Kanban-työkaluja käytetään tehtävien hallintaan ja edistymisen seurantaan.

Päivittäisissä kokoontumisissa käydään läpi ryhmän edistymistä, mahdollisia esteitä ja päivän tavoitteita. Tämä auttaa varmistamaan, että molemmat ryhmän jäsenet ovat ajan tasalla ja voivat reagoida mahdollisiin haasteisiin nopeasti. Lisäksi sprint review -sessiossa esitellään saavutetut tulokset ja kerätään palautetta, joka ohjaa seuraavan sprintin suunnittelua.

2.2 Teknologioiden valinta

Tämän projektin teknologiat valittiin käyttäjäystävällisyyden, kehityksen tehokkuuden ja ylläpidon näkökulmasta. Frontend-kehitykseen valittiin React Native -kirjaston, koska se mahdollistaa sovelluksen kehittämisen yhdellä koodipohjalla sekä Androidille että iOS:lle. Tämä vähentää kehitykseen käytettyä aikaa ja resursseja, sillä erillisiä koodipohjia ei tarvitse ylläpitää. React Native tarjoaa myös laajan kirjaston komponentteja ja työkaluja, joiden avulla voidaan luoda responsiivisia ja dynaamisia käyttöliittymiä. (React Native 2025a; React Native 2025c.)

Expo valittiin kehitysympäristöksi, sillä se yksinkertaistaa mobiilisovelluksen rakentamista, jakelua ja testaamista ilman natiivikohtaisia määrityksiä. Expo tarjoaa valmiita työkaluja, kuten kamera- ja karttapalvelujen integraatioita, sekä mahdollisuuden testata sovellusta suoraan mobiililaitteilla. Tämä mahdollistaa tehokkaan kehitysprosessin, erityisesti pienissä tiimeissä. (Expo s.a. a.)

Backend-ratkaisuna käytettiin Firebase-alustaa, joka tarjoaa kattavat toiminnot käyttäjänhallintaan, tietokantapalveluihin ja reaaliaikaiseen datan synkronointiin. Firebase integroituu hyvin React Nativeen, mikä vähentää konfigurointiin kuluvaa aikaa ja toimii ilman että pitää ylläpitää omaa palvelinta. Tämä säästää aikaa ja mahdollistaa nopeamman käyttöönoton verrattuna esimerkiksi MySQL-pohjaiseen ratkaisuun, jossa olisi tarvittu oma palvelin ja manuaalista hallintaa. Erityisesti Firebase Authentication mahdollistaa turvallisen ja nopean käyttäjäkirjautumisen toteuttamisen ilman, että meidän täytyy itse kehittää monimutkaisia tunnistautumisjärjestelmiä. (Firebase 2025b.)

Tietokantaratkaisuksi valittiin Firestore-tietokanta, koska se tarjoaa skaalautuvan ja helposti hallittavan taustajärjestelmän ilman tarvetta ylläpitää omaa palvelininfrastruktuuria. Firestore tukee reaaliaikaisia tietopäivityksiä ja offline-tilaa, mikä parantaa sovelluksen käyttökokemusta myös ilman jatkuvaa verkkoyhteyttä. Firestore mahdollistaa myös rakenteiden helpon muutoksen ja monimutkaisempien kyselyiden käsittelyn, mikä on tärkeää sovelluksen tulevien laajennusten kannalta. (Firebase 2025a.)

Kaikki valitut teknologiat tukevat modernia ja tehokasta kehitystä, mahdollistavat sovelluksen skaalautuvuuden ja tarjoavat käyttäjäystävällisen kokemuksen. Valinnat sopivat erityisesti pienelle kehitystiimille, jossa on tärkeää hyödyntää ajankäyttöä tehokkaasti ja minimoida ylläpidon monimutkaisuus.

2.3 Käyttäjätarpeiden määrittely

Projektin alkuvaiheessa tutkittiin muita samalla alalla toimivia yrityksiä ja heidän palveluvalikoimaansa, joiden pohjalta määriteltiin sovelluksen keskeiset toiminnallisuudet. Tutkimme, millaisia palveluita markkinoilla on saatavilla, kuten Flovi (Flovi s.a.), Facebookin Siirtoautot Suomi -ryhmä ja Hertz (Lempinen 2025). Tavoitteena oli tunnistaa, mitä ominaisuuksia käyttäjät arvostavat nykypäivänä ja mitkä seikat voisivat parantaa nykyisiä ratkaisuja.

Käyttäjät arvostavat erityisesti helppokäyttöistä käyttöliittymää, jossa tärkeimmät toiminnot löytyvät nopeasti ilman monimutkaisia valikoita. Auton varaamiseen liittyvän tiedon, kuten saatavuuden, tulee olla esillä selkeästi ja helposti ymmärrettävässä muodossa, jotta käyttäjät voivat tehdä päätöksiä nopeasti ja vaivattomasti. Yritysasiakkaille tärkeää on tehokas työkalu autojen hallintaan ja varausten käsittelyyn, joka mahdollistaa ajoneuvojen lisäämisen, muokkaamisen ja varausprosessien hallinnan sujuvasti.

Näiden löydösten pohjalta suunniteltiin sovelluksen käyttöliittymä niin, että se tarjoaa selkeän ja luontaisen käyttökokemuksen. Käyttäjien tarpeita huomioitiin myös teknologian valinnassa, sillä React Native ja Firebase mahdollistavat responsiivisen ja nopean käyttökokemuksen.

2.4 Käyttöliittymän ja toiminnallisuuksien suunnittelu

Sovelluksen käyttöliittymän ja toiminnallisuuksien suunnittelu aloitettiin projektin alkuvaiheessa, jotta käyttäjäkokemus saatiin muokattua mahdollisimman selkeäksi. Suunnittelussa hyödynnettiin Figmaa ja Microsoft Visiota, jotka mahdollistivat käyttöliittymän hahmottamisen jo ennen varsinaisen kehitystyön aloittamista. Figma ja vuokaaviot tukivat suunnittelutyötä merkittävästi ja auttoivat hahmottamaan sekä teknisen toteutuksen että käyttäjäkokemuksen vaatimukset jo ennen varsinaista kehitysvaihetta. Yhteistyö ja suunnitelmien säännöllinen päivittäminen paransivat lopputuloksen laatua ja tekivät kehitysprosessista sujuvamman.

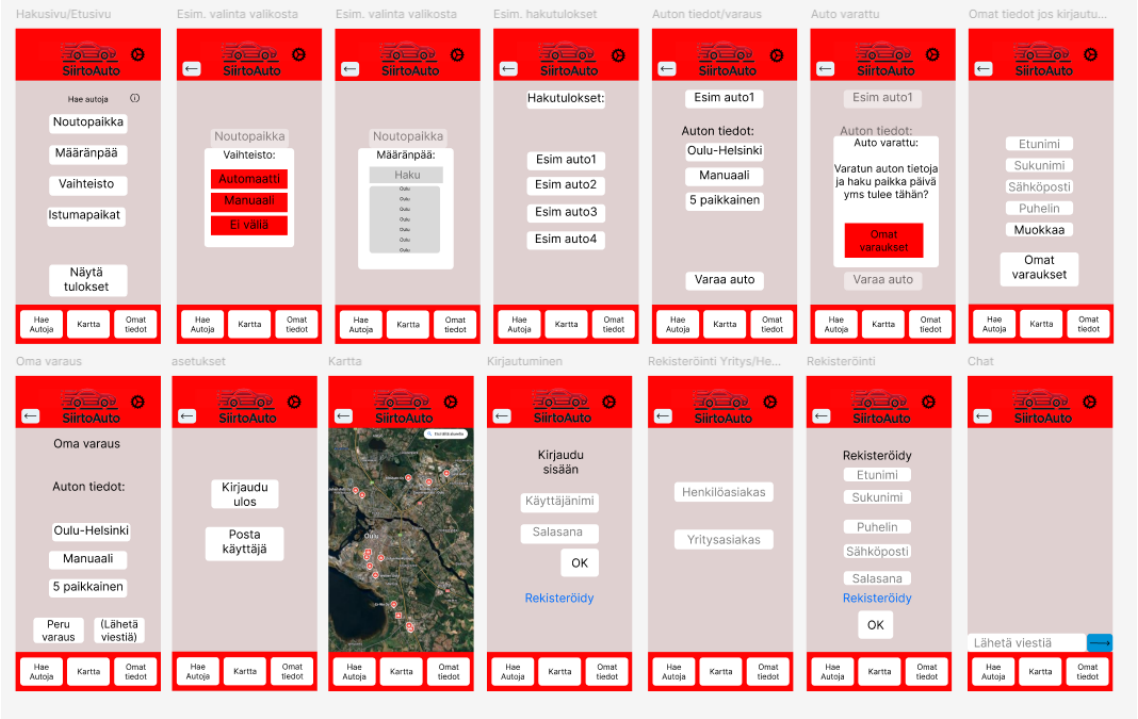
2.4.1 Figma suunnittelu

Figma valittiin käyttöliittymäsuunnittelun työkaluksi sen selainpohjaisuuden, reaaliaikaisen yhteistyömahdollisuuden ja monipuolisten ominaisuuksien vuoksi. Koska projekti toteutettiin ryhmätyönä, oli tärkeää, että molemmat kehittäjät pystyivät työskentelemään käyttöliittymän suunnittelun parissa yhtä aikaa ilman erillisiä ohjelmistoasennuksia. Figma toimii suoraan selaimessa, mikä teki sen käytöstä joustavaa ja mahdollisti suunnitelmien nopean jakamisen ilman tiedostojen erillistä synkronointia tai tallentamista.

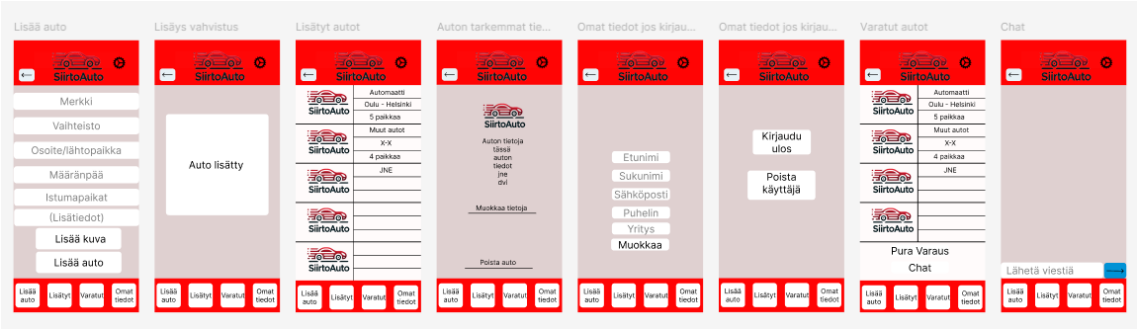
Toinen keskeinen syy Figman valintaan oli sen hinnoittelumalli. Figma tarjoaa ilmaisessa versiossaan mahdollisuuden työskennellä enintään kolmen projektin parissa ilman aikarajoitusta, mikä riitti hyvin tämän projektin tarpeisiin. Vastaavasti Adobe XD tarjoaa ainoastaan 7 päivän ilmaisen kokeilujakson, jonka jälkeen ohjelmiston käyttö olisi vaatinut maksullisen tilauksen. Koska projektin aikataulu oli useita viikkoja, Adobe XD:n kokeilujakso ei olisi riittänyt, ja maksulliseen ohjelmistoon lisenssin ostaminen ei ollut perusteltua.

Kaiken kaikkiaan Figma valittiin, koska se tarjosi joustavamman, monipuolisemman ja tehokkaamman tavan työskennellä tiiminä verrattuna Adobe XD:hen. Selainpohjaisuus, maksuttomuus sekä vahvat komponenttipohjaiset työkalut tekivät siitä optimaalisen valinnan tämän projektin suunnitteluun. (Corellia 2025.)

Käyttöliittymän suunnittelun tavoitteena oli luoda looginen ja helposti käytettävä sovellusrakenne, jossa kaikki tärkeät toiminnot, kuten auton haku, varaaminen ja hallinta, löytyvät vaivattomasti. Figmaan luotiin ensin perusrakenne, missä hahmoteltiin sovelluksen päänäkymät ja niiden väliset siirtymät. Näitä malleja muokattiin ja tarkennettiin säännöllisesti projektin edetessä. Tämä lähestymistapa varmisti, että sovelluksen käytettävyys ja navigaatio vastasivat käyttäjien tarpeita. (Figma s.a.) (KUVAT 1. ja 2.)



KUVA 1. Henkilöasiakkaan Figma-suunnitelma



KUVA 2. Yritysasiakkaan Figma-suunnitelma

2.4.2 Vuokaaviot

Vuokaavioiden avulla havainnollistettiin eri prosessien kulku sovelluksessa, kuten auton varaaminen ja hallinta. Niiden päivittämistä jatkettiin koko projektin ajan, sillä sovelluksen rakenteeseen ja toimintoihin tehtiin muutoksia kehitystyön edetessä. Vuokaavioiden avulla pystyttiin varmistamaan, että kaikki toiminnallisuudet olivat loogisia. Vuokaavioiden avulla projektiryhmä hahmotti sovelluksen logiikkaa kehitystyön aikana, sillä niiden avulla pystyttiin selkeästi näkemään, mihin eri sivuilta siirrytään ja miten eri toiminnot liittyvät toisiinsa.

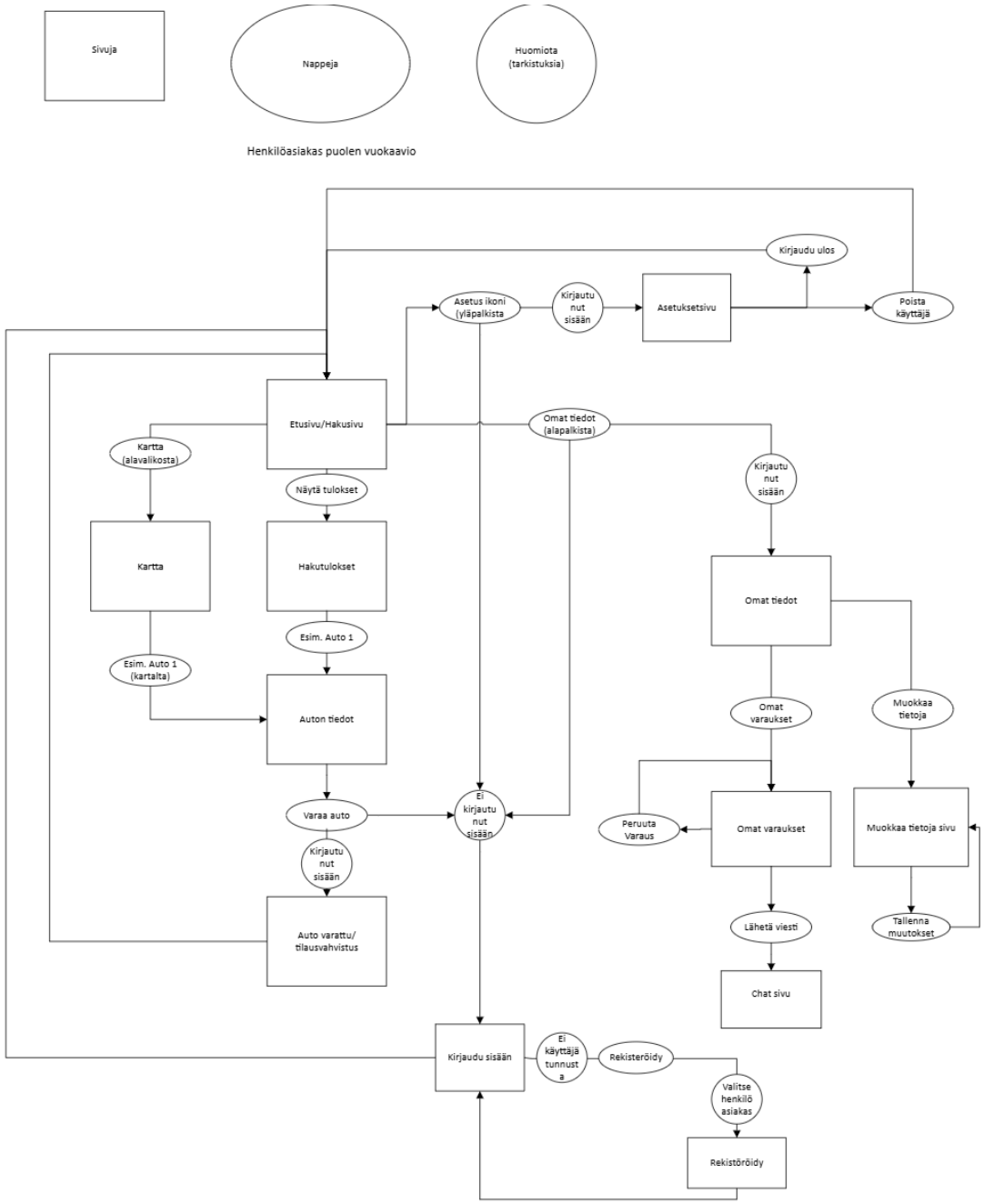
2.4.3 Henkilöasiakkaan vuokaavio

Henkilöasiakkaan vuokaavio kuvaa käyttäjän etenemisen sovelluksessa. Sovellus alkaa etusivulta, jossa käyttäjä voi siirtyä alapalkin kautta karttanäkymään, omiin tietoihin tai takaisin etusivulle. Alapalkki pysyy näkyvillä kaikissa näkymissä, mikä mahdollistaa nopean siirtymisen eri sivujen välillä ilman, että käyttäjän täytyy palata takaisin alkuun.

