Niko Hämäläinen

VUOKRAUKSIEN HALLINTASOVELLUKSEN TIETOKONEVERSION KEHITTÄMINEN JA KÄYTTÖÖNOTON VALMISTELU

VUOKRAUKSIEN HALLINTASOVELLUKSEN TIETOKONEVERSION KEHITTÄMINEN JA KÄYTTÖÖNOTON VALMISTELU

Niko Hämäläinen
Opinnäytetyö
Kevät 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Niko Hämäläinen
Opinnäytetyön nimi: Vuokrauksien hallintasovelluksen tietokoneversion kehittäminen ja käyttöönoton valmistelu
Työn ohjaaja: Lasse Haverinen
Työn valmistumislukukausi ja -vuosi: 2021
Sivumäärä: 27 + 1 Liite

Opinnäytetyön tavoitteena oli kehittää tietokoneversio 2020 syksyllä aloitetun yritysprojektin aikana tehdystä mobiilisovelluksesta ja valmistella molemmat sovellukset käyttöönottoa varten testaamalla ne, jotta sovellukset toimivat odotetusti, ja valmistella sovellukset jaettaviksi käyttäjille.

Tietokoneversio kehitettiin Windows-käyttöjärjestelmälle WPF-sovelluksena käyttäen C#-ohjelmointikieltä ja tietokantana toimi Google Firebase -palvelun Firestore. Mobiilisovellus valmisteltiin jaettavaksi Google Play Console -palvelun avulla ja tietokoneversio Dropbox-palvelun avulla.

Lopputuloksena oli käyttövalmis tietokoneversio, hyvin testattu mobiili- ja tietokonesovellus ja jakelukanavat, joiden avulla sovellukset voitiin jakaa käyttäjille.

Asiasanat: mobiilisovellus, ohjelmointi, tietokanta, testaus

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Niko Hämäläinen
Title of thesis: Rent infromation applications computer version development and preparations for use
Supervisor: Lasse Haverinen
Term and year when the thesis was submitted: 2021
Pages: 27+ 1 Appendix

The purpose of this thesis was to develop a desktop application from an existing mobile application which development started during autumn of 2020, test both applications so they work as expected and prepare them to be distributed to users.

The desktop application was developed for windows operating system as a WPF application using C# programming language and Google Firestore was used as the database for the project. The mobile application was prepared for distribution using Google Play Console and the desktop application was distributed using Dropbox.

The result was a finished desktop application, well tested mobile and desktop applications and ready to go distribution methods which could be used to share the applications to users.

Keywords: mobile application, programming, database, testing

SISÄLLYS

LIITTEET

Liite 1 Mobiilisovelluksen päätoiminnallisuudet

SANASTO

Android	Android on mobiililaitteille suunniteltu käyttöjärjestelmä
APK	Android-sovellusohjelmien pakettitiedosto, joka sisältää ohjelman ja sen tarvitsemat tiedostot.
GUI	Graphical User Interface, graafinen käyttöliittymä
NoSQL-tietokanta	Perinteisestä relaatiomallista poikkeava tietokanta
NuGet	Microsoftin pakettienhallintaohjelma, jonka avulla voidaan luoda, jakaa ja käyttää ohjelmapaketteja.
UID	Unique identifier
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

1 JOHDANTO

Työn toimeksiantaja toimi pieni Keski-Pohjanmaalla sijaitseva rakennusalan yritys, joka myös vuokrasi rakennusalan tarvikkeita ja palveluita. Sovelluksen päätarkoituksena oli helpottaa vuokrauksiin liittyvien tietojen tallettamista ja sovelluksen oli tarkoitus tulla pelkästään yrityksen työntekijöiden käyttöön.

Projekti aloitettiin 2020 syksyllä yritysprojektina, jonka tarkoituksena oli kehittää mobiili- ja tietokonesovellus, joiden avulla voitiin seurata varastossa olevien tuotteiden tietoja, vuokrauksien tietoja ja asiakastietoja. Mobiiliversio kehitettiin Android-puhelimille käyttäen Java-ohjelmointikieltä ja tietokantana toimi Google-Firebase -palvelun Firestore. Mutta yritysprojektille varattu aika ei riittänyt viimeistelemään projektia, joten tietokoneversion kehittäminen, sovelluksien testaaminen ja jakelukanavien valmistelu päätettiin suorittaa opinnäytetyönä.

Työn tehtävänä oli kehittää aiemmin tehdystä mobiiliversiosta tietokoneversio, testata molempien versioiden toiminallisuus ja valmistella jakelukanavat, jotta sovellukset saatiin jaettua käyttäjille nopeasti ja helposti.

Toimeksiantajan kriteerit tietokoneversiolle olivat, että sen tulisi sisältää samat päätoiminnot kuin aiemmin kehitetyllä mobiilisovelluksella ja että se toimisi Windows-käyttöjärjestelmällä. Tavoitteena oli myös parantaa muutamien osioiden käytettävyyttä, jotka olivat mobiilisovelluksella hieman vaikeasti käytettävissä puhelimien pienellä näytöllä ja näppäimistöllä.

Opinnäytetyössä tarkastellaan käytettyjä ohjelmistoja, tietokoneversion toteutusta, sovelluksien jakamistapoja ja toimenpiteitä, joiden avulla sovellukset ovat testattu käyttöönottoa varten.

2 PROJEKTISSA KÄYTETYT OHJELMISTOT JA PALVELUT

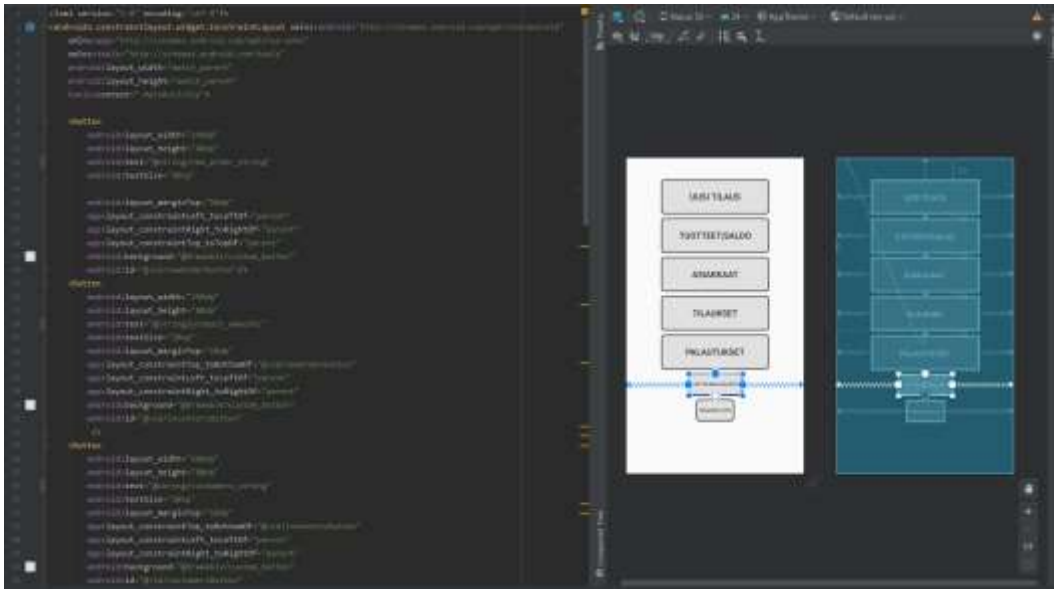
Mobiilisovellus kehitettiin Android-puhelimille käyttäen Android Studio -ohjelmointiympäristöä ja Java-ohjelmointikieltä. Tietokoneversio kehitettiin Windows-käyttöjärjestelmälle käyttäen Visual Studio -ohjelmointiympäristöä, WPF-käyttöliittymäkehystä ja C#-ohjelmointikieltä. Tietokantana sovelluksille toimi Google Firestore -tietokanta, mobiilisovellus valmisteltiin jaettavaksi Google Play Console -palvelun avulla ja tietokoneversio Dropbox-pilvitallennuspalvelun avulla.

2.1 Android Studio -ohjelmointiympäristö

Android Studio on virallinen Googlen ohjelmointiympäristö Android-sovelluksien kehittämiseen ja se pohjautuu JetBrains' IntelliJ IDEA ympäristöön. Android Studio tarjoaa yhtenäisen alustan, jolla voidaan kehittää sovelluksia kaikille Android-laitteille (1).

2.2 XML-kieli

XML on tekstimuotoinen standardoitu viestin muodostustapa, joka välittää tiedon avainarvo-pareista muodostettuina elementteinä (2). Projektissa XML-kieltä käytettiin Android-sovelluksen käyttöliittymän luontiin (kuva 1).



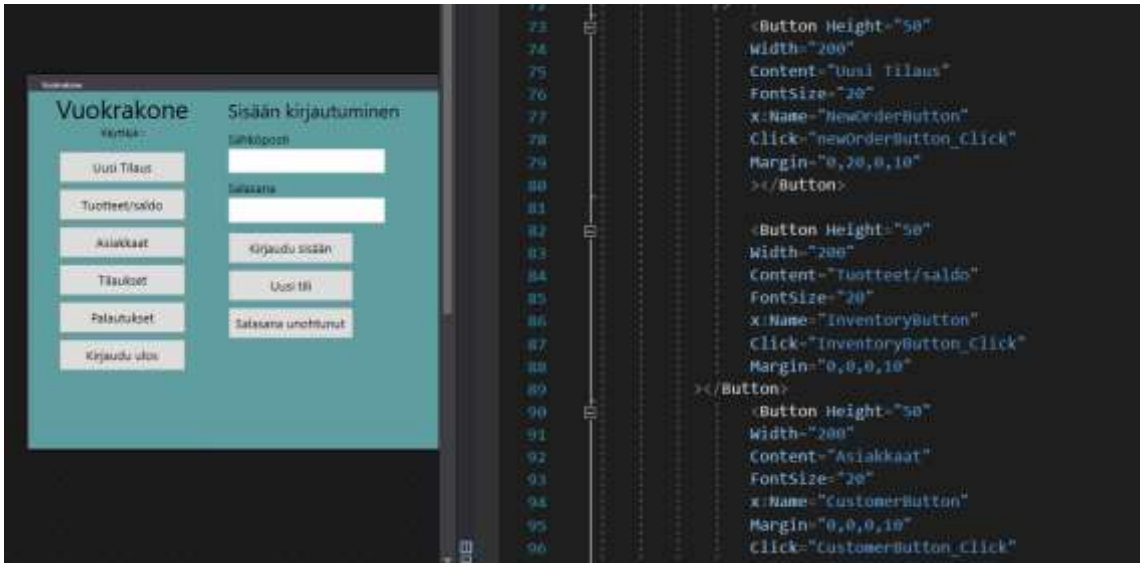
KUVA 1. Esimerkki XML-rakenteesta ja mobiilisovelluksen päävalikosta Android Studio -ohjelmointiympäristössä.

2.3 WPF- käyttöliittymäkehys

WPF eli Windows Presentation Foundation on Microsoft yrityksen kehittämä avoimen lähdekoodin GUI-rakenne, jossa käytetään XAML-kieltä käyttöliittymän luomiseen. WPF on osa Windowsin .NET-ympäristöä (3).

2.4 XAML-kieli

XAML on Microsoft yrityksen kehittämä XML-pohjainen ohjelmointikieli rakenteellisten arvojen ja olioiden alustamiseen (kuva 2). Sitä käytetään .NET Framework 3.0- ja 4.0 -teknologioiden kanssa, joihin kuuluu esimerkiksi WPF- ja UWP-sovellukset (4).



KUVA 2. Esimerkki XAML-rakenteesta WPF-sovelluksen käyttöliittymästä Visual Studio -ohjelmointiympäristössä.