Etusivulla on "Näytä tulokset" -nappi, jonka kautta käyttäjä siirtyy hakutulosnäkymään. Hakutulosnäkymästä voi valita yksittäisen auton ja siirtyä auton tietosivulle. Auton tietosivulla on "Varaa auto" -nappi, jonka painaminen edellyttää kirjautumista sisään. Mikäli käyttäjä ei ole kirjautunut, hänet ohjataan kirjautumissivulle, jossa on mahdollisuus rekisteröityä valitsemalla "Rekisteröidy"-nappi. Rekisteröitymisen yhteydessä käyttäjä valitsee roolikseen henkilöasiakkaan. Kirjautumisen jälkeen käyttäjällä on mahdollisuus varata auto.

Auton varauksen jälkeen käyttäjä siirtyy varausvahvistusnäkymään. Omat varaukset löytyvät omien tietojen sivulta, johon pääsee alapalkin kautta. Omat varaukset -sivulla on mahdollista peruuttaa varauksia tai siirtyä chat-näkymään painamalla "Lähetä viesti" -nappia. Chat-sivulla käyttäjä voi lähettää viestejä yritykselle varaukseen liittyen.

Asetusnäkymä avautuu asetuskuvakkeen kautta. Asetussivulla käyttäjä voi kirjautua ulos tai poistaa käyttäjätilinsä. Uloskirjautumisen tai käyttäjän poistamisen jälkeen käyttäjä ohjataan takaisin etusivulle uloskirjautuneena. (KUVA 3.)



KUVA 3. Henkilöasiakkaan vuokaavio

2.4.4 Yritysasiakkaan vuokaavio

Yritysasiakkaan vuokaavio havainnollistaa käyttäjän etenemisen sovelluksessa sen jälkeen, kun hän on jo kirjautunut sisään yrityskäyttäjänä. Sovellus avautuu aloitussivulle, josta käyttäjä voi navigoida alapalkin kautta lisättyihin autoihin, varattuihin autoihin tai omiin tietoihin. Yläpalkissa on asetuskuvake, jonka kautta pääsee asetusnäkymään.

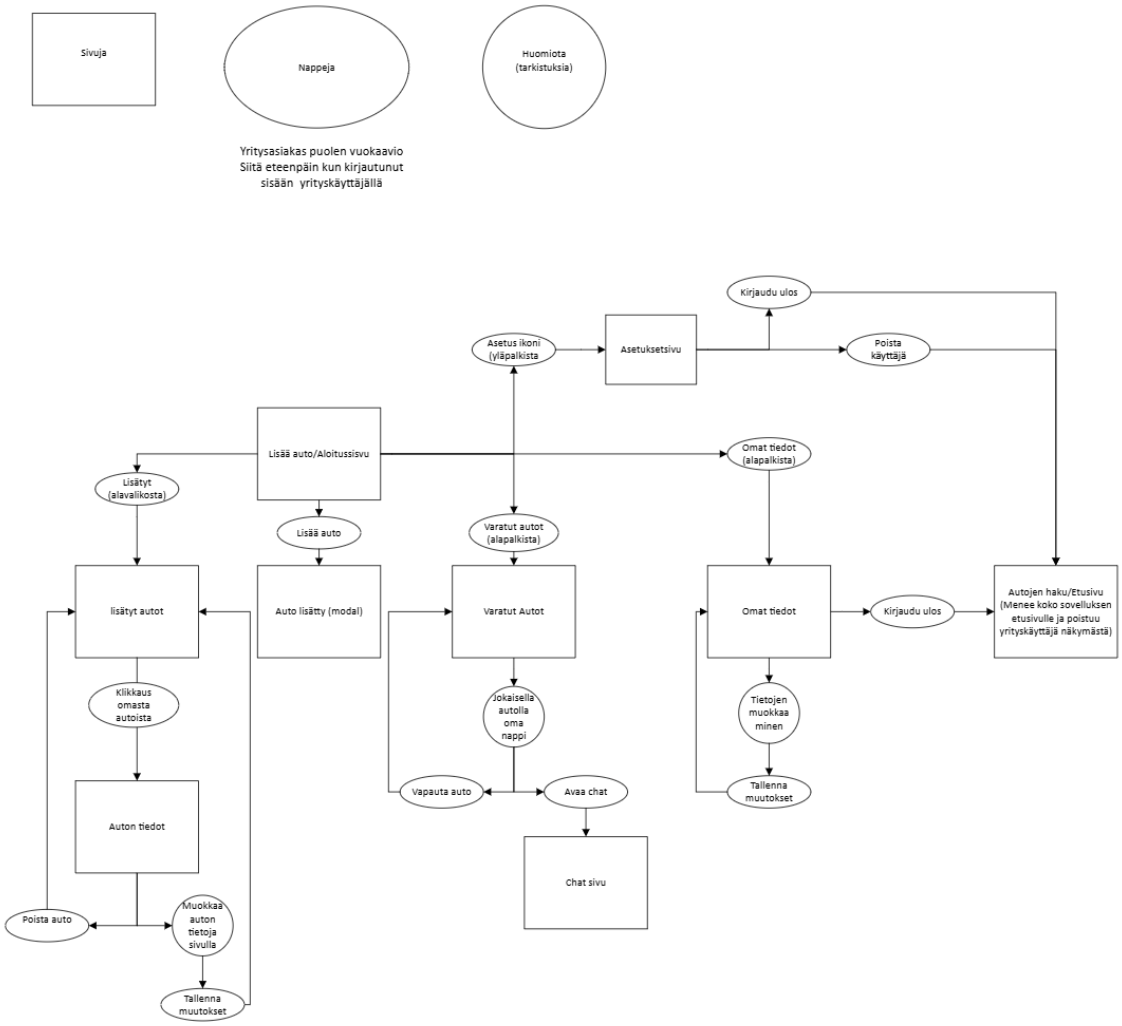
Aloitussivulla yritysasiakas voi lisätä uuden auton syöttämällä auton tiedot ja painamalla "Lisää auto" -nappia, minkä jälkeen sovellus näyttää "Auto lisätty" -vahvistuksen. Tämän jälkeen auto tallentuu tietokantaan ja ilmestyy lisättyjen autojen näkymään.

"Lisätyt autot" -näkymässä yrityskäyttäjä näkee listan kaikista lisäämistään autoista. Klikkaamalla haluamaansa autoa käyttäjä siirtyy auton tietoihin, missä hän voi muokata tietoja tai poistaa auton. Muokkausten tekemisen jälkeen muutokset tallentuvat tietokantaan vasta, kun käyttäjä painaa "Tallenna muutokset" -nappia.

Alapalkin kautta pääsee "Varatut autot" -näkymään, jossa näytetään kaikki asiakkaiden varaamat autot. Menemällä tietyn auton tarkempiin tietoihin löytyy napit, joiden avulla auto voidaan vapauttaa takaisin saataville tai avata chat-näkymä, jossa yritys voi kommunikoida henkilöasiakkaan kanssa varaukseen liittyvistä asioista.

"Omat tiedot" -näkymässä yrityskäyttäjä voi tarkastella ja muokata omia tietojaan. Tietojen muokkaaminen onnistuu painamalla "Tietojen muokkaaminen" -nappia ja tallentamalla tehdyt muutokset. Samasta näkymästä on mahdollista kirjautua ulos.

Asetuksissa käyttäjä voi kirjautua ulos tai poistaa yrityskäyttäjätilinsä kokonaan. Uloskirjautuminen ja yritystilin poistaminen palauttaa sovelluksen samaan etusivu näkymään kuin henkilöasiakkaan. (KUVA 4.)



KUVA 4. Yritysasiakkaan vuokaavio

3 TEKNOLOGIAT

Sovelluksen kehittämisessä käytettiin teknologioita, jotka mahdollistavat tehokkaan ja sujuvan kehitysprosessin sekä varmistavat sovelluksen toimivuuden eri laitteilla. Valitut työkalut tukevat monialustakehitystä, tietoturvaa ja skaalautuvuutta, jotta sovellus toimii sekä henkilö- että yritysasiakkaille suunnitelluilla toiminnallisuuksilla.

Frontendin toteutuksessa hyödynnettiin teknologioita, joiden avulla sovellus saatiin toimimaan eri käyttöjärjestelmissä samalla toteutuksella. Kehitysympäristö valittiin niin, että sovelluksen rakentaminen ja testaaminen onnistuisi mahdollisimman vaivattomasti.

Backendissä käytettiin pilvipohjaista ratkaisua, joka mahdollistaa käyttäjätietojen hallinnan, tietokantapalvelut ja reaaliaikaisen tiedonsiirron ilman erillistä palvelinta. Lisäksi tietoturva varmistettiin käyttöoikeussääntöjen avulla, jotka rajoittavat pääsyä tietoihin käyttäjäroolien perusteella.

3.1 Sovelluksen frontend

Sovelluksen frontend kehitettiin käyttäen React Native -kirjastoa, koska sen avulla voidaan luoda mobiilisovellus sekä Androidille että iOS:lle yhdellä koodipohjalla. Tämä vähentää kehitystyöhön käytettävää aikaa ja yksinkertaistaa ylläpitoa verrattuna perinteisiin natiivikehitysratkaisuihin. React Native hyödyntää natiivikomponentteja, mikä takaa hyvän suorituskyvyn ja käyttökokemuksen.

Kehitysympäristöksi valittiin Expo, koska se tarjoaa kehittäjille valmiita työkaluja ja palveluita, jotka helpottavat sovelluksen rakentamista ja testaamista. Expon avulla sovellusta voi kehittää ja testata ilman tarvetta monimutkaisille natiiviasetuksille. Se mahdollistaa myös nopean käyttöönoton sekä automaattiset päivitykset ilman, että sovellusta tarvitsee julkaista uudelleen sovelluskaupoissa. Expon laajat integraatiomahdollisuudet, kuten WebView ja karttapalvelut, tekivät siitä houkuttelevan vaihtoehdon kehitysprosessiin. (Expo s.a. a; Expo 2024.)

Navigointi toteutettiin React Navigation -kirjastolla, joka mahdollisti eri näkymien väliset siirtymät käyttäen pinonavigointia (Stack Navigator). Käyttäjä pystyy siirtymään eri toimintoihin, kuten auton hakuun, varauksiin ja käyttäjäasetuksiin, sekä etenemään syvemmälle tiettyihin toimintoihin, kuten yksittäisen auton tietoihin tai omien varaustensa hallintaan. Tämä rakenne mahdollistaa selkeän ja loogisen navigoinnin ilman monimutkaisia valikkorakenteita. (React Native 2025d.)

Lisäksi React Nativen monipuolinen komponenttikirjasto sekä aktiivinen avoimen lähdekoodin kehittäjäyhteisö mahdollistivat tehokkaan sovelluskehityksen. Sen laajat sisäänrakennetut komponentit ja API:t, kuten FlatList, ScrollView ja TouchableOpacity, tarjoavat tehokkaita ratkaisuja käyttöliittymän luomiseen ja käyttäjäkokemuksen parantamiseen. Näiden avulla voidaan toteuttaa esimerkiksi suorituskykyisiä listanäkymiä, joustavaa skrollausta sekä responsiivisia käyttöliittymän elementtejä ilman ylimääräistä koodia. (React Native 2025b.)

Käyttöliittymätyylit määriteltiin erillisiin tyylitiedostoihin, kuten RegisterStyle.js ja HomeScreenStyle.js, jotta sovelluksen rakenne pysyi selkeänä ja helposti muokattavana. Lähes jokaisella näkymällä oli oma tyylitiedostonsa, mikä helpotti ulkoasun hallintaa ja kehitystyötä.

Tyylit toteutettiin React Nativen StyleSheet.create -metodilla, mikä paransi suorituskykyä ja teki koodista selkeämpää. Käyttöliittymän suunnittelussa hyödynnettiin joustavia asetteluratkaisuja, mikä varmisti sovelluksen toimivuuden eri näyttökokoisilla laitteilla. Painikkeisiin ja syötekenttiin lisättiin sopivat marginaalit ja pyöristetyt reunat, jotta käyttöliittymästä tuli miellyttävämpi käyttää. (React Native 2025e.)

Värimaailma pidettiin yhtenäisenä koko sovelluksessa käyttämällä teemavärejä. Tyylien erottaminen erillisiin tiedostoihin teki sovelluksen kehittämisestä tehokkaampaa, sillä muutoksia pystyttiin tekemään keskitetysti ilman, että niitä piti muokata jokaisessa komponentissa erikseen.

Frontend-teknologiat olivat jo ennestään tuttuja opinnäytetyön kirjoittajille kouluprojektien kautta, mikä nopeutti kehitysprosessia ja mahdollisti sujuvamman työskentelyn. React Native ja Expo valittiin myös niiden laajan yhteisötuen ja jatkuvan kehityksen vuoksi, mikä takaa hyvän tuen mahdollisille ongelmatilanteille. Kaikki

nämä ratkaisut mahdollistivat käyttäjäystävällisen ja teknisesti toimivan mobiilisovelluksen, jossa huomioitiin sekä henkilöasiakkaiden että yritysasiakkaiden tarpeet. React Native ja Expo tarjosivat joustavan ympäristön, jonka avulla kehitys voitiin toteuttaa tehokkaasti ja sovellus saatiin nopeasti käyttövalmiiksi.

3.2 Sovelluksen backend

Sovelluksen backend on rakennettu Firebase-alustan päälle, joka mahdollistaa pilvipohjaisen, skaalautuvan ja kustannustehokkaan ratkaisun ilman erillistä palvelinympäristöä. Firebasen valinta perustui ennen kaikkea sen helppokäyttöisyyteen ja tuttuuteen kehitystiimin keskuudessa. Lisäksi Firebase tarjoaa ilmaisen aloitustason, mikä teki siitä houkuttelevan vaihtoehdon sovelluksen kehityksen alkuvaiheessa.

Firebase tarjoaa laajan valikoiman palveluita, joista tässä projektissa keskeisimmässä roolissa ovat Firebase Authentication ja Firestore-tietokanta. Firebase Authentication vastaa käyttäjien tunnistautumisesta ja tilinhallinnasta, kun taas Firestore toimii tietokantana, jossa säilytetään kaikki sovelluksen keskeiset tiedot, kuten käyttäjäprofiilit, yritysasiakkaiden tiedot sekä chat-viestit. (Firebase 2025a; Firebase 2025e.)

3.2.1 Firebase Authentication – Käyttäjähallinta

Sovelluksen autentikointijärjestelmä on toteutettu Firebase Authentication -palvelun avulla, joka mahdollistaa käyttäjien rekisteröitymisen ja kirjautumisen sovellukseen sähköpostin ja salasanan avulla. Muihin autentikointimenetelmiin, kuten Google- tai puhelinnumerotunnistautumiseen, ei ole tässä vaiheessa siirrytty, sillä haluttiin keskittyä yksinkertaiseen ja turvalliseen kirjautumisratkaisuun. Firebase Authentication tarjoaa valmiit työkalut käyttäjätilien hallintaan, mukaan lukien rekisteröitymisen, sisäänkirjautumisen sekä salasanan palauttamisen, mikä helpottaa käyttäjähallintaa sovelluksen kehityksessä ja ylläpidossa. (Firebase 2025b.)

3.2.2 Firestore – Tietokanta

Sovelluksen tietojen tallentaminen ja hallinta on toteutettu Firestore-tietokannan avulla. Firestore on NoSQL-pohjainen pilvitietokanta, jossa data järjestetään kokoelmiin ja dokumentteihin. Tietomallin rakenne on suunniteltu siten, että se tukee sovelluksen keskeisiä toimintoja tehokkaasti ja mahdollistaa skaalautuvan sekä reaaliaikaisen tiedonkäsittelyn.

Sovelluksessa on kolme keskeistä tietokantakokoelmaa: käyttäjät, yritysasiakkaat ja chat-viestit. Käyttäjädata tallennetaan kahteen eri kokoelmaan käyttäjätyypin perusteella. Yksityisasiakkaiden tiedot säilytetään "users"-kokoelmassa, joka sisältää käyttäjän perustiedot, kuten etu- ja sukunimen, sähköpostiosoitteen, puhelinnumeron sekä tilin luontiajan. Yritysasiakkaiden tiedot tallennetaan erilliseen "companyusers"-kokoelmaan, joka sisältää samat tiedot kuin yksityisasiakkailla, mutta lisäksi myös yrityksen nimen. Tämä erottelu mahdollistaa selkeämmän käyttäjäroolien hallinnan ja tukee sovelluksen toiminnallisuuksia yritys- ja henkilöasiakkaille.

Chat-toiminto on toteutettu Firestoreen perustuen, jossa jokainen viesti tallennetaan "chats"-kokoelmaan. Jokainen viesti sisältää viestin lähettäjän tunnisteen, tekstisisällön ja aikaleiman, mikä mahdollistaa viestihistorian tarkan seurannan. Firestoren reaaliaikainen synkronointi takaa, että viestit päivittyvät sovelluksessa välittömästi ilman manuaalista päivittämistä. Näin ollen käyttäjät voivat kommunikoida sujuvasti ja ilman viiveitä.

Firestore-tietokannassa on myös alakokoelmia, jotka laajentavat sovelluksen toiminnallisuuksia. Yksityisasiakkaiden puolella "users"-kokoelman alla sijaitsee alakokoelma "varaukset", jossa näkyvät asiakkaan tekemät varaukset. Tämä mahdollistaa yksittäisen käyttäjän varaushistorian seurannan ja helpottaa palveluiden hallintaa. Yritysasiakkaiden osalta "companyusers"-kokoelman alla on kaksi alakokoelmaa: "varatutautot" ja "lisätyt autot". "Varatutautot"-alakokoelma sisältää tiedot asiakkaiden varaamista ajoneuvoista, kun taas "lisätyt autot" pitää kirjaa yrityksen hallinnoimista ja vuokrattavissa olevista ajoneuvoista. Näiden alakokoelmien avulla Firestore mahdollistaa hierarkkisen tiedonhallinnan, mikä

tehostaa tietokantakyselyitä ja mahdollistaa suorituskykyisen datan käsittelyn. (Firebase 2025b).

3.2.3 Firestore Security Rules – Käyttöoikeuksien hallinta

Firestore tarjoaa myös mahdollisuuden määrittää säännöt, joiden avulla voidaan hallita tietokannan käyttöoikeuksia. Tässä projektissa käytetään seuraavia Firestore-sääntöjä. (KUVA 5.) Näiden sääntöjen avulla varmistetaan, että tietokanta on turvallinen ja käyttäjäkohtaiset tiedot pysyvät suojattuina. (Firebase 2025c.)

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    //  Henkilöasiakas voi lukea ja muokata vain omia tietojaan
    match /users/{userId} {
      allow read, write, delete: if request.auth != null && request.auth.uid == userId;

      //  Henkilöasiakkaan varaukset (henkilö voi poistaa itse, mutta myös yritys voi poistaa)
      match /varaukset/{varausId} {
        allow read, write, delete: if request.auth != null && request.auth.uid == userId;

        //  Yritysasiakas voi poistaa varauksen, jos auto kuuluu yritykselle
        allow delete: if request.auth != null && resource.data.companyId == request.auth.uid;
      }
    }

    //  Yritysasiakkaat voivat hallita omia tietojaan
    match /companyusers/{userId} {
      allow read, write, delete: if request.auth != null && request.auth.uid == userId;

      //  **Kaikki voivat nähdä vapaana olevat autot**
      match /autot/{autoId} {
        allow read: if true;

        //  Yritys voi hallita vain omia autojaan (muokata ja poistaa)
        allow write, delete: if request.auth != null && request.auth.uid == userId;

        //  Henkilöasiakas voi poistaa auton vapaista autoista (eli varata sen)
        allow delete: if request.auth != null;

        //  Henkilöasiakas voi lisätä auton takaisin vapaiden autojen listalle, jos se oli varauksessa
        allow create: if request.auth != null;
      }

      //  Varatut autot (yritys ja henkilöasiakas voivat molemmat poistaa varauksia)
      match /VaratutAutot/{autoId} {
        allow read: if request.auth != null && request.auth.uid == userId;

        //  Yritysasiakas voi poistaa varauksen, jos auto kuuluu yritykselle
        allow delete: if request.auth != null && resource.data.companyId == request.auth.uid;

        //  Henkilöasiakas voi poistaa oman varauksensa yrityksen varatuista autoista
        allow delete: if request.auth != null && resource.data.userId == request.auth.uid;

        //  Henkilöasiakas voi lisätä tänne varatun auton (eli tehdä varauksen)
        allow create: if request.auth != null;
      }
    }

    //  Yritysasiakas voi poistaa henkilöasiakkaan varauksen
    match /users/{userId}/varaukset/{varausId} {
      allow delete: if request.auth != null && resource.data.companyId == request.auth.uid;
    }

    //  Henkilöasiakas voi poistaa yrityksen varatun auton, jos hän on varaaja
    match /companyusers/{companyId}/VaratutAutot/{autoId} {
      allow delete: if request.auth != null;
    }

    //  **Kaikkien käyttäjien julkinen lukuoikeus**
    match /users/{document=**} {
      allow read: if true;
    }

    match /companyusers/{document=**} {
      allow read: if true;
    }

    //  Chatin viestit (käyttäjät voivat vain lisätä omia viestejään)
    match /chats/{chatId}/messages/{messageId} {
      allow read: if request.auth != null;
      allow create: if request.auth != null;
      allow delete: if request.auth != null;
    }