2.5 Google Firebase-palvelu ja Firestore-tietokanta

Google Firestore on pilvipohjainen noSQL-tietokanta, joka on osa Googlen tarjoamaa Firebase-palvelua (5). Firestore-tietokannan rakenne koostuu kokoelmista ja dokumenteista. Kokoelmat sisältävät dokumentteja ja dokumentit sisältävät tallennetun tiedon kentissä avain-arvopareina (6).

Firebase-palvelu on suunniteltu mobiili- ja websovellusten kehittämiseen. Palveluun voidaan luoda projekti, joka sisältää useita ominaisuuksia, kuten edellä mainitun Firestore-tietokannan, helpon käyttäjien autentikoinnin Firebase-palveluun, Test Lab -palvelun mobiilisovelluksien testaamiseen, App Distribution -palvelun mobiilisovelluksien jakamiseen ja muutamia muita ominaisuuksia. Palvelua voi käyttää ilmaiseksi mutta joitain sen ominaisuuksia ja käyttöä on rajoitettu (7). Esimerkiksi Firestore-tietokannan tallennustila on rajoitettu yhteen gigatavuun, dokumentin lukukerrat ovat rajoitettu 50 000 kertaan päivässä ja dokumenttien lisääminen ja poistaminen ovat rajoitettu 20 000 kertaan päivässä.

2.6 Google Play Console

Google Play Console on suunniteltu mobiilisovelluksien julkaisemiseen ja kehittämiseen Google Play -kauppaan, kun Google on tarkistanut ja hyväksynyt sovelluksen (8). Tarjolla on myös kehittäjille useita työkaluja, joiden avulla sovelluksia voidaan testata, esimerkiksi pilvessä tehtävillä testeillä tai jakamalla testattava sovellus käyttäjille testikanavien avulla.

2.7 Dropbox-pilvitallennuspalvelu

Dropbox on pilvitallennuspalvelu, jonka avulla tiedostot voidaan synkronoida usealla eri laitteella. Dropbox-palvelun tiedostoja ja kansioita voidaan myös jakaa linkkien välityksellä (9).

3 TYÖN TOTEUTUS

3.1 Tietokoneversion toteutus

Tietokoneversion toteutukseen harkittiin JavaFX-ohjelmistoalustaa ja WPF-käyttöliittymäkehystä. Molemmat vaihtoehdot tarjosivat hyvät työkalut työpöytäsovelluksien kehittämiseen. Sovelluksen olisi voinut toteuttaa kummalla tahansa valinnalla, mutta kumpikaan vaihtoehto ei ollut ylivoimaisesti parempi tavalla tai toisella, joten päätettiin valita tapa, joka tutkimisen jälkeen vaikutti nopeammalta ja helpommalta tavalta kehittää sovellus. Lopputuloksena tietokoneversio päätettiin toteuttaa WPF-sovelluksena. Sovelluksessa käytettiin Firebase-palvelun tarjoamaa käyttäjien autentikointia käyttäjätilien luomiseen ja Firestore-tietokantaa käytettiin vuokrauksiin liittyvien tietojen sekä käyttäjien tietojen tallettamiseen.

Sovelluksessa käytetty Firestore-tietokanta koostui neljästä eri kokoelmasta, jotka olivat käyttäjät, asiakkaat, tuotteet ja tilaukset. Käyttäjät-kokoelman dokumentit sisälsivät käyttäjien tiedot, kuten nimen, roolin ja sähköpostin. Asiakkaat-kokoelman dokumentit sisälsivät asiakkaiden yhteys- ja osoitetiedot. Tuotteet-kokoelman dokumentit sisälsivät vuokrattavien tuotteiden tiedot kuten hinnan, painon ja määrän varastossa. Tilaukset-kokoelman dokumentit sisälsivät vuokraustilauksien tiedot, kuten asiakkaan yhteystiedot, vuokrattujen tuotteiden tiedot ja muita tilaukseen liittyviä tietoja. Tarkat tiedot jokaisen kokoelman dokumenttien rakenteesta löytyvät liitteessä 1.

3.1.1 **Sovelluksen osiot ja niiden toiminnallisuudet**

Sovellus koostui seitsemästä osiosta (kuva 3). Sovelluksen aloitusnäkymänä oli sisäänkirjautuminen-osio, jossa pystyttiin luomaan uusi käyttäjätili Firebase-palveluun, uusimaan salasana sähköpostin avulla ja kirjautumaan sisään syöttämällä olemassa olevan tilin sähköpostiosoite ja salasana.

Päänäkymä sisälsi napit jokaiselle sovelluksen osiolle sekä uloskirjautumiselle. Päänäkymän avulla navigoitiin sovelluksen eri osioiden välillä.