    //  **Chatin päädokumentti (voi poistaa kuka tahansa kirjautunut käyttäjä)**
    match /chats/{chatId} {
      allow delete;
    }
  }
}
```

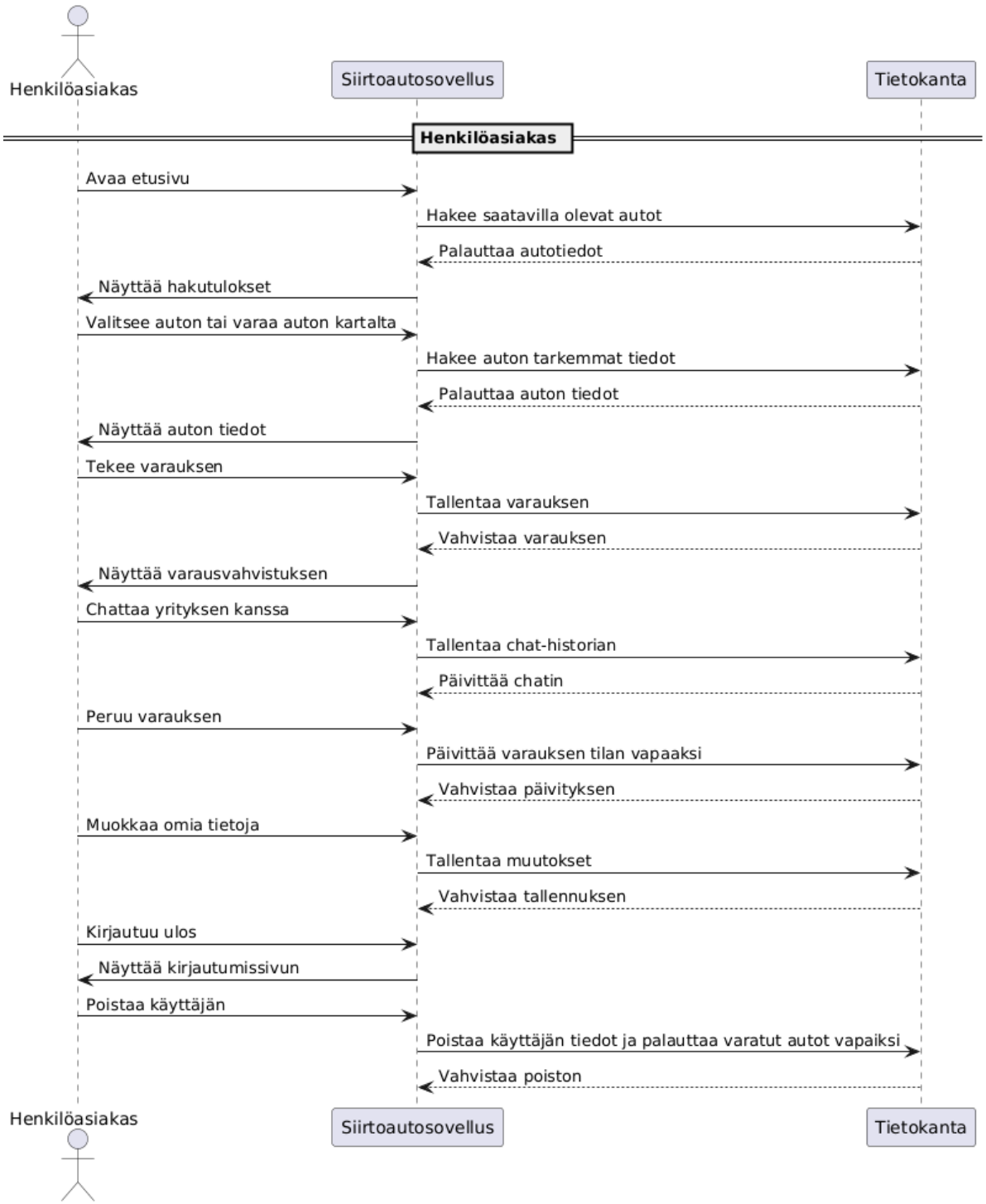
KUVA 5. Firestore säännöt

3.3 Sekvenssikaaviot

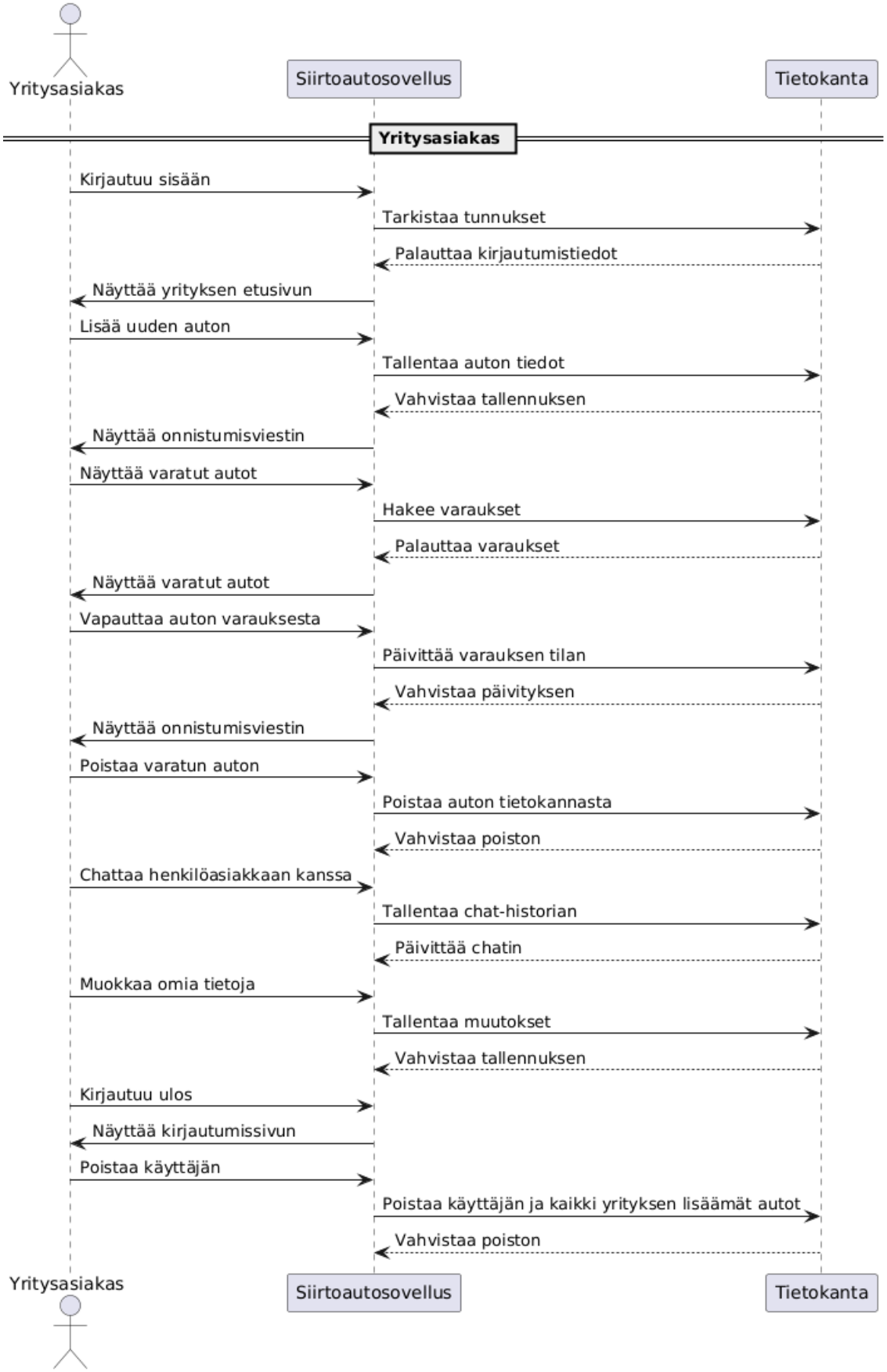
Sekvenssikaavioiden avulla havainnollistetaan sovelluksen eri komponenttien välistä vuorovaikutusta ja tiedonsiirtoa. Tässä projektissa sekvenssikaavioita käytettiin erityisesti henkilö- ja yritysasiakkaiden toimintojen kuvaamiseen. Kaavioiden avulla pystyttiin jäsentämään teknisiä ratkaisuja, kuten varauksen tallennusta Firestore-tietokantaan ja käyttäjän tunnistamista Firebase Authenticationin avulla.

KUVA 6. Kaavio kuvaa, miten henkilöasiakas selaa autoja, tarkastelee tietoja ja tekee varauksen. Sovellus hakee ja tallentaa tiedot Firebasen avulla, ja käyttäjälle esitetään lopuksi varausvahvistus.

KUVA 7. Sekvenssikaavio kuvaa yritysasiakkaan toimintaa, kuten sisäänkirjautumista, uuden auton lisäämistä ja varattujen autojen tarkastelua. Toiminnot tallennetaan Firestoreen, ja jokaisen vaiheen jälkeen käyttäjälle näytetään vahvistus onnistuneesta suorituksesta.



KUVA 6. Henkilöasiakkaan sekvenssikaavio



KUVA 7. Yritysasiakkaan sekvenssikaavio

4 SOVELLUKSEN TOIMINNALLISUUDET

Sovelluksen toiminnallisuudet on suunniteltu tarjoamaan käyttäjille kokonaisvaltainen ja helppokäyttöinen kokemus siirtoautojen etsimiseen, varaamiseen ja hallintaan. Toiminnallisuudet on tehty käyttäjäryhmien tarpeiden mukaan, ja niiden tavoitteena on yhdistää tekninen tehokkuus ja käyttäjäystävällisyys. Keskeisiä toiminnallisuuksia ovat rekisteröityminen ja kirjautuminen, autojen haku, varaaminen, lisääminen, karttaominaisuudet sekä chat.

Toiminnallisuudet tukevat sekä henkilöasiakkaiden että yritysasiakkaiden tarpeita. Henkilöasiakkaat voivat käyttää sovellusta löytääkseen ja varatakseen itselleen sopivia autoja, kun taas yritysasiakkaille on tarjolla työkaluja ajoneuvojen lisäämiseen ja hallintaan. Tämä lähestymistapa mahdollistaa sovelluksen monipuolisen käytön eri käyttäjäryhmissä. Sovellus sisältää monipuolisia ominaisuuksia, jotka helpottavat käyttäjien päätöksentekoa ja nopeuttavat palveluiden käyttöä. Lisäksi sovellus mahdollistaa toiminnallisuuksien laajentamisen, mikä tukee tulevaisuuden kehitystarpeita ja sovelluksen pitkäaikaista käyttöä.

4.1 Rekisteröityminen ja kirjautuminen

Rekisteröityminen ja kirjautuminen ovat keskeisiä toiminnallisuuksia, jotka mahdollistavat sovelluksen käytön ja varaustoimintojen hyödyntämisen. Käyttäjä aloittaa rekisteröitymisen syöttämällä perustiedot, kuten etunimen, sukunimen, sähköpostiosoitteen ja salasanan. Yritysasiakas lisäksi syöttää yrityksen nimen. Nämä tiedot tallennetaan sovelluksen tietokantaan, ja ne mahdollistavat käyttäjän yksilöimisen sekä tilin hallinnoinnin. Rekisteröitymisen jälkeen käyttäjä voi kirjautua sisään samoilla tunnistetiedoilla, mikä mahdollistaa palvelun laajemman käytön.

Kirjautumisprosessi on suunniteltu nopeaksi ja sujuvaksi, jotta käyttäjä pääsee helposti käsiksi sovelluksen eri toiminnallisuuksiin. Kirjautumisen avulla käyttäjä voi esimerkiksi tehdä varauksia ja tarkastella omia tietojaan. Mikäli käyttäjä ei ole

vielä rekisteröitynyt, tulee rekisteröityä, jossa tarvittavat tiedot täytetään ennen ensimmäistä kirjautumista.

Sovellus tukee selaamista ilman kirjautumista tai rekisteröitymistä, jolloin käyttäjä voi katsella tarjolla olevia siirtoautoja, mutta ei voi tehdä varauksia. Tämä mahdollistaa palvelun toiminnallisuuksien alustavan tutustumisen ilman sitoutumista, mikä voi madaltaa kynnystä rekisteröitymiseen.

4.1.1 Henkilöasiakkaiden rekisteröinti

Kirjautuminen ja rekisteröityminen mahdollistavat käyttäjän pääsyn sovelluksen eri palveluihin. Rekisteröitymisen yhteydessä käyttäjä syöttää perustiedot, kuten etunimen, sukunimen, sähköpostiosoitteen ja salasanan. Tiedot tallennetaan järjestelmään, minkä jälkeen käyttäjä voi kirjautua sisään ja hyödyntää sovellusta.

Kirjautuminen tapahtuu sähköpostiosoitteella ja salasanalla. Jos tiedot ovat virheellisiä, sovellus ilmoittaa asiasta ja pyytää yrittämään uudelleen.

Kirjautumisnäkymä on selkeä ja helposti navigoitava. Käyttäjä voi siirtyä rekisteröitymissivulle Kirjautuminen-kohdasta. Sovellus mahdollistaa selaamisen ilman kirjautumista, mutta varauksen tekeminen edellyttää tilin luomista.

4.1.2 Yritysasiakkaiden rekisteröinti

Yritysasiakkaiden rekisteröityminen ja kirjautuminen tarjoavat pääsyn palvelun yrityskohtaisiin toimintoihin. Rekisteröityessä yritysasiakas syöttää perustiedot, kuten etunimen, sukunimen, yrityksen nimen, oman sähköpostiosoitteensa ja salasanan. Tiedot tallennetaan järjestelmään, minkä jälkeen yritysasiakas voi kirjautua sisään ja hallinnoida palvelussa olevia tietojaan. Kirjautuminen tapahtuu sähköpostiosoitteella ja salasanalla. Jos tiedot ovat virheellisiä, sovellus ilmoittaa asiasta ja pyytää yrittämään uudelleen.

Kirjautumisnäkymä on selkeä ja helposti navigoitava. Käyttäjä voi siirtyä rekisteröitymissivulle Kirjautuminen-kohdasta. Yritysasiakkaat voivat kirjautumisen jälkeen lisätä ja hallinnoida siirtoautoilmoituksiaan.

4.1.3 Käyttäjätietojen tietojen poistaminen

Sovellukseen toteutettiin ominaisuus, joka mahdollistaa sekä henkilöasiakkaan että yritysasiakkaan tilin pysyvän poistamisen. Poistamisen yhteydessä kaikki käyttäjän tiedot poistetaan Firestore-tietokannasta sekä Firebase Authentication -palvelusta. Lisäksi poistamisen yhteydessä käsitellään mahdolliset varaukset ja varmistetaan, että ne eivät jää aktiivisiksi järjestelmään. Mikäli henkilöasiakas on varannut auton, varaus poistetaan ja auto palautetaan yrityksen vapaiden autojen joukkoon. Yritysasiakkaan kohdalla varmistetaan, että hänen lisäämänsä autot ja mahdolliset varatut autot käsitellään asianmukaisesti ennen tilin poistamista.

Ominaisuuden kehittämisessä oli tärkeää varmistaa, että tiedot poistuvat oikein ja ettei järjestelmään jää keskeneräisiä varauksia tai käyttökelvottomia tietoja. Näin varmistetaan, että poistotoiminto ei aiheuta tietokantaan virhetilanteita ja että käyttöliittymä pysyy toimivana.

Poistamisen toteutus jaettiin kahteen erilliseen tiedostoon, joista Settings.js vastaa henkilöasiakkaan poistamisesta ja CompanySettings.js yritysasiakkaan poistamisesta. Molemmat poistotoiminnot seuraavat samanlaista perusrakennetta, jossa käyttäjälle näytetään ensin vahvistusikkuna ennen tietojen lopullista poistamista.

Henkilöasiakkaan poistaminen alkaa sillä, että sovellus tarkistaa, onko käyttäjä kirjautunut sisään. Tämän jälkeen haetaan käyttäjän tekemät varaukset Firestore-tietokannasta ja poistetaan ne yksi kerrallaan. Samalla tarkistetaan, että käyttäjän varaamat autot palautetaan yritysasiakkaan listoille, jotta ne voidaan varata uudelleen. Kun kaikki varaukset on käsitelty, käyttäjän tunniste poistetaan Firestoresta, minkä jälkeen käyttäjä kirjataan ulos ja hänen Firebase Authentication -tilinsä poistetaan. Tämä varmistaa, että käyttäjän tiedot eivät jää mihinkään järjestelmän osaan.

Yritysasiakkaan poistaminen on monimutkaisempi prosessi, koska yrityksellä voi olla hallussaan useita autoja ja varattuja autoja. Poistamisen yhteydessä haetaan yrityksen lisäämät autot Firestoresta ja poistetaan ne. Jos jokin auto on varattuna, se täytyy ensin vapauttaa ennen kuin se voidaan poistaa tietokannasta. Samalla

poistetaan myös yritykseen liittyvät chat-viestit, jotta ne eivät jää näkyviin käyttöliittymään. Kun kaikki yritykseen liittyvät tiedot on käsitelty, yrityksen tunniste poistetaan Firestoresta, minkä jälkeen käyttäjä kirjataan ulos ja Firebase Authentication -tili poistetaan.

Teknisesti poistaminen toteutettiin Firestore-tietokannan metodeilla deleteDoc, setDoc ja getDocs, joiden avulla voidaan hakea, päivittää ja poistaa tietoja tehokkaasti. Näiden lisäksi käytettiin Firebase Authentication -palvelun deleteUser-metodia, jolla käyttäjän tili poistetaan lopullisesti. Käyttöliittymässä varmistettiin, että käyttäjä saa visuaalisen vahvistuksen tilin poistamisesta, ja virhetilanteissa käyttäjälle annetaan selkeä ilmoitus.

Käyttäjän tietojen poistaminen on tärkeä osa sovelluksen toimintaa, sillä jokaisella käyttäjällä täytyy olla mahdollisuus poistaa omat tietonsa, mikäli hän ei halua enää käyttää palvelua. Poistomekanismi on välttämätön tietosuojasyistä ja vastaa yleisiä käytäntöjä, joita sovelletaan verkkopalveluiden käyttäjätietojen hallintaan.

Poistamisen yhteydessä oli tärkeää varmistaa, että järjestelmään ei jää epätäydellisiä tietoja tai käyttökelvottomia varauksia. Jos esimerkiksi henkilöasiakas poistaisi tilinsä ilman, että hänen varauksensa käsiteltäisiin asianmukaisesti, kyseiset autot voisivat jäädä näkymään varattuina ilman mahdollisuutta vapauttaa niitä. Tämä voisi aiheuttaa ongelmia yritysasiakkaiden näkökulmasta, sillä he eivät välttämättä tietäisi, että auto on itse asiassa vapautunut. Samoin yritysasiakkaan kohdalla oli tärkeää varmistaa, että kaikki hänen lisäämänsä autot ja varatut autot poistuvat tietokannasta, jotta järjestelmä pysyy eheänä.

Tekninen toteutus valittiin niin, että kaikki käyttäjän tiedot poistuvat yhdellä kertaa, eikä käyttäjän tarvitse erikseen poistaa varauksiaan ennen tilin poistamista. Tämä vähentää käyttäjän työmäärää ja tekee prosessista sujuvamman. Firestore-tietokannan käyttö mahdollisti tehokkaan tavan käsitellä tietojen poistoa, ja Firebase Authenticationin tarjoama deleteUser-metodi varmisti, että käyttäjän tili poistuu lopullisesti järjestelmästä. (Firebase 2025d.)

4.2 Autojen haku

Autojen hakutoiminto on yksi sovelluksen keskeisimmistä ominaisuuksista, sillä se tarjoaa käyttäjille mahdollisuuden löytää juuri heidän tarpeisiinsa sopivia siirtoautoja nopeasti ja vaivattomasti. Tämä toiminto on suunniteltu vastaamaan monipuolisesti käyttäjä tarpeisiin ja mahdollistamaan autojen etsimisen joustavasti erilaisilla kriteereillä. Hakutoiminnon helppokäyttöisyys on olennaista, jotta varaaminen ja autojen selaaminen tapahtuu mahdollisimman vaivattomasti.

4.2.1 Hakuprosessin vaiheet

Käyttäjä voi määrittää haluamansa hakuehdot tai vaihtoehtoisesti tarkastella kaikkia saatavilla olevia autoja ilman rajoituksia jättämällä kaikki kentät tyhjiksi. Sovellus hyödyntää hakuprosessissa erillisiä modaalikomponentteja, joiden avulla käyttäjä voi tehdä valinnat ilman, että tietoja tarvitsee kirjoittaa itse kokonaan manuaalisesti. Noutopaikka ja määränpää voidaan valita modaalin kautta, joka ehdottaa sijainteja käyttäjän kirjoittaessa. Tämä vähentää kirjoitusvirheitä ja nopeuttaa hakuprosessia. Vaihteiston ja istumapaikkojen valinta toimii samalla periaatteella, mutta tarjoaa käyttäjälle valmiit vaihtoehdot, joiden välillä voi tehdä valinnan yhdellä napautuksella. Kun käyttäjä on tehnyt haluamansa valinnat, sovellus ohjaa hakutulossivulle, jossa vapaana olevat autot, jotka vastaavat hakukriteerejä näytetään.

4.2.2 Hakutuloksien esitystapa

Hakutulokset esitetään käyttäjälle selkeässä ja visuaalisesti miellyttävässä listamuodossa, jossa jokainen auto on yksilöity tärkeimmillä tiedoilla. Näitä tietoja ovat esimerkiksi auton sijainti, vaihteiston tyyppi, istumapaikkojen määrä. Listan avulla käyttäjä voi vertailla eri vaihtoehtoja keskenään ja tehdä päätöksen nopeasti ja vaivattomasti.

Listasta käyttäjä voi valita minkä tahansa auton tarkempia tietoja varten. Tämän näkymän kautta käyttäjä voi myös siirtyä suoraan varaussivulle, mikä nopeuttaa

koko prosessia. Hakutoiminto on suunniteltu siten, että se toimii saumattomasti sekä uusille että kokeneille käyttäjille. Helppokäyttöisyys ja selkeä käyttöliittymä takaavat, että auton etsiminen ja varaaminen onnistuu mahdollisimman sujuvasti.

4.3 Auton varaaminen

Auton varaus tapahtuu sen jälkeen, kun käyttäjä on valinnut itselleen sopivan auton hakutuloksista. Varaamisen ehtona on, että käyttäjä on kirjautunut sovellukseen. Mikäli käyttäjä ei ole kirjautunut, hänet ohjataan kirjautumissivulle, ja jos ei ole rekisteröitynyt, niin voi valita ”Rekisteröidy” kirjautumissivulta, jonka jälkeen voi jatkaa varaamista. Auton tiedot voivat sisältää esimerkiksi noutopaikan, luovutuspaikan ja muuta olennaista tietoa auton saatavuudesta.

Kun auto on varattu, varauksen tiedot tallennetaan Firestore-tietokantaan, ja ne liitetään käyttäjän henkilökohtaisiin tietoihin. Käyttäjä voi tarkastella omia varauksiaan sovelluksen "Omat tiedot" -välilehdeltä. Tämä välilehti tarjoaa myös mahdollisuuden peruuttaa varaus, mikäli käyttäjä ei enää tarvitse autoa. Varauksen peruuttaminen tapahtuu helposti "Peru varaus" -painikkeen kautta, jolloin auto poistetaan varattujen autojen luettelosta ja vapautetaan muiden käyttäjien käyttöön.

Sovelluksessa on otettu huomioon myös käyttäjäturvallisuus, sillä varauksen tekeminen ja varauksien katselu on rajoitettu vain kirjautuneille käyttäjille. Tämä varmistaa, että vain valtuutetut käyttäjät voivat tehdä varauksia ja että heidän tietonsa pysyvät suojattuina. Käyttäjän henkilötiedot, kuten nimi, puhelinnumero ja sähköpostiosoite, tallennetaan tietokantaan, mutta niitä käsitellään ja suojataan Firestore-tietokannan sääntöjen mukaisesti.

Auton varaaminen on mahdollista myös suoraan kartalta, jossa käyttäjä voi valita itselleen sopivan auton sen sijainnin perusteella. Kartta esittää autot visuaalisesti ja mahdollistaa varauksen tekemisen suoraan valitusta autosta, mikä tekee varaamisesta entistä helpompaa. Lisätietoja kartalta varaamisesta löytyy alaluvusta 4.4.3.

4.4 Karttaominaisuudet

Karttaominaisuudet ovat olennainen osa sovellusta, sillä ne tarjoavat käyttäjille visuaalisen ja tehokkaan tavan tarkastella autojen sijainteja. Autojen löytäminen kartalta on käyttäjäystävällisempi vaihtoehto pelkkään tekstimuotoiseen listaan verrattuna, sillä se antaa selkeän kokonaiskuvan saatavilla olevista autoista ja niiden sijainneista. Karttanäkymä mahdollistaa autojen nopean paikantamisen ja helpottaa päätöksentekoa.

Sovellus käyttää Mapbox-karttapalvelua, joka on sovelluksessa React Native WebView -komponentin avulla. WebView mahdollistaa helpon ja sujuvan kartta-kokemuksen verkkoteknologioiden avulla. Autojen sijainnit haetaan Firestore-tie-tokannasta, jossa jokaisella yrityksellä on oma kokoelmansa. Kokoelmasta löytyy yrityksen lisätyt autot ja niiden tiedot. Kun käyttäjä avaa karttanäkymän, sovellus hakee kaikkien yritysten autot, noutaa niiden osoitetiedot ja muuntaa ne Mapbox Geocoding API:n avulla tarkaksi sijainniksi ja näyttää ne kartalla.

Kartalla jokainen auto näkyy punaisella markkerilla, ja markkeria klikkaamalla käyttäjä saa näkyviin auton lisätiedot. Näihin tietoihin sisältyvät muun muassa auton noutopaikka, määränpää, vaihteiston tyyppi ja istumapaikkojen määrä. Kartalta auton valinnut käyttäjä voi siirtyä varaamaan sen ilman, että hänen tar-vitsee palata erilliseen hakunäkymään.

4.4.1 Lähimpien autojen näyttäminen

Kartassa on ominaisuus, joka on suunniteltu tukemaan myös käyttäjän sijaintiin perustuvaa navigointia. Sovellus hyödyntää Expo Location -kirjastoa, jonka avulla käyttäjän sijainti voidaan hakea ja näyttää kartalla sinisellä markkerilla (Expo s.a. b.). Jos käyttäjä antaa sovellukselle luvan käyttää sijaintiaan, hänen nykyinen sijaintinsa näytetään kartalla. Kuitenkin, jos käyttäjä ei anna sovellukselle lupaa käyttää sijaintitietoja, sovellus käyttää oletussijaintina Helsinkiä. Tämä

varmistaa, että sovellus voi toimia normaalisti myös ilman käyttäjän tarkkaa sijaintitietoa, mutta ei tarjoa käyttäjälle täyttä kokemusta.

Sijaintitietoa hyödynnetään lähimpien autojen löytämisessä ja niiden vertailussa. Käyttäjän ei tarvitse selata listamuotoisia hakutuloksia koska karttanäkymä auttaa nopeasti paikantamaan lähimmät autot ja tekee varaamisesta nopeampaa. Autojen noutopaikat näytetään kartalla punaisina markkereina, joiden avulla käyttäjä voi helposti hahmottaa eri vaihtoehdot ja arvioida, mikä niistä sopii hänen tarpeisiinsa parhaiten.

4.4.2 Reitin näyttäminen valittuun autoon

Sovelluksessa on toteutettu toiminto, jonka avulla käyttäjä voi näyttää reitin valitsemalleen autolle. Tämä toiminnallisuus hyödyntää Mapbox Directions API:a, joka näyttää käyttäjän nykyisen sijainnin ja valitun auton sijainnin välisen reitin. Reitti piirretään kartalle punaisella viivalla, jolloin käyttäjä näkee suoraan, kuinka pääsee autolle.

Kun käyttäjä näyttää reitin kartalla, näytölle ilmestyy arvioitu matka kilometreinä sekä arvioitu ajoaika minuutteina. Tämä tekee auton varaamisesta entistä sujuvampaa, sillä käyttäjä saa jo ennen varausta käsityksen siitä, kuinka kaukana auto sijaitsee ja kuinka kauan sen saavuttamiseen menee.

4.4.3 Auton varaaminen suoraan kartalta

Karttanäkymässä on myös toiminnallisuus, joka mahdollistaa myös autojen varaamisen suoraan markkerista. Käyttäjä voi valita kartalta haluamansa auton ja siirtyä suoraan varaussivulle ilman, että hänen tarvitsee etsiä samaa autoa uudelleen listahaun kautta.

Varausprosessi toimii siten, että kun käyttäjä painaa markkerin avaamassa pop-up-ikkunassa "Varaa auto" -painiketta, kartta lähettää WebView-komponentin kautta viestin React Native -sovellukselle. Tämä viesti sisältää valitun auton tiedot, kuten sen ID:n, sijainnin ja muut tärkeät ominaisuudet. React Native -sovellus

kuuntelee viestejä WebView-komponentin onMessage-ominaisuudella ja ohjaa käyttäjän AutonTiedot-sivulle, jossa hän voi viimeistellä varauksen. Tämä ratkaisu parantaa varaustoiminnon joustavuutta ja nopeutta, sillä käyttäjän ei tarvitse tehdä ylimääräisiä vaiheita varausprosessissa. Auton voi varata heti sen löydyttyä.

4.4.4 Tekninen toteutus

Karttaominaisuuksien toteutus yhdistää useita teknologioita ja rajapintoja, joiden avulla autojen tiedot voidaan hakea ja esittää visuaalisesti käyttäjälle. Firestore-tietokanta toimii autojen tietojen säilytyspaikkana, ja jokaisen yrityksen autot haetaan erikseen tietokannasta. Jokaisen auton osoite muunnetaan Mapbox Geocoding API:n avulla koordinaateiksi, jotta ne voidaan lisätä kartalle. (Mapbox s.a.)

Sovellus toteuttaa WebView-kartan, jonka avulla voidaan kommunikoida React Native -sovelluksen kanssa reaaliajassa. WebView-komponentti vastaanottaa käyttäjän toiminnot, kuten varaamisen ja reitin näyttämisen, ja ohjaa käyttäjän oikeisiin näkymiin. (Expo 2024.)

Sovellus hyödyntää lisäksi Firestore-sääntöjä, joilla varmistetaan, että vain oikeutetut käyttäjät voivat varata autoja. Tietokantahakujen optimointi mahdollistaa nopean tiedonsiirron, mikä tekee karttanäkymästä sujuvan käyttää. (Firebase 2025d.)

4.4.5 Karttaominaisuuksien merkitys käyttäjäkokemukselle

Karttaominaisuudet tekevät sovelluksesta huomattavasti käyttäjäystävällisemmän ja tehokkaamman. Käyttäjä voi etsiä autoja suoraan kartalta, vertailla vaihtoehtoja visuaalisesti ja valita sopivan auton ilman monimutkaisia hakuprosesseja. Reittitoiminnallisuus tukee käyttäjän päätöksentekoa ja varmistaa, että auton löytämiseen ei kulu turhaa aikaa.

Auton varaaminen suoraan kartalta on yksi sovelluksen keskeisimmistä ominaisuuksista, sillä se tekee varaamisesta nopeaa ja helppoa. Kartan visuaalinen esitystapa vähentää manuaalista tiedonhakua ja tekee sovelluksesta tehokkaan työkalun henkilöasiakkaille. Karttanäkymän avulla käyttäjä voi hahmottaa tilanteensa reaaliajassa ja tehdä päätöksiä nopeasti.

4.5 Autojen lisääminen ja hallinta

Sovellus tarjoaa yritysasiakkaille käytännölliset työkalut autojen lisäämiseen ja hallintaan. Autojen hallinta on suunniteltu mahdollisimman suoraviivaiseksi, jotta yritykset voivat lisätä ajoneuvojaan ilman turhia monimutkaisuuksia. Samalla varmistetaan, että tiedot pysyvät ajan tasalla ja yrityskäyttäjät voivat helposti hallita lisättyjä autoja. Jokainen lisätty auto näkyy heti hakutuloksissa ja karttanäkymässä, jolloin henkilöasiakkaat voivat varata sen.

4.5.1 Autojen lisääminen

Yritysasiakkaan lisätessä auton sovellukseen hän täyttää yksinkertaisen lomakkeen, johon syötetään auton keskeiset tiedot. Käyttökokemuksen parantamiseksi lomake hyödyntää valintamodaaleita, jotka auttavat käyttäjää syöttämään tietoja nopeammin ja tarkemmin. Käyttäjän kirjoittaessa tietoja modaalit ehdottavat automaattisesti sopivia vaihtoehtoja, kuten osoitteita ja kaupunkeja. Tämä vähentää virheiden mahdollisuutta ja tekee oikean tiedon löytämisestä helpompaa ilman, että käyttäjän tarvitsee syöttää kaikkea käsin. Lomakkeen suunnittelu painottaa helppokäyttöisyyttä ja virheiden minimointia, mikä nopeuttaa autojen lisäysprosessia ja takaa tiedon laadun. Tämä mahdollistaa sen, että ajoneuvot ovat nopeasti saatavilla henkilöasiakkaiden hakuja varten.

Kun kaikki tiedot on täytetty, auto tallennetaan Firestore-tietokantaan. Ennen tallennusta järjestelmä tarkista, että kaikki kentät on täytetty oikein, eikä autoa voi lisätä puutteellisilla tiedoilla. Käyttäjälle näytetään onnistumisviesti, kun lisäys on suoritettu.

4.5.2 Ilmoitusten hallinta

Yritysasiakkaiden käytössä on erillinen hallintanäkymä, josta he voivat tarkastella, muokata ja poistaa aiemmin lisäämiään autoja. Tämä näkymä tarjoaa listan yrityksen autoista ja antaa mahdollisuuden päivittää auton tietoja suoraan sovelluksessa. Näkymän avulla varmistetaan, että kaikki ilmoitukset pysyvät ajan tasalla. Kun käyttäjä muokkaa auton tietoja, muutokset tallennetaan Firestore-tietokantaan, ja onnistuneesta päivityksestä annetaan vahvistusilmoitus.

Kun ilmoitus halutaan poistaa, yritysasiakas voi poistaa sen hallintanäkymän kautta. Autoa poistaessa sovellus avaa erillisen varmistusikkunan, ettei poistoa suoriteta vahingossa. Vahvistuksen jälkeen auto poistetaan tietokannasta, jolloin se katoaa sekä hakutuloksista että karttanäkymästä. Jos poistaminen onnistui käyttäjä saa vahvistusilmoituksen, ja hallintanäkymä päivittyy automaattisesti.

Autojen lisäämisen, hallinnan ja poistamisen tarkoituksena on tehdä yritysasiakkaiden työstä mahdollisimman sujuvaa ja tehokasta. Sovelluksella pyritään varmistamaan, että kaikki tiedot ovat helposti hallittavissa ja jokainen muutos näkyy palvelussa, mikä parantaa käyttäjäkokemusta ja takaa järjestelmän luotettavuuden kaikille osapuolille.

4.6 Chat-ominaisuus

Sovellukseen toteutettiin chat-toiminnallisuus, joka mahdollistaa henkilöasiakkaan ja yritysasiakkaan välisen reaaliaikaisen viestinnän varauksen yhteydessä. Chat parantaa asiakaskokemusta tarjoamalla suoran yhteyden palveluntarjoajaan, mikä vähentää epäselvyyksiä ja tekee varausten käsittelystä tehokkaampaa.

Chat-sovellus toteutettiin hyödyntämällä React Native -kehystä käyttöliittymän rakentamiseen sekä Firebase Firestore -tietokantaa viestien tallentamiseen ja synkronoimiseen. Firestore mahdollistaa viestien reaaliaikaisen synkronoinnin käyttäjien välillä ilman ylimääräisiä päivityskomentoja. Chatin käyttöliittymässä käytettiin FlatList-komponenttia, joka mahdollistaa tehokkaan viestien latauksen

ja selaamisen. Viestit järjestetään aikajärjestyksessä ja ne päivittyvät välittömästi Firestore-tietokannasta reaaliaikaisten kuuntelujen avulla. (Gehani, V. 2022; React Native. 2025b.)

4.6.1 Käytetyt teknlogiat

Chatin toteutuksessa käytettiin useita React Native -komponentteja ja Firebase Firestore -palveluja, joista keskeisimmät ovat useState, useEffect, FlatList, Firestore, onSnapshot ja addDoc.

Käyttäjän syöttämä viesti tallennetaan useState-hookiin, joka säilyttää viestikentän sisällön ennen sen tallentamista tietokantaan. useEffect-hook puolestaan mahdollistaa sen, että sovellus kuuntelee Firestore-tietokantaa ja päivittää käyttöliittymän reaaliaikaisesti uusien viestien tullessa. Tämä toteutettiin käyttämällä onSnapshot-funktiota, joka mahdollistaa Firestore-dokumenttien automaattisen kuuntelemisen ilman manuaalista päivittämistä.

Chatin käyttöliittymä toteutettiin hyödyntämällä FlatList-komponenttia, joka on optimoitu näyttämään suuria määriä tietoa ilman suorituskykyongelmia. Tämä mahdollistaa sen, että vanhemmat viestit eivät kuormita sovelluksen muistia, vaan niitä ladataan tarpeen mukaan vieritettäessä. KeyboardAvoidingView-komponenttia käytettiin estämään tekstikentän peittyminen, kun käyttäjä kirjoittaa viestin, ja TouchableWithoutFeedback-komponentilla mahdollistettiin näppäimistön sulkeminen, kun käyttäjä painaa ruutua.

Viestit tallennetaan Firestore-tietokantaan, joka on NoSQL-pohjainen pilvitietokanta. Tietokanta tukee reaaliaikaista synkronointia ja mahdollistaa viestien automaattisen päivittymisen kaikille keskustelun osapuolille ilman erillisiä API-pyyntöjä. Tämä tekee Firestoresta optimaalisen valinnan chat-sovellukseen.

Jokaiselle varaukselle luodaan oma chattihuone, joka tallennetaan Firestoreen polkuun chats/{chatId}/messages/. Tämä mahdollistaa sen, että vain keskusteluun osallistuvat käyttäjät voivat nähdä ja lähettää viestejä. Firestore Security Rules -säännöt määrittävät, että käyttäjät voivat lisätä vain omia viestejään ja poistaa ainoastaan omia viestejään.