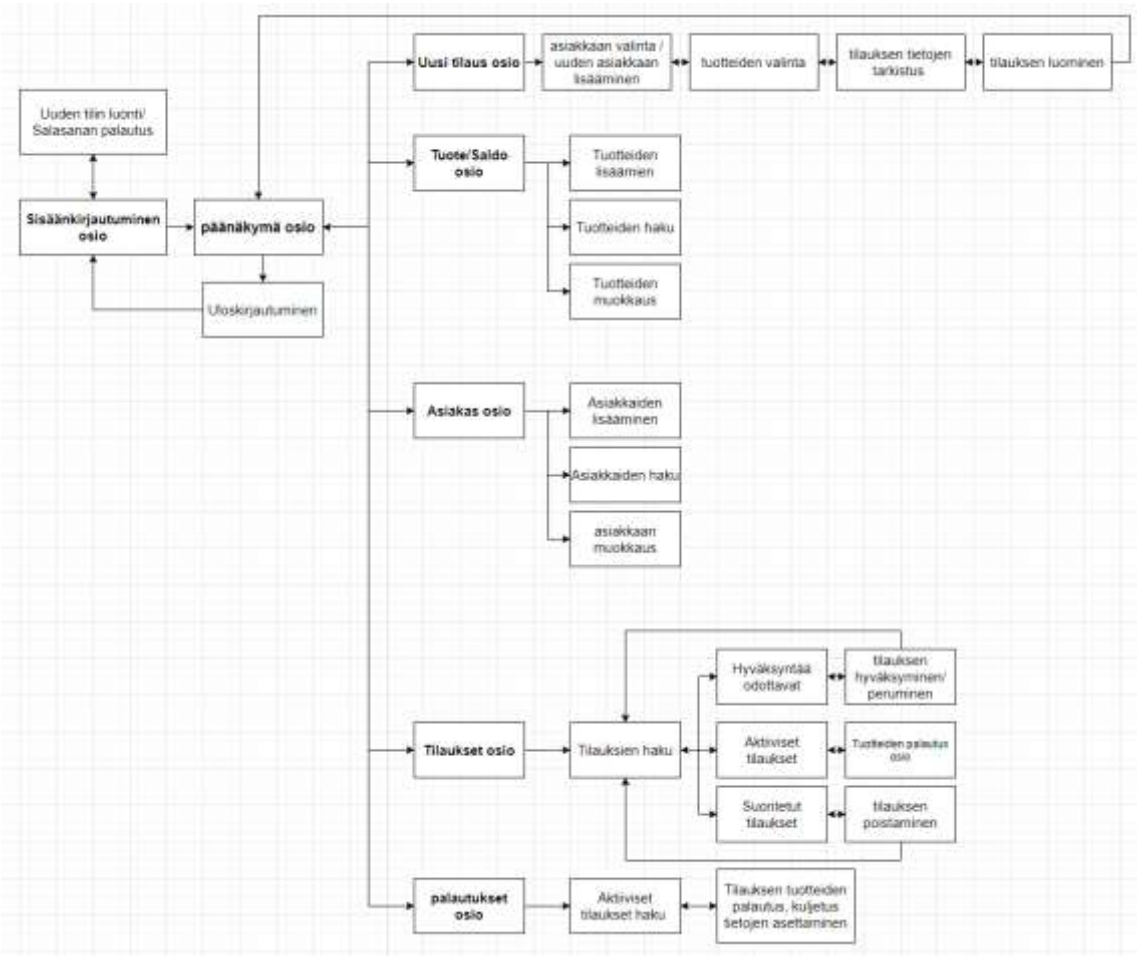
Uusi tilaus -osio koostui kolmesta eri näkymästä, asiakkaan valinnasta, vuokrattavien tuotteiden valinnasta ja yhteenvedosta. Asiakkaan valinnassa joko valittiin tietokannassa olevan asiakkaan tiedot, muokattiin olemassa olevan asiakkaan tietoja tai lisättiin uuden asiakkaan tiedot ja edettiin valitsemaan tuotteita, joita oli mahdollista hakea kategorioiden perusteella tai tuotekoodin perusteella. Valitut tuotteet lisättiin ostoskoriin ja tuotteiden valinnan jälkeen edettiin yhteenvetoon, jossa tarkistettiin valitut tuotteet, tilauksen osoitetiedot, asennusaika, asiakaskohtainen alennus, ja lopuksi tilauksen luominen hyväksyttiin, jossa tuotteiden määrät vähennettiin tietokannasta ja tilauksen tiedot lisättiin tietokantaan.

Tuote/saldo-osio sisälsi toiminnallisuudet, joilla lisätään uusia tuotteita tietokantaan, muokataan sekä haetaan tietokannassa olevien tuotteiden tietoja ja poistetaan tietokannassa olevia tuotteita.

Asiakkaat-osio sisälsi toiminnallisuudet, joilla lisätään uusia asiakastietoja tietokantaan, muokataan sekä haetaan tietokannassa olevia asiakastietoja ja poistetaan asiakastietoja tietokannasta.

Tilaukset-osion toiminnallisuuksiin kuului tilauksien haku useilla eri vaihtoehdoilla, luotujen tilauksien hyväksyminen tai peruminen, tilauksien tietojen näyttäminen ja mahdollisuus poistaa suoritettuja tilauksia.

Palautukset-osio sisälsi toiminnallisuudet, joilla haettiin aktiiviset tilaukset ja tilausten tuotteet palautettiin. Uutena toiminnallisuutena osioon lisättiin myös mahdollisuus asettaa tilaukselle kuljetuksen tiedot, joka sisälsi kuljettajan nimen, kuljetukseen kuluneen ajan ja matkan. Kun valitun tilauksen tuote palautetaan, tallennetaan tietokantaan tilauksen dokumenttiin palautuksen ajankohta ja vastaanottajan nimi. Kun kaikki tilauksen tuotteet ovat palautettu ja kuljetuksen tiedot ovat tallennettu tilauksen dokumenttiin, asetetaan tila suorituksi.



KUVA 3. Sovelluksen osiot ja niiden toiminnot

3.1.2 **Firebase-palvelun käyttöönotto ja sisäänkirjautuminen**

Jotta sovelluksella pystyttiin käyttämään Firebase-palvelun ominaisuuksia, sovellukseen lisättiin FirebaseAdmin NuGet -paketti, jonka avulla sovellukseen integroitiin Firebase-palvelu. Paketin kirjastojen avulla pystyttiin tekemään pyyntöjä Firestore-tietokantaan ja luomaan uusia käyttäjätilejä. Projektiin lisättiin ympäristömuuttujaksi Firebase service account -tilin JSON-tiedosto, jonka avulla sovelluksen pyynnöt autentikoitiin (10).

Sovelluksessa käyttäjätilien luomiseen käytettiin Firebase-palvelun sähköposti autentikointia, jonka avulla käyttäjille luotiin sähköpostin ja salasanan avulla käyttäjätili Firebase-palveluun. Firestore-tietokantaan myös lisättiin käyttäjät-kokoelmaan dokumentti, joka sisälsi käyttäjän roolin, nimen, sähköpostin ja dokumentin nimi oli käyttäjätilin UID-koodi.