Olisi ollut mahdollista valita myös Firebase Realtime Database mutta Firestore valittiin tietokantaratkaisuksi, koska se tarjoaa paremman skaalautuvuuden, reaaliaikaisen kuuntelun ilman ylimääräisiä API-kutsuja ja tehokkaamman kyselyn tuen verrattuna Firebase Realtime Databaseen. Firestore mahdollistaa myös monimutkaisempien kyselyiden, kuten viestien aikajärjestyksen, tehokkaan toteuttamisen, mikä on oleellista chat-sovelluksessa.

5 PROJEKTIN TOTEUTUS

Projektin toteutus eteni käytännönläheisesti ketterän kehityksen periaatteiden mukaisesti. Sovelluksen kehitys ja hallinta perustui selkeään työnjakoon, aktiiviseen viestintään ja tehokkaaseen versionhallintaan, jotta projekti saatiin tehtyä valmiiksi suunnitellusti. Projektin kehitys jaettiin viikoittaisiin sprintteihin, joissa jokaiselle sprintille asetettiin selkeät tavoitteet.

Tiimi teki lähes päivittäin yhteistyötä ja kommunikoi kehityksen edistymisestä. Yleisesti viestintä tapahtui Discordin ja WhatsAppin kautta. Lisäksi pidettiin viikoittain kaksi palaveria ohjaavan opettajan kanssa Teamsin välityksellä. Ohjaavan opettajan kanssa käydyissä palavereissa käytiin läpi sprintin aikana toteutetut muutokset, keskusteltiin esiin nousseista haasteista ja saatiin palautetta sovelluksen kehityksestä.

5.1 Kehitysvaiheet ja työnjako

Opinnäytetyön kehitysvaiheet voidaan jakaa neljään päävaiheeseen: suunnitteluun, toteutukseen, testaukseen ja virheiden korjaukseen. Kehitysprosessin aikana projektin etenemistä seurattiin viikoittaisissa palavereissa, joissa mukana olivat sekä ohjaava opettaja että projektiryhmän jäsenet.

Suunnitteluvaihe alkoi Figmassa, jossa mallinnettiin sekä yritys- että henkilöasiakaspuolen näkymät ja toiminnallisuudet. Tämä vaihe oli keskeinen, sillä sen avulla pystyttiin varmistamaan, että sovelluksen rakenne ja navigointi vastasivat loppukäyttäjien tarpeita. Teknologiavalinnat tehtiin tässä vaiheessa alustavasti, mutta Firebase päätettiin ottaa käyttöön vasta, kun backend-logiikkaa tarvittiin kirjautumiseen, rekisteröintiin, varausten hallintaan ja chattiin.

Toteutusvaihe eteni osa-alueittain, jotta kehitys pysyi hallittavana ja mahdollisti toistuvan testauksen. Ensimmäisenä toteutettiin sovelluksen perusrakenne, sivut ja navigointi, joiden avulla käyttäjät pystyvät liikkumaan sujuvasti sovelluksen eri osioiden välillä. Tämän jälkeen keskityttiin kirjautumis- ja

rekisteröitymisominaisuuksiin. Seuraavaksi siirryttiin varausten hallintaan, johon kuului kartta ja siihen liittyvät lisäominaisuudet. Viimeisimpänä toteutettiin chat-toiminnallisuus. Kaikki sovelluksen toiminnot edellyttävät sujuvaa tietokannan ja käyttäjätunnistuksen hallintaa. Ensimmäiseksi toteutettiin henkilöasiakaspuoli ja sen jälkeen yrityspuolen toiminnot.

Testausvaiheessa suoritettiin käytettävyystestauksia sekä virallisten testausmenetelmien että vapaamuotoisen käyttäjäpalautteen avulla. Käytännön testauksessa tarkasteltiin muun muassa kirjautumisen sujuvuutta, varausten oikeellisuutta sekä chatin toimintaa eri skenaarioissa. Havaitut virheet pyrittiin minimoimaan ja korjattiin systemaattisesti, ja kehitysprosessin lopussa tehtiin vielä viimeinen tarkistus, jotta sovellus vastasi laadullisia tavoitteita.

Työnjako määriteltiin jo ennen mobiilisovelluksen toteutuksen aloitusta, mikä teki kehityksestä tehokasta ja selkeästi seurattavaa. Alustavat roolit jakautuivat käyttäjäkokemuksen ja frontendin suunnitteluun sekä backendin toteutukseen. Työn etenemistä ja vastuujakoa seurattiin GitHubin Kanban-taulun avulla, johon ryhmän jäsenet saattoivat merkitä omat tehtävänsä, kirjata tehdyt työt sekä hallita tulevia kehitysvaiheita. Tämä työskentelytapa mahdollisti sujuvan yhteistyön ja tehokkaan tehtävien seurannan koko projektin ajan.

5.2 Versionhallinta ja koodin dokumentointi

Versionhallinta toteutettiin Gitin avulla, ja projektin koodi säilytettiin GitHubissa. Versionhallinnan avulla kehitystiimi pystyi työskentelemään rinnakkain ilman, että muutokset aiheuttivat ristiriitoja. Käytännössä jokainen ominaisuus toteutettiin omassa haarassaan, joka yhdistettiin päähaaraan vasta testauksen ja koodikatselmoinnin jälkeen. Tämä varmisti, että päähaara säilyi aina toimivana.

GitHubin pull request -toiminto mahdollisti koodin tarkistamisen ennen yhdistämistä, mikä paransi koodin laatua ja auttoi havaitsemaan mahdolliset virheet jo varhaisessa vaiheessa. Lisäksi versionhallintaa hyödynnettiin dokumentoinnissa siten, että muutokset merkittiin commit-viesteihin mahdollisimman selkeästi. (Git 2025.)

Versionhallinnan työkaluna käytettiin Git Bashia, jonka avulla kaikki tärkeimmät versionhallintatoimenpiteet, kuten commitit ja branchien hallinta, suoritettiin komentoriviltä. Git Bash tarjosi kehittäjille tehokkaan tavan hallita projektia. (Git 2025.)

Koodin dokumentointi oli olennainen osa kehitysprosessia. Jokainen merkittävä muutos dokumentoitiin kommenttien avulla, jotta kehitystiimin jäsenet pystyivät ymmärtämään niiden tarkoituksen ja toimintalogiikan. Projektin edetessä dokumentointi pidettiin ajan tasalla koodin sisällä.

README.md-tiedosto koottiin kehitysprosessin lopussa varmistaen, että se vastasi projektin lopullista toteutusta ja kattoi kaikki olennaiset osa-alueet. Dokumentaatio laadittiin tarjoamaan selkeät ja yksityiskohtaiset ohjeet sovelluksen asentamiseen, käyttöönottoon ja tekniseen rakenteeseen. (Github 2025.)

README.md sisältää projektin kuvauksen, jossa esitellään sovelluksen käyttötarkoitus ja keskeiset ominaisuudet, kuten autojen selaaminen ja varaaminen ja karttapohjainen navigointi. Myös yrityskäyttäjien mahdollisuus lisätä ja hallita ajoneuvojaan esitellään. Lisäksi tiedostoon kirjattiin yksityiskohtaiset asennusohjeet, mukaan lukien tarvittavat teknologiat, riippuvuuksien asentaminen ja Expo-ympäristön käyttö. Konfigurointiosiossa annettiin ohjeet Firebase- ja Mapbox-avainten asettamiseen, jotta sovellus voidaan käynnistää oikein.

README.md-tiedoston selkeä rakenne ja kattavuus tekevät siitä hyödyllisen myös jatkokehityksessä ja sovelluksen ylläpidossa. Dokumentaatio mahdollistaa uusien kehittäjien nopean perehdytyksen ja auttaa ylläpitämään sovelluksen kehitystyötä systemaattisesti. (GitHub 2025.)

6 KÄYTETTÄVYYSTESTAUS

Käytettävyystestaus on keskeinen osa sovelluskehitystä, sillä sen avulla voidaan varmistaa, että sovellus on käyttäjäystävällinen ja vastaa käyttäjien tarpeita. Testauksen tavoitteena on tunnistaa mahdolliset käytettävyysongelmat sekä arvioida sovelluksen selkeyttä ja toiminnallisuutta eri käyttäjäryhmien näkökulmasta.

Tässä testauksessa sovelluksen toimivuutta arvioitiin sekä henkilö- että yritysasiakkaan roolissa. Lisäksi testattiin sovelluksen suorituskykyä eri alustoilla ja laitteilla, jotta voitiin varmistaa sen sujuva käyttö erilaisissa ympäristöissä. Käyttäjien palautteen perusteella tehtiin tarvittavia parannuksia, joiden avulla sovelluksen käytettävyyttä voitiin kehittää entistä sujuvammaksi. (Taulukko 1.)

Taulukko 1. Käytettävyystestauksen havainnot

Teema	Yhteenveto	Kehitysehdotus
Rekisteröityminen ja kirjautuminen	Koettiin sujuvaksi ja selkeäksi. Harvoja epäselvyyksiä.	Ei erityisiä
Karttanäkymä	Pidettiin hyödyllisenä, mutta sijaintiluvan puuttuminen aiheutti kaatumista.	Oletussijainti lisätty.
Chat-toiminto	Hyödyllinen toiminto, mutta vaikea löytää.	Sijoittelua ja näkyvyyttä parannettava.
Hakutoiminto	Käyttäjät eivät aina ymmärtäneet, että hakukriteerit eivät ole pakollisia.	Lisättävä ohjeistusta tai oletusvalinta 'näytä kaikki'.
Yritysasiakkaan näkymä	Toimiva ja selkeä, erityisesti auton lisääminen sai kiitosta.	Lisättävä myös automerkit valmiiksi valinnoiksi.
Hyödyllisyys ja suositeltavuus	Useimmat kokivat sovelluksen hyödylliseksi ja aikaa säästäväksi.	Perusratkaisu toimiva.

6.1 Käytettävyystestauksen tavoitteet

Käytettävyystestauksen päätavoitteena oli selvittää, kuinka selkeä ja helppokäyttöinen sovellus on, sekä tunnistaa mahdolliset ongelmat ja kehitystarpeet. Testauksessa arvioitiin erityisesti sovelluksen navigointia, rekisteröitymis- ja varausprosessin sujuvuutta, karttanäkymän toimivuutta sekä chat-toiminnon löydettävyyttä. Sovelluksen tuli palvella sekä henkilö- että yritysasiakkaita, joten testauksen avulla haluttiin varmistaa, että molemmat käyttäjäryhmät pystyvät hyödyntämään sovelluksen keskeisiä toimintoja vaivattomasti.

Testaukseen osallistuvat kokeilivat sovellusta sekä henkilöasiakkaan että yritysasiakkaan roolissa, jotta saatiin kattava käsitys eri toimintojen käytettävyydestä. Molempien käyttäjäryhmien näkökulmat olivat tärkeitä ja sovelluksen oli toimittava sujuvasti kummallekin ryhmälle.

Lisäksi testauksessa kiinnitettiin huomiota sovelluksen tekniseen toimivuuteen eri laitteilla ja käyttöjärjestelmissä. Responsiivisuuden testaaminen oli keskeinen osa prosessia, sillä sovelluksen tuli mukautua erilaisiin näyttökokoihin ilman käyttöliittymäongelmia. Käytettävyyden lisäksi arvioitiin sovelluksen suorituskykyä ja mahdollista viivettä eri toiminnoissa, kuten hakutulosten latautumisessa ja varausprosessissa.

6.2 Testauksen toteutus

Käytettävyystestauksen suunnittelu keskittyi siihen, että testattavat toiminnot kattaisivat sovelluksen keskeiset käyttötapaukset ja tarjoaisivat kattavan kuvan käyttäjäkokemuksesta. Testaajille laadittiin yksityiskohtaiset ohjeet, joiden avulla he etenivät sovelluksessa molemmissa rooleissa. Näin saatiin kattava käsitys siitä, miten eri käyttäjäryhmät hahmottavat sovelluksen ja kuinka sujuvasti he löytävät tarvitsemansa toiminnot.

Testaus suoritettiin sekä Android- että iOS-laitteilla, jotta mahdolliset alustakohtaiset erot saatiin selvitettyä. Lisäksi testauksessa käytettiin Android Studio -emulaattoria, jonka avulla voitiin simuloida erilaisia käyttöympäristöjä ja varmistaa

sovelluksen toimivuus eri laitteilla. Testauksen alussa käyttäjille annettiin lyhyt ohjeistus sovelluksen käyttämiseen, mutta heidän annettiin edetä mahdollisimman itsenäisesti, jotta voitiin arvioida, kuinka helposti eri toiminnot löytyivät.

Testauksessa keskityttiin erityisesti varaustoimintoon, koska se oli sovelluksen keskeisin ominaisuus. Käyttäjät arvioivat varausprosessin selkeyttä ja sitä, kuinka helposti he pystyivät löytämään ja varaamaan autoja. Lisäksi karttanäkymän toimivuutta arvioitiin sen perusteella, kuinka selkeästi autot näkyivät ja kuinka hyvin reittitoiminto ohjasi käyttäjän noutopaikkaan. Chat-toiminnon löydettävyys ja käytettävyys testattiin varmistamalla, että käyttäjät pystyivät helposti kommunikoimaan yrityksen kanssa varaukseen liittyvissä asioissa.

Yritysasiakkaan näkökulmasta testauksessa keskityttiin auton lisäämiseen, muokkaamiseen ja hallintaan. Testaajat arvioivat, kuinka selkeä ja looginen auton lisäämisprosessi oli ja kuinka helposti jo lisättyjä autoja pystyi päivittämään tai poistamaan. Lisäksi chat-toimintoa testattiin varmistamalla, että viestit välittyivät oikein ja että viestejä pystyy lähettämään vaivattomasti.

Testaukseen osallistui kavereita, perheenjäseniä ja läheisiä, jotka eivät olleet mukana projektin kehittämisessä. Tämä mahdollisti eri tasoisten käyttäjien osallistumisen, sillä osa testaajista oli täysin uusia sovellustestaukselle, kun taas osalla oli aiempaa kokemusta ohjelmistokehityksestä ja koodauksesta. Tämä auttoi saamaan monipuolista palautetta sekä teknisestä toimivuudesta että sovelluksen käytettävyydestä.

Testauksen aikana käyttäjät suorittivat tehtäviä, jotka vastasivat sovelluksen tyypillisiä käyttötapauksia, kuten auton varaamista, tietojen päivittämistä, varauksen perumista ja chatin käyttämistä. Käyttäjät antoivat arvioita sovelluksen eri osa-alueista asteikolla 1–5 ja vastasivat avoimiin kysymyksiin, joissa he kuvasivat kokemuksiaan ja mahdollisia ongelmakohtia. Testitulokset kerättiin Forms-kyselyllä ja analysoitiin tunnistamalla toistuvat teemat ja kehitystarpeet, joiden perusteella sovellusta voitiin parantaa entistä käyttäjäystävällisemmäksi.

6.3 Havaitut ongelmat ja kehitysehdotukset

Käytettävyystestauksessa esiin nousseita ongelmia tarkasteltiin yksityiskohtaisesti, ja niiden perusteella tehtiin useita kehitysehdotuksia, joilla sovelluksen käytettävyyttä voitiin parantaa. Karttanäkymän kaatuminen oli yksi merkittävimmistä ongelmista, ja se havaittiin johtuvan siitä, että sovellus ei saanut lupaa käyttää käyttäjän sijaintia. Tämä johti tilanteeseen, jossa kartta ei latautunut kunnolla ja aiheutti sovelluksen kaatumisen. Ongelma korjattiin niin, että mikäli käyttäjä ei anna lupaa sijainnin käyttöön, sovellus asettaa oletuspaikaksi Helsingin, mikä takaa karttanäkymän toimivuuden kaikille käyttäjille.

Chat-toiminnon löydettävyys koettiin haastavaksi. Erityisesti henkilöasiakkaat kokivat sen jäävän helposti huomaamatta, ja osa käyttäjistä ei huomannut chatin olemassaoloa. Yritysasiakkailla sen käyttöä hankaloitti kirjoitusalueen sijoittuminen näppäimistön taakse joillakin mobiililaitteilla. Tämä teki viestien kirjoittamisesta vaikeaa. Parannuksena chat-toiminnon viestikentän sijoittelua muokattiin niin, että näppäimistö ei peitä sitä.

Hakutoiminnossa havaittiin epäselvyyksiä erityisesti siinä, miten hakukriteereitä tuli käyttää. Osa käyttäjistä ei ymmärtänyt, että hakuehdot eivät ole pakollisia, mikä johti turhiin täyttövaiheisiin ja haun rajoittamiseen tarpeettomasti. Istumapaikkojen määrän esitystapa herätti epäselvyyttä. Käyttäjille ei ollut selvää, tarkoittiko ilmoitettu luku vapaita paikkoja vai auton kokonaiskapasiteettia. Käyttäjät ehdottivat myös, että hakuun voisi lisätä valmiin vaihtoehdon, jossa kaikki vaihtoehdot olisivat oletuksena valittuina. Testauksessa havaittiin, että jos käyttäjä ei ollut kirjautuneena sovellukseen ja hän etsi auton ennen sisäänkirjautumista, hänen täytyi kirjautumisen jälkeen valita auto uudelleen. Tämä teki varausprosessista tarpeettoman monimutkaisen ja lisäsi ylimääräisiä vaiheita.

Testaus toi esiin myös muita haasteita, joista merkittävimpiä olivat responsiivisuusongelmat ja käyttöliittymän elementtien sijoittelu eri näyttökokoihin. Testauksessa havaittiin, että alareunan navigointipalkki ei kaikilla laitteilla skaalautunut oikein, minkä vuoksi osa painikkeista leikkautui pois näytöltä. Lisäksi eräissä näkymissä painikkeiden muotoilu ei ollut yhtenäinen, mikä heikensi sovelluksen visuaalista johdonmukaisuutta. Näitä ongelmia korjattiin hyödyntämällä

joustavampia asetteluratkaisuja ja välttämällä kiinteitä pikseliarvoja, jotta sovellus mukautuu paremmin erikokoisille näytöille ja erilaisille laitteille. Käyttöliittymän skaalautuvuutta parannettiin ottamalla käyttöön ScrollView-komponentti. Sen avulla kaikki käyttöliittymän elementit pysyvät näkyvissä myös pienemmillä näytöillä. Ratkaisu tekee sovelluksen käytöstä sujuvampaa. Se estää tilanteet, joissa käyttäjä ei pääse käsiksi tärkeisiin toimintoihin, koska ne jäävät näytön ulkopuolelle.

Testauksessa havaittujen ongelmien perusteella tehtiin monia merkittäviä parannuksia, jotka paransivat sovelluksen käytettävyyttä huomattavasti. Näiden muutosten myötä sovellus on entistä selkeämpi, ja sen käyttö sujuvampaa eri laitteilla ja eri käyttäjäryhmille. Vaikka testauksessa nousi esiin kehitystarpeita, sovellus sai paljon positiivista palautetta sen perustoiminnallisuuksien selkeydestä ja käyttöliittymän yksinkertaisuudesta. Jatkokehityksessä voidaan edelleen keskittyä erityisesti siihen, että käyttöliittymä säilyy mahdollisimman loogisena ja kaikki tärkeimmät toiminnot ovat helposti saatavilla ilman tarpeettomia vaiheita.

6.4 Käytettävyystestauksen johtopäätökset

Käytettävyystestaus tarjosi arvokasta tietoa sovelluksen vahvuuksista ja kehityskohdista, joiden perusteella voitiin tehdä konkreettisia parannuksia käyttäjäkokemuksen sujuvoittamiseksi. Testin tulokset osoittivat, että sovellus vastasi pääosin käyttäjien tarpeisiin ja sen perusominaisuudet toimivat odotetusti. Sovelluksen selkeä rakenne ja looginen navigointi saivat kiitosta, ja erityisesti varausprosessia pidettiin onnistuneena. Monet käyttäjät kokivat sen nopeaksi ilman tarpeettomia välivaiheita, mikä vahvisti sovelluksen käytettävyyden perustaa.

Vaikka sovelluksen keskeiset toiminnallisuudet toimivat hyvin, testaus toi esiin myös kehitysalueita, jotka vaikuttivat käyttökokemukseen. Esimerkiksi sovellusta ensimmäistä kertaa käyttäneille ei ollut aina täysin selvää, miten tietyt toiminnot toimivat. Tämä osoitti tarpeen selkeämmälle ohjeistukselle tai käyttöliittymän hienosäädölle. Testitulosten perusteella jatkokehityksessä voidaan keskittyä siihen, että käyttöliittymä tarjoaa riittävästi opastusta ilman, että se hidastaa kokeneempien käyttäjien toimintaa.

Käyttäjät pitivät autojen hakua ja varaamista sujuvana, mutta toivat esiin tarpeen lisäominaisuuksille, jotka mahdollistaisivat entistä tarkemmat valinnat. Erityisesti hakukriteerien laajentaminen ja varausjärjestelmään lisättävä aikaväliominaisuus koettiin hyödyllisiksi kehityskohteiksi. Aikaväliominaisuus antaisi käyttäjälle mahdollisuuden määrittää noutoajan tarkemmin. Vaikka sovelluksen perustoiminnot olivat selkeitä, testaus osoitti, että käyttäjien tarpeet vaihtelevat. Tietyt tilanteet voivat edellyttää joustavampia vaihtoehtoja käytettävyyden parantamiseksi.

Testauksen aikana saatu palaute osoitti myös, että tiettyjen ominaisuuksien löydettävyyttä ja käytettävyyttä voitaisiin parantaa, jotta käyttäjät pystyvät hyödyntämään sovelluksen kaikkia toimintoja mahdollisimman tehokkaasti. Esimerkiksi etusivulle kaivattiin enemmän informatiivisuutta, jotta käyttäjä saisi heti sovelluksen avattuaan paremman käsityksen tarjolla olevista vaihtoehdoista ja omista varauksistaan. Käyttäjäkokemuksen jatkuva optimointi edellyttää sitä, että sovellus tarjoaa sekä selkeät perusominaisuudet että joustavia vaihtoehtoja erilaisten tarpeiden mukaan.

Testitulosten analysointi vahvisti, että käytettävyystestauksella on keskeinen rooli ohjelmistokehityksessä. Sen avulla voitiin tunnistaa sekä teknisiä että käyttöliittymään liittyviä haasteita. Lisäksi se auttoi ymmärtämään, miten käyttäjät hahmottavat sovelluksen eri toiminnot ja mitkä tekijät vaikuttavat heidän tyytyväisyyteensä. Jatkokehityksessä voidaan hyödyntää testituloksista saatuja oppeja erityisesti käyttöliittymän hienosäätämiseen, selkeämpien ohjeiden tarjoamiseen ja siihen, että sovellus säilyttää loogisen ja helposti navigoitavan rakenteensa eri käyttäjäryhmille.

Kaiken kaikkiaan testitulokset osoittivat, että sovelluksen perusominaisuudet olivat hyvin toteutettuja ja toimivat odotetusti, mutta pienillä muutoksilla voidaan vielä parantaa käytettävyyttä ja selkeyttää joitakin toimintoja. Käytettävyystestauksen merkitys ohjelmistokehityksessä korostuu erityisesti siinä, että käyttäjäkokemusta voidaan kehittää käyttäjien aidon palautteen perusteella, jolloin sovellus vastaa paremmin todellisiin käyttötarpeisiin.

7 POHDINTA

Tässä luvussa arvioidaan opinnäytetyön ja sovelluskehitysprojektin kokonaisuutta, tarkastellaan projektin onnistumisia ja haasteita sekä pohditaan mahdollisia jatkokehityssuuntia. Lisäksi käsitellään opinnäytetyön tavoitteiden täyttymistä ja projektin merkitystä ammatillisen osaamisen kehittämisessä.

7.1 Projektin onnistumiset ja haasteet

Projektin aikana saavutettiin merkittäviä onnistumisia, erityisesti sovelluksen toiminnallisuuksien kehittämisessä ja käytettävyyden parantamisessa. Yksi keskeisistä onnistumisista oli sovelluksen päätoimintojen, kuten siirtoautojen haun, varaamisen ja hallinnan, saaminen käyttäjäystävälliseen mobiilisovellukseen. Käyttöliittymän suunnittelu Figman avulla osoittautui tehokkaaksi tavaksi varmistaa selkeä navigaatio ja hyvä käyttäjäkokemus jo varhaisessa vaiheessa.

Teknologisesti projektin suurimpia onnistumisia oli React Native ja Firebase -ympäristön hyödyntäminen, mikä mahdollisti nopean kehityksen ja skaalautuvan tietokantaratkaisun. Firebase Authentication helpotti käyttäjähallinnan toteuttamista, ja Firestore-tietokanta mahdollisti reaaliaikaisen tiedonsiirron, mikä paransi varaustoimintojen sujuvuutta. Sovelluskehityksessä käytetty Scrum-malli ja viikoittaiset sprintit auttoivat projektin hallinnassa ja edistymisen seurannassa, mikä paransi työskentelyn tehokkuutta.

Käytettävyystestauksessa saatu palaute oli pääosin positiivista, ja testien tulosten pohjalta tehtiin merkittäviä parannuksia, kuten karttanäkymän kaatumisen korjaaminen ja chat-toiminnon löydettävyyden parantaminen. Tämä osoitti, että käyttäjäpalautteen hyödyntäminen oli projektin vahvuuksia.

Projektin suurimpia haasteita olivat muun muassa sijaintitietojen käsittely karttanäkymässä, sillä sovellus ei aluksi osannut käsitellä tilannetta, jossa käyttäjä ei myöntänyt lupaa sijaintitietojen käyttöön. Tämä korjattiin määrittämällä oletussijainti. Lisäksi käyttöliittymän responsiivisuus eri laitteilla aiheutti haasteita, ja

joidenkin painikkeiden sijoittelua ja skaalausta jouduttiin säätämään useita kertoja. Toinen merkittävä haaste oli aikataulun hallinta ja resurssien riittävyys. Koska projektin laajuus oli suhteellisen suuri kahden kehittäjän tiimille, kaikkien suunniteltujen ominaisuuksien toteuttaminen vaati tarkkaa priorisointia.

Kaiken kaikkiaan projekti onnistui tavoitteidensa mukaisesti, ja se tarjosi arvokasta kokemusta sovelluskehityksen käytännön toteutuksesta, projektinhallinnasta ja tiimityöskentelystä. Haasteista huolimatta lopputulos vastasi käyttäjien tarpeita ja loi vahvan pohjan sovelluksen mahdolliselle jatkokehitykselle.

7.2 Sovelluksen mahdollinen jatkokehitys

Sovelluksen mahdollinen jatkokehitys keskittyy liiketoimintamallien kehittämiseen ja teknologian ylläpitoon. Mahdollisia ansaintamalleja ovat varausmaksut, jäsenyydet ja yhteistyökumppanuudet, jotka voivat tukea palvelun kasvua ja ylläpitoa. Käyttäjäkokemusta voidaan parantaa lisäpalveluilla, kuten mukautetuilla tarjouksilla ja lisäominaisuuksilla.

Teknologia puolella sovellus hyödyntää Expoa ja Firebasea, jotka tarjoavat helpon päivitettävyyden ja hyvän suorituskyvyn. Expon avulla React Native -päivitysten hallinta on yksinkertaista, ja tarvittaessa voidaan siirtyä laajempaan natiivikehitykseen. Firebase skaalautuu automaattisesti, mutta kustannusten kasvaessa voi olla tarpeen optimoida tietokantarakenteita tai siirtyä omaan backend-ratkaisuun.

7.2.1 Tulevaisuuden liiketoimintamahdollisuudet

Sovelluksen kaupallistaminen on täysin mahdollista, sillä kansainvälisistä esimerkeistä, kuten Movacarista ja DriiveMe-palveluista, on nähtävissä, että siirtoautojen välittäminen voi olla toimiva liiketoimintamalli. Näissä palveluissa käyttäjiltä peritään nimellinen maksu jokaisesta varauksesta, mikä takaa palvelun jatkuvuuden ja kattaa sen ylläpidosta syntyviä kuluja (DriiveMe s.a.; Movacar s.a.). Suomessa kilpailu siirtoautojen markkinapaikalla on vielä vähäistä, mikä antaa mahdollisuuden ensimmäisen toimijan etuun. Käyttäjien näkökulmasta pieni,

esimerkiksi euron suuruinen välityspalkkio varauksesta ei todennäköisesti muodostu esteeksi, mutta suuremmassa mittakaavassa se voisi tuottaa vakaata tulovirtaa palvelulle.

Kaupallistamismalli voisi perustua kuukausi- tai vuosijäsenyyteen, jossa käyttäjät maksaisivat palvelun käytöstä säännöllisesti. Jäsenyyden avulla käyttäjät voisivat saada etuja, kuten etuoikeuden varata autoja ennen muita, rajattoman pääsyn tarjouksiin tai erikoisalennuksia pidemmille matkoille. Jäsenyyteen perustuva liiketoimintamalli voisi tarjota vakaamman kassavirran, mutta se voisi samalla rajoittaa käyttäjien määrää alkuvaiheessa, jos ilmaisten tai edullisten varausten houkuttelevuus kärsii.

Sovelluksen merkittävin tulonlähde voisi kuitenkin syntyä yhteistyösopimuksista vuokraamojen ja muiden autoalan toimijoiden kanssa. Autovuokraamot kohtaavat jatkuvasti tarpeen siirtää autoja paikasta toiseen, ja heille on huomattavasti edullisempaa hyödyntää sovelluksen käyttäjiä kuin maksaa kuljetusyrityksille tai omille työntekijöille siirtoajojen suorittamisesta. Esimerkiksi kansainvälisesti toimiva Transfercar perustaa liiketoimintamallinsa juuri vuokraamojen kustannussäästöihin. Suomessa tämä voisi tarkoittaa sopimuksia suurimpien autovuokraamoketjujen ja autokauppojen kanssa, jolloin yritykset maksaisivat palvelulle joko listausmaksuja autojen ilmoittamisesta sovelluksessa tai provisiopalkkioita onnistuneista siirroista. (Transfercar s.a.)

Lisäksi palveluun voitaisiin integroida lisäpalveluita, joiden avulla voitaisiin kasvattaa tuottoja. Esimerkiksi huoltoasemien kanssa tehtävä yhteistyö voisi mahdollistaa etuja sovelluksen käyttäjille, kuten alennuksia tai ilmaisia kahveja matkan varrella. Myös matkavakuutukset, lisäkilometrit tai mahdollisuus pidempään käyttöaikaan pientä lisämaksua vastaan voisivat tuoda lisätuloja ilman, että ne vaikuttaisivat varsinaisen varauspalvelun perusmaksuihin.

Sovelluksen ansaintamallia voidaan laajentaa myös sisäisen mainonnan kautta, mikä tarjoaa uusia tulonlähteitä ilman, että se häiritsee käyttäjäkokemusta. Yritykset, kuten huoltoasemat, vakuutusyhtiöt, autopesulat ja vuokraamot, voivat hyödyntää sovellusta markkinointikanavanaan ja tavoittaa suoraan autojen siirroista kiinnostuneita käyttäjiä. Yrityksille voidaan myös tarjota mahdollisuus ostaa