FirebaseAdmin-paketti ei sisältänyt toimintoa, jonka avulla Firebase-palvelusta olisi voinut tarkistaa, vastaako sisäänkirjautumisessa syötetty sähköposti ja salasana olemassa olevaa käyttäjätiliä. Siksi päädyttiin käyttämään FirebaseAuthentication.net NuGet -pakettia, joka on tarkoitettu Firebase-palvelun käyttäjien autentikointiin Firebase REST -rajapinnan avulla. Paketin kirjastojen avulla lähetettiin Firebase REST -rajapintaan POST-pyyntö sisäänkirjautumiselle sähköpostilla ja salasanalla. Jos kirjautumistiedot vastasivat olemassa olevaa Firebase-käyttäjätiliä, REST-rajapinta palautti Firebase-käyttäjätilin tiedot ja käyttäjä kirjattiin sovellukseen sisään. Firestore-tietokannasta myös noudettiin käyttäjätilin UID-koodin avulla käyttäjän dokumentti, joka sisälsi roolin ja nimen.

3.1.3 **Firestore-tietokannan säännöt ja roolit**

Firestore-tietokantaan pystyttiin asettamaan sääntöjä (kuva 4), joiden avulla voitiin rajoittaa esimerkiksi, kuka voi poistaa tietystä kokoelmasta dokumentteja. Niiden avulla tarkistetaan pyyntöä tekevän Firebase-käyttäjätilin rooli viittaamalla käyttäjän UID-koodin avulla dokumenttiin, joka sisälsi käyttäjän roolin.

```
    match /Customers/{customer} {
allow create,read,update: if
get(/databases/$(database)/documents/Users/$(request.auth.uid)).data.clearance == 'user'
```

KUVA 4. Firestore-sääntö Customers-kokoelmalle, jossa sallitaan kokoelman dokumenttien luominen, lukeminen ja päivittäminen, jos käyttäjätilin UID-koodia vastaavan dokumentin clearance-kentän arvo on user.

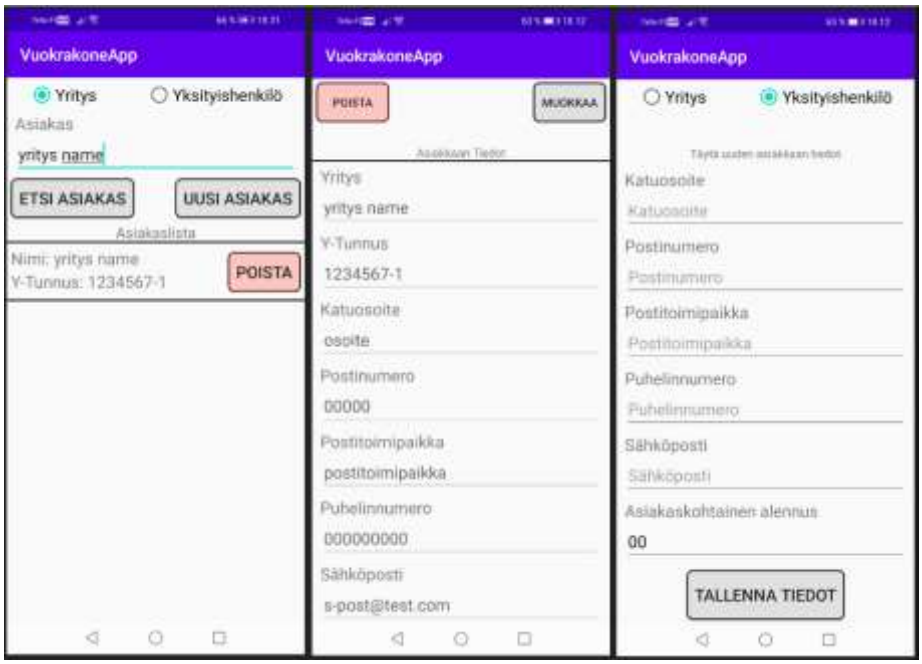
Projektin Firestore-tietokannassa oli asetettuna kolme roolia, dummyUser, user ja admin. DummyUser asetettiin uusille käyttäjille ja roolilla ei ollut oikeuksia lukea tai tehdä muutoksia Firestore-tietokantaan. User-roolilla oli oikeus päivittää ja luoda dokumentteja ja lukea tietokantaa. Admin-roolilla pystyttiin lukemaan, kirjoittamaan ja poistamaan tietoja tietokannasta sekä muuttamaan ei-admin-käyttäjien rooleja mobiilisovelluksen avulla.

Koska tietokoneversiossa pyynnöt autentikoitiin Firebase service account -tilin avulla, nämä säännöt eivät päteneet tietokoneversiolla tehtyihin pyyntöihin, joten

sisäänkirjautumisen yhteydessä noudettavan roolin mukaan itse sovelluksen toiminnallisuuksia rajoitettiin.

3.1.4 **Käyttöliittymän suunnittelu ja toteutus**

Käyttöliittymän suunnittelussa tavoitteena oli luoda käyttöliittymä, joka olisi mahdollisimman samantyyppinen mobiilisovelluksen, jotta käyttäjien olisi helppo oppia molempien sovelluksien käyttäminen. Mobiilisovelluksen eri osiot koostuivat useista eri näkymistä rajoitetun näytön koon vuoksi. Esimerkiksi mobiilisovelluksen asiakkaat-osion haku, muokkaus ja uuden asiakkaan lisääminen jaettiin kolmeen eri näkymään (kuva 5).



KUVA 5. Mobiilisovelluksen asiakkaat-osion haku-, asiakkaan tiedot- ja uusi asiakas -näkymät

Tietokoneversion käyttöliittymä jaettiin neljään eri sarakkeeseen XAML-kielen Grid-elementin avulla. Käyttöliittymässä oli koko ajan näkyvissä päävalikko ensimmäisessä sarakkeessa, jolla navigoitiin eri osioiden välillä. Lopuille kolmelle sarakkeelle asetettiin Content Control -näkymä. Kun jonkin osion nappia painettiin, asetettiin Content Control -näkymään valitun osion käyttöliittymä (kuva 6). Näkymän kolmen sarakkeen avulla jäljitellään mobiilisovelluksen eri näkymiä ja toiminnallisuuksia.

KUVA 6. Tietokoneversion asiakkaat-osion näkymä

3.1.5 **Tiedon haku ja käsittely**

Firestore-tietokannasta dokumentit noudettiin rakentamalla Query, johon asetettiin ehtoolauseita, joiden perusteella kokoelmasta noudettiin sitä vastaavat dokumentit (kuva 7).

```
CollectionReference CustomerRef = db.Collection("Customers");
Query query = CustomerRef.WhereEqualTo("business", true);
```

KUVA 7. Query jonka avulla haetaan Customers-kokoelmasta dokumentit joissa business-kenttä on true.

Sovellus sisälsi jokaiselle Firestore-tietokannan kokoelmalle oman luokkansa (kuva 8). Kun Firestore-tietokannasta noudettiin kokoelmasta dokumentti tai dokumentteja, luotiin jokaiselle dokumentille instanssi sitä vastaavasta luokasta, jonka muuttujiin asetettiin dokumentin sisältämät tiedot. Noudetuista dokumenteista luodut instanssit lisättiin listaan ja se näytettiin käyttöliittymässä ListView-komponentissa. Listasta voitiin valita dokumentti tuplaklikkaamalla sitä ja osion mukaan dokumentin tiedot asetettiin muokattaviin tekstikenttiin, kuten kuvassa 5, tai sen tiedot asetettiin näkyviin tekstinä.

KUVA 8. Firestore-tietokannassa oleva Customers-kokoelman dokumentti ja sovelluksen CustomerObject luokka

Sovelluksessa Firestore-tietokantaan uudet dokumentit lähetettiin dictionary muodossa, joka koostui avain-arvopareista (kuva 9). Myös dokumenttien muokkaaminen suoritettiin luomalla dictionary muuttuja, joka koostui Firestore-tietokannan dokumentin kenttien nimistä ja niille asetettavista uusista arvoista.

```
Google.Cloud.Firestore.DocumentReference docRef2 = db.Collection("Products").Document(barcode);
Dictionary<string, object> product = new Dictionary<string, object>
{
{ "amount", amount },
{ "totalAmount", totalAmount},
{ "category", category},
{ "name", name},
{ "price", price},
{ "weight", weight}

};
 await docRef2.SetAsync(product);
```

KUVA 9. Firestore-tietokannan tuotteet-kokoelmaan tallennettavan tuotteen dictionary

3.2 Sovelluksien testaaminen

Sovelluksien testaamista suoritetaan hyvin paljon kehittämisen yhteydessä, mutta kaikkia virheitä ei aina löydetä, joten sovelluksen tarkka testaaminen eri menetelmien avulla on hyvin tärkeää, jotta sovelluksen toimivuudesta voidaan olla varmoja ennen käyttöönottoa. Sovelluksien testaamisen tarkoituksena oli löytää virheelliset toiminnallisuudet tai puutteet ja korjata ne.

Testaus vaiheessa painotettiin mobiilisovellusta, koska tietokoneversio tulisi pelkästään käyttöön yhdelle tietokoneelle, toisin kuin mobiiliversio, joka tulisi käyttöön usealle eri puhelinmallille.

3.2.1 **Käytetyt testausmenetelmät**

Manuaalinen testaaminen on menetelmä, jossa sovelluksia ja niiden toimintaa testataan ilman automatisoituja työkaluja loppukäyttäjän näkökulmasta. Menetelmässä luodaan testaussuunnitelma, joka sisältää sovelluksen testattavia käyttötapauksia (11). Esimerkiksi käyttötapaus voi olla sisäänkirjautumisen yhteydessä, toimiiko sovellus odotetusti, jos käyttäjätunnus-kenttä jätetään tyhjäksi ja painetaan kirjaudu sisään -painiketta.

Tietokoneversion testaamiseen oli haastavaa löytää työkalua tai testaus menetelmää, joka olisi ollut tehokkaampi kuin manuaalinen testaus, koska useimmat ongelmat pystyttiin havaitsemaan käyttöliittymää käyttäessä. Myös mobiilisovelluksen testaamisessa päädyttiin manuaaliseen testaamiseen, mutta apuna käytettiin myös Firebase-palvelun tarjoamaa Test Lab ominaisuutta, jonka avulla sovellusta pystyttiin testaamaan usealla eri puhelimella.

3.2.2 **Manuaalisen testaamisen toteutus**

Ennen testaamisen suorittamista sovelluksille luotiin testaussuunnitelmat, joissa oli lajiteltuna eri osiot, niiden käyttötapaukset sekä ohjeistukset testien suorittamiseen ja odotetut tulokset käyttötapauksille. Testaaminen suoritettiin käymällä läpi kaikki osion käyttötapaukset, jonka jälkeen edettiin seuraavaan osioon. Tes-

taamisen aikana ilmenneet puutteet ja ongelmat otettiin ylös, jotta ne voitiin korjata testaamisen jälkeen. Kun testaaminen oli suoritettu, korjattiin löydetyt puutteet ja aloitettiin testaaminen alusta.