parempaa näkyvyyttä sovelluksessa, jolloin heidän tarjoamansa ajoneuvot tai palvelut nousevat esiin hakutuloksissa. Tämä toimintamalli voisi houkutella erityisesti pieniä ja keskisuuria yrityksiä, jotka eivät muuten pystyisi kilpailemaan suurten toimijoiden kanssa digitaalisten markkinointikanavien kautta. Esimerkiksi autovuokraamot voisivat maksaa siitä, että heidän autonsa näkyvät korostettuina, mikä lisäisi niiden varaustodennäköisyyttä ja nopeuttaisi siirtojen järjestämistä.

Markkinapotentiaali Suomessa on rajallinen mutta kiinnostava, erityisesti sesonkiaikoina ja tietyillä reiteillä. Esimerkiksi Lapin ja Etelä-Suomen välillä esiintyy selkeää tarvetta kausiluonteisille siirroille, kun vuokraamot haluavat palauttaa autoja takaisin etelään talvikauden jälkeen. Pitkät siirtoajot ovat houkuttelevia käyttäjille, sillä ne mahdollistavat edullisen tai täysin ilmaisen kyydin, ja samalla ne ovat välttämättömiä vuokraamoille. Myös pääkaupunkiseudulta muualle Suomeen voi syntyä vastaavia palautussiirtoja suurten tapahtumien tai ruuhka-aikojen jälkeen.

Haasteena on kuitenkin kaksipuolinen markkina, jossa täytyy samanaikaisesti houkutella riittävästi sekä firmoja että käyttäjiä, jotta palvelun kysyntä ja tarjonta kohtaavat. Sovelluksen alkuvaiheessa voi olla järkevää aloittaa yhteistyö yhden firman kanssa tietyllä reitillä ja testata konseptin toimivuutta ennen laajentamista. Markkinointia voitaisiin kohdentaa erityisesti opiskelijoille, reppureissaajille ja muille budjettimatkailijoille, jotka hyötyisivät siirtoautopalvelun tarjoamasta edullisesta matkustusvaihtoehdosta.

Kumppanuuksien laajentaminen autoliikkeiden ulkopuolelle voisi avata uusia mahdollisuuksia palvelun kehittämiseen. Esimerkiksi matkailuyritykset, kaupungit tai alueelliset matkailutoimistot voisivat nähdä palvelun osana edullista matkailukonseptia ja markkinoida sitä eteenpäin. Autoalan yritykset saattavat tarvita siirtoja toimipisteidensä välillä, ja sovellusta voisi kehittää myös näiden yksityisten tai yritysten siirtoajojen hallintaan. Tällöin palvelu voisi veloittaa suoraan yrityksiltä palvelumaksun auton siirroista.

Vaikka suomalainen markkina ei ole valtava, konsepti voisi pitkällä aikavälillä skaalautua myös Suomen ulkopuolelle, esimerkiksi Baltiaan tai muihin Pohjoismaihin, joissa vastaavanlaisia palveluita on jo toiminnassa. Kansainväliset

kokemukset osoittavat, että siirtoautopalvelut voivat menestyä, kun ne on toteutettu oikein ja kysynnän sekä tarjonnan tasapaino saadaan kohdilleen.

7.2.2 Teknologian ylläpito

Sovelluksen pitkäaikainen ylläpito ja yhteensopivuus uusien React Native -versioiden kanssa edellyttää säännöllistä päivitystä ja teknologiaratkaisujen joustavuutta. Expo-pohjainen toteutus tarjoaa helpon tavan päivittää sovellus uusimpiin React Native -versioihin, sillä Expo hallitsee monia riippuvuuksia automaattisesti. Tämä helpottaa ylläpitoa ja vähentää manuaalista työtä verrattuna suoraan React Native CLI:llä toteutettuun projektiin. Lisäksi Expon tarjoama jatkuva tuki ja päivitykset varmistavat, että sovellus pysyy ajan tasalla uusimpien mobiilialustojen ja laiteominaisuuksien kanssa. (Expo s.a. a.)

Expon käyttö tuo mukanaan myös rajoituksia, erityisesti silloin, jos sovellukseen halutaan lisätä laajennettuja natiivimoduuleja tai muita vaativia ominaisuuksia. Tässä tapauksessa sovellus voi hyödyntää npx expo prebuild -komentoa, joka luo ja päivittää natiiviprojektin automaattisesti sovelluksen käyttämien kirjastojen ja asetusten perusteella. Tämä mahdollistaa mukautetun natiivikoodin lisäämisen ilman, että sovellus menettää Expon tarjoamat kehityksen ja hallinnan hyödyt. Näin ollen nykyinen teknologiapino mahdollistaa nopean kehityksen alkuvaiheessa, mutta jättää silti tilaa skaalautuvuudelle ja laajennuksille tulevaisuudessa. (Expo 2025.)

Sovelluksen skaalautuvuuden kannalta Firebase tarjoaa vahvan taustajärjestelmän, joka on suunniteltu kestämään suuria käyttäjämääriä. Firestore skaalautuu dynaamisesti kuormituksen mukaan, mikä tarkoittaa, että sovelluksen suorituskyky ei kärsi, vaikka käyttäjämäärä kasvaisi merkittävästi. Tämä on tärkeää erityisesti silloin, kun sovellukseen tulee samanaikaisesti useita käyttäjiä, jotka selaavat ja varaavat autoja reaaliaikaisesti (Firebase 2025a). Firebasen suurin haaste on kuitenkin hinnoittelu, sillä käyttömäärän kasvaessa myös palvelun kustannukset nousevat. Tämän vuoksi on tärkeää optimoida tietokantarakenteet ja vähentää turhia tietokantakutsuja esimerkiksi välimuistiratkaisuilla. (Firebase 2025e.)

Tekoälyä voisi hyödyntää parantamaan hakutuloksia ja tekemään käyttäjille sopivampia suosituksia. Se voisi myös auttaa arvioimaan, missä ja milloin autoja tarvitaan eniten, jotta ne olisivat saatavilla oikeaan aikaan oikeassa paikassa. Tämä voisi tehdä palvelusta sujuvamman ja tehokkaamman sekä käyttäjille että autojen tarjoajille. Tekoälyä voisi hyödyntää myös chatbotissa, joka vastaisi käyttäjien kysymyksiin nopeasti ja auttaisi esimerkiksi varausten tekemisessä, perumisessa tai ongelmatilanteissa. Chatbot voisi olla käytettävissä ympäri vuorokauden ja tarjota nopeita vastauksia ilman, että käyttäjän tarvitsee odottaa asiakaspalvelijaa. Tämä tekisi asiakaspalvelusta sujuvampaa ja helpottaisi käyttäjäkokemusta.

Expo ja Firebase muodostavat toimivan teknologiapinon tuotantokäyttöön, erityisesti sovelluksen alkuvaiheessa. Ne mahdollistavat nopean julkaisun ja helpon ylläpidon ilman suuria kustannuksia. Jos sovellus kasvaa ja vaatimukset muuttuvat, teknologiaa voidaan mukauttaa esimerkiksi siirtymällä omaan backend-ratkaisuun tai hyödyntämällä natiivikoodia. Tämä joustava lähestymistapa varmistaa, että sovellus pysyy ajan tasalla ja skaalautuu tarpeiden mukaan.

7.3 Opinnäytetyön tavoitteiden täyttyminen

Opinnäytetyön tavoitteena oli kehittää toimiva ja käyttäjäystävällinen prototyyppi siirtoautojen hallintaan ja varaamiseen. Työ keskittyi sekä tekniseen toteutukseen että palvelun käytettävyyden parantamiseen. Sen aikana sovellettiin modernia sovelluskehitystä ja projektinhallinnan menetelmiä. Lisäksi tavoitteena oli oppia hyödyntämään React Native -kehitysympäristöä ja Firebase-palveluita tehokkaasti sekä toteuttaa skaalautuva ja tietoturvallinen ratkaisu, joka voisi toimia pohjana sovelluksen mahdolliselle jatkokehitykselle.

Tavoitteet täyttyivät pääosin suunnitellusti. Sovellus toteutettiin React Native -kehitysympäristössä, ja sen keskeiset toiminnallisuudet saatiin onnistuneesti käyttöön. Firebase-tietokanta mahdollisti reaaliaikaisen tietojen hallinnan, ja käyttökokemusta parannettiin palautteen perusteella. Käyttäjähallinnan ja autentikoinnin toteutus Firebase Authenticationilla osoittautui toimivaksi ratkaisuksi, mikä

helpotti rekisteröitymisprosessia ja varmisti käyttäjien tietoturvan. Sovelluksen käyttöliittymän suunnittelussa panostettiin selkeyteen ja sen kehityksessä huomioitiin käyttäjäpalautteen kautta saadut parannusehdotukset.

Kehitysprosessin aikana opittiin arvokkaita taitoja sovelluskehityksestä, tietoturvasta ja projektinhallinnasta. Työ tarjosi käytännön kokemusta niin teknisestä toteutuksesta kuin tiimityöskentelystäkin. Lisäksi työ antoi pohjan mahdolliselle jatkokehitykselle ja liiketoimintamahdollisuuksien kartoittamiselle. Projektin toteutus vaati myös kykyä soveltaa ohjelmistokehityksen parhaita käytäntöjä ja etsiä ratkaisuja esiin nousseisiin haasteisiin, kuten karttanäkymän tietojen käsittelyyn ja käyttöliittymän responsiivisuuteen eri laitteilla.

Vaikka projektin aikataulun hallinta ja resurssien riittävyys olivat haasteellisia, työ saatiin valmiiksi suunnitelman mukaisesti. Projektiin liittyvät sprintit auttoivat hallitsemaan työmäärää ja priorisoimaan tärkeitä toiminnallisuuksia. Erityisesti Scrum-menetelmien käyttö osoittautui hyödylliseksi, sillä se mahdollisti joustavan kehitysprosessin, jossa uusia ominaisuuksia voitiin testata ja parantaa käyttäjäpalautteen perusteella.

Opinnäytetyö keskittyi käytännön sovelluksen toteutukseen, mikä teki siitä arvokkaan oppimisprosessin ja antoi hyvät valmiudet jatkokehitystä varten. Sovelluksen perusominaisuudet toimivat odotetusti, ja projektin aikana syntyi selkeitä ideoita jatkokehitystä varten. Tulevaisuudessa sovellusta voidaan laajentaa uusilla ominaisuuksilla, kuten paremmalla analytiikalla, tekoälyavusteisella suositusjärjestelmällä sekä entistä kehittyneemmällä varausjärjestelmällä.

Työ myös osoitti, että React Native -pohjainen kehitys on tehokas tapa toteuttaa alustariippumaton mobiilisovellus, mutta se vaatii tarkkaa optimointia ja käytettävyyden testausta eri laitteilla. Lisäksi Firebase osoittautui joustavaksi tietokantaratkaisuksi, mutta sen hinnoittelu ja skaalautuvuus asettavat pidemmällä aikavälillä haasteita, jotka on huomioitava, mikäli sovelluksen käyttäjämäärä kasvaa merkittävästi.

Kaiken kaikkiaan opinnäytetyö saavutti sille asetetut tavoitteet ja tarjosi tekijöille arvokasta kokemusta sovelluskehityksen eri osa-alueilta. Samalla työ auttoi hahmottamaan ohjelmistoprojektin tekemistapaa ja siihen liittyviä haasteita

käytännön tasolla, mikä on tärkeää tulevaisuuden työelämässä. Tiimi sai arvokasta kokemusta myös projektinhallinnasta ja tiimityöskentelystä.

LÄHTEET

Corellia 2023. Adobe XD vai Figma? Mitä eroa ohjelmistoilla ja mihin ne parhaiten soveltuvat? Luettavissa: https://corellia.fi/adobe-xd-vai-figma-mita-eroa-ohjelmistoilla-ja-mihin-ne-parhaiten-soveltuvat/. Luettu 28.1.2025.

DriiveMe. s.a. €1 The easiest way to move a vehicle. Luettavissa: https://www.driiveme.co.uk/. Luettu 26.2.2025.

Expo 2024. React Native WebView. Luettavissa: https://docs.expo.dev/versions/latest/sdk/webview/. Luettu 1.2.2025.

Expo. 2025. Continuous Native Generation (CNG). Luettavissa: https://docs.expo.dev/workflow/continuous-native-generation/. Luettu 26.2.2025.

Expo s.a. a. Create amazing apps that run everywhere. Luettavissa: https://docs.expo.dev/. Luettu 30.1.2025.

Expo s.a. b. Expo Location. Luettavissa: https://docs.expo.dev/versions/latest/sdk/location/. Luettu 7.2.2025.

Gehani, V. 2022. React Native Chat Application using Firebase Hooks – Part 1. Medium. Luettavissa: https://medium.com/enappd/react-native-chat-application-using-firebase-hooks-part-1-8f5c27e23af7. Luettu 3.2.2025.

Figma s.a. What is Figma? Luettavissa: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma. Luettu 19.1.2025.

Firebase 2025a. Cloud Firestore. Luettavissa: https://firebase.google.com/docs/firestore. Luettu 22.1.2025.

Firebase 2025b. Firebase Authentication. Luettavissa: https://firebase.google.com/docs/auth. Luettu 30.1.2025.

Firebase 2025c. Writing conditions for Cloud Firestore Security Rules. Luettavissa: https://firebase.google.com/docs/firestore/security/rules-conditions. Luettu 22.1.2025.

Firebase 2025d. Managing users in Firebase. Luettavissa: https://firebase.google.com/docs/auth/web/manage-users. Luettu 14.1.2025

Firebase 2025e. Pricing plans. Luettavissa: https://firebase.google.com/pricing. Luettu 24.2.2025.

Flovi. s.a. Flovi Palvelut. Luettavissa: https://flovi.io/fi. Luettu 14.1.2025.

Git. 2025. Reference – Git documentation. Luettavissa: https://git-scm.com/docs. Luettu 07.02.2025.

GitHub. 2025. About READMEs. Luettavissa: https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-readmes. Luettu 12.2.2025.

Lempinen, T. 2025. Vuokra-autojätti antaa nyt autojaan ilmaiseksi käyttöön useissa eri Pohjoismaissa. Ilta-Sanomat. Luettavissa: https://www.is.fi/autot/art-2000010961682.html. Luettu 14.1.2025.

Mapbox. s.a. Mapbox Documentation. Luettavissa: https://docs.mapbox.com/. Luettu 01.02.2025.

Movacar. s.a. Rental car for €1. Saatavilla: https://www.movacar.com. Luettu 26.2.2025.

Parsonen, V. 2024. Kaksi eri maailmaa: vesiputous vs. ketterä kehitys. ERPWare. Luettavissa: https://erpware.fi/kaksi-eri-maailmaa-vesiputous-vs-ketterа-kehitys/. Luettu 14.1.2025.

React Native. 2025a. Core Components and APIs. Luettavissa: https://reactnative.dev/docs/components-and-apis. Luettu 20.1.2025.

React Native. 2025b. FlatList. Luettavissa: https://reactnative.dev/docs/flatlist. Luettu 23.1.2025.

React Native. 2025c. Introduction. Luettavissa: https://reactnative.dev/docs/getting-started. Luettu 24.1.2025

React Native. 2025d. Navigating Between Screens. Luettavissa: https://reactnative.dev/docs/navigation. Luettu 24.1.2025

React Native. 2025e. StyleSheet. Luettavissa: https://reactnative.dev/docs/stylesheet. Luettu 21.1.2025

RNFirebase. 2023. Cloud Firestore. Luettavissa: https://rnfirebase.io/firestore/usage. Luettu 20.1.2025

Transfercar. s.a. What is Transfercar? Luettavissa: https://www.transfercarus.com/about/how-does-it-work.html. Luettu 26.2.2025.