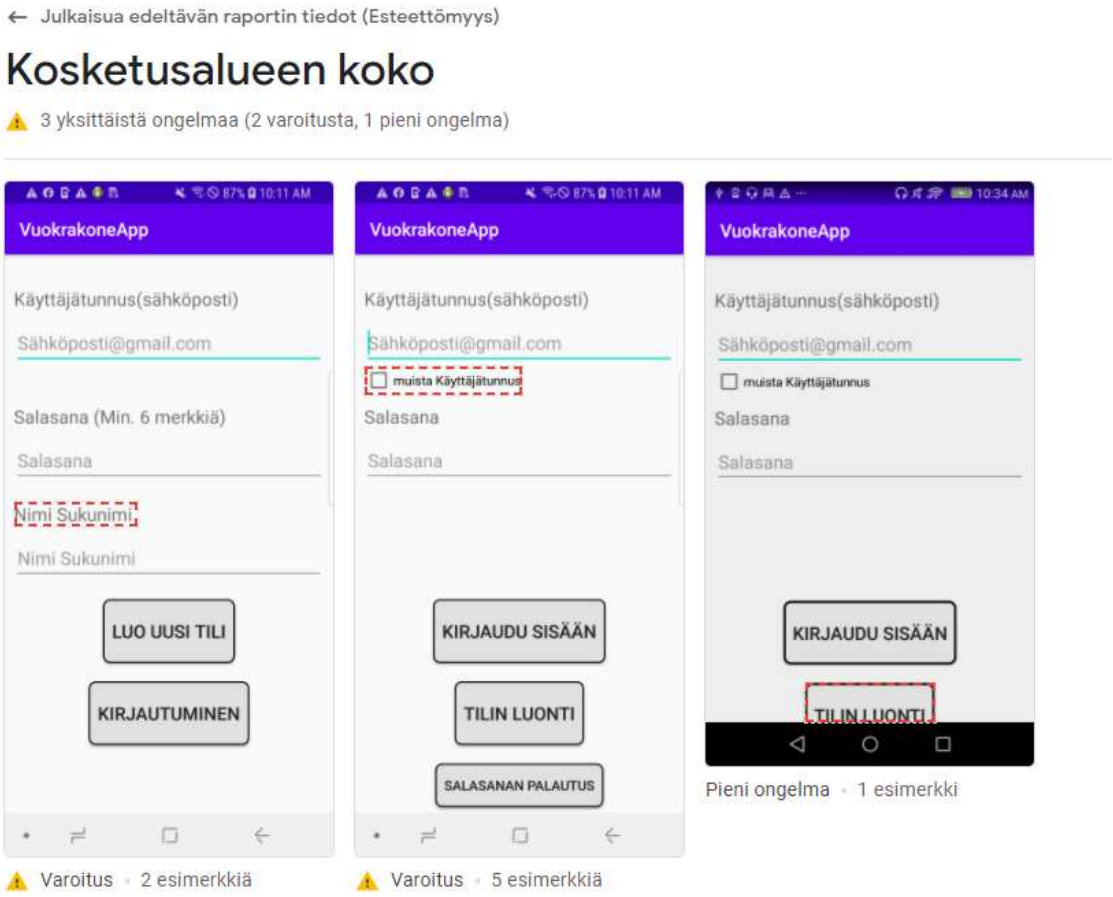
3.2.3 **Firebase Testlab**

Test Lab on ilmainen Firebase-palvelussa, mutta sen käyttö on rajoitettu kymmeneen testiin päivässä virtuaalisilla laitteilla ja viiteen testiin fyysisillä laitteilla. Testeihin voidaan valita useita eri puhelimia, jotta sovelluksen voi testata mahdollisimman laajalla valikoimalla. Google Play Console -palvelun avulla on myös mahdollisuus testata sovelluksia, sillä se käyttää myös Test Lab -palvelua, mutta sen laitevalikoima on huonompi ja testit ajetaan vain, kun sovelluksesta ladataan uusi versio Google Play Console -palvelun testikanavalle.

Testi aloitettiin lataamalla sovelluksen APK-tiedosto Test Lab -palveluun, jonka jälkeen testille valitaan puhelimet, joilla testit ajetaan. Testeille on myös vaihtoehtona asettaa aikaraja, sisäänkirjautumistiedot ja kolme deep link -osoitetta, joihin testi siirtyisi käynnistyksen jälkeen, ja testiin voidaan myös asettaa manuaalisesti nappien painalluksia ja tekstikenttien täyttöä syöttämällä, millä nimellä käyttöliittymä elementit ovat nimetty sovelluksen XML-tiedostoissa.

Kun testit ovat valmistuneet, löytyy valituille puhelimille raportit, jotka sisältävät testien keräämät tiedot, kuten mahdolliset käyttöliittymä ongelmat (kuva 10), tietoa sovelluksen suorituskyvystä sekä muista mahdollisista ongelmista.

Tässä projektissa testejä käytettiin pääasiallisesti eri puhelimien käyttöliittymä ongelmien löytämiseen ajamalla testejä, joille asetettiin käyttäjäntunnukset, ja muutamissa testeissä asetettiin näppäimien painalluksia, joiden avulla testit ohjattiin tiettyihin osioihin ja testattiin sovelluksen toiminnallisuuksia.



KUVA 10. Test Lab -testin löytämät mahdolliset käyttöliittymä ongelmat

3.2.4 **Testaamisen tulokset**

Testaamisen yhteydessä ei löytynyt suuria sovellusta kaatavia ongelmia, mutta ilmeni useita puutteita ja epähuomioita, jotka oli hyvä huomata ja korjata ennen sovelluksen käyttöönottoa.

Manuaalisen testaamisen yhteydessä löytyi korjattavia puutteita, kuten kirjoitusvirheitä, puutteellisia tai puuttuvia ilmoituksia tiettyjen toimintojen kohdalla. Firestore-tietokannan indeksointi oli puutteellinen eli tilaukset-osion tietyillä hakuvaihtoehdoilla ei löytynyt tuloksia, vaikka olisi pitänyt löytyä. Sovellukset myös sisälsivät mahdollisuuksia asettaa merkkejä tekstikenttiin, joita ei olisi pitänyt olla mahdollista syöttää. Manuaalinen testaaminen oli hyvä tapa testata toiminnallisuutta sekä käyttöliittymää, mutta se oli hyvin aikaa vievä prosessi.

Test Lab -testien yhteydessä ilmeni muutamia huomion arvoisia ongelmia käyttöliittymissä eri laitteilla. Muutamilla puhelinmalleilla ilmeni, että jotkin käyttöliittymäelementit saattavat jäädä näytön ulkopuolelle pienemmillä näytöillä (kuva 10). Testit myös ilmoittivat mahdollisista kontrastiongelmista, joissa teksti saattoi olla liian taustaan sekoittuva. Muita korjattuvia ongelmia ei palvelun avulla havaittu.

3.3 Sovelluksien jakaminen

3.3.1 **Tietokoneversion jakaminen**

Tietokoneversion jakaminen päätettiin toteuttaa jakamalla tiedosto pilvitallennuspalvelun avulla. Vaihtoehtoina harkittiin Dropbox-, Google Drive- ja OneDrive-palvelua. Vaikka edellä mainitut palvelut tarjosivat kaikki samat tarvittavat ominaisuudet sovelluksen jakamista varten, valittiin Dropbox-palvelu, koska se oli toimeksiantajalle ennestään tuttu palvelu.

WPF-projektin julkaisuversio pakattiin zip-tiedostoon, joka sisälsi pikakuvakkeen sovelluksen käynnistämistä varten. Zip-tiedosto tallennettiin Dropbox-kansioon, jonka linkki oli sovelluksen asennusohjeistuksessa. Sovelluksen lataaminen ja käyttöönottaminen oli hyvin helppoa. Zip-tiedosto ladattiin Dropbox-palvelusta, se purettiin valittuun sijaintiin tietokoneella, jonka jälkeen sovelluksen pystyi käynnistämään kansion sisällä olevasta pikakuvakkeesta.

3.3.2 **Mobiilisovelluksen jakelukanava**

Mobiilisovelluksen jakamiseen etsittiin jakelukanavaa, jonka avulla sovellus pystyttiin jakamaan pelkästään tietyille henkilöille, sovelluksen asentaminen olisi mahdollisimman helppoa käyttäjille ja sovelluksen päivittäminen kaikille käyttäjille olisi helppoa ja nopeaa.

Jakamiseen harkittiin muutamia jakelukanavia, kuten Huawei AppGallery, Amazon Appstore, Google Play tai Firebase App Distribution, vaikka edellä mainitut jakelukanavat täyttivät projektin tarpeet, valittiin Google Play, sillä se oli yleisin Android sovelluskauppa ja se oli esiasennettuna useimmille Android-puhelimille.

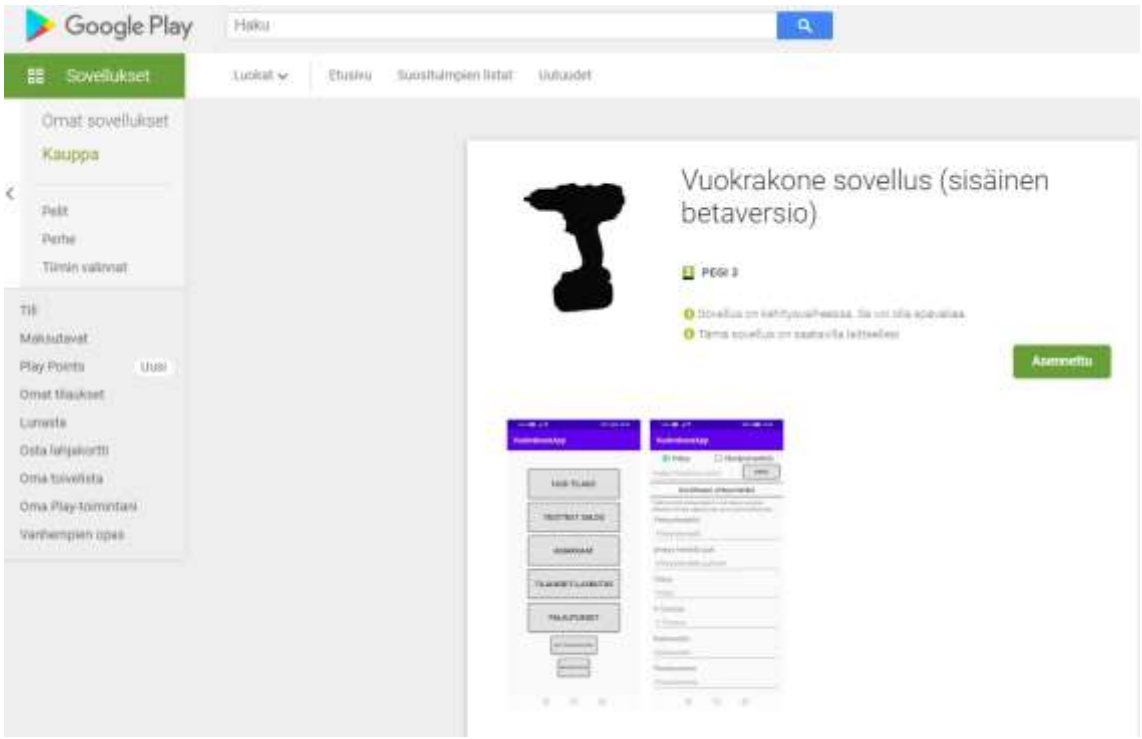
Sovelluksen APK ladattiin myös Dropbox-kansioon varmuuden vuoksi, jos käyttäjällä ei ollut mahdollisuutta ladata sovellusta Google Play -kaupan avulla. Linkki Dropbox-kansioon löytyi sovelluksen asennusohjeistuksesta.

3.3.3 **Google Play Console -testikanava**

Jotta sovellus pystyttiin lataamaan Google Play -kauppaan, täytyi käyttää Google Play Console -palvelua, joka taas vaati Google Play -kehittäjätilin. Sen luontiin täytyi luoda Google-tili tai käyttää olemassa olevaa tiliä, hyväksyä käyttöehtosopimus ja maksaa 25 dollaria. Tilin luonnin jälkeen oli mahdollista käyttää Google Play Console -palvelun ominaisuuksia.

Koska sovellusta ei haluttu julkaista julkiseksi sovellukseksi Google Play -kauppaan kaikkien ladattavaksi, vaihtoehtona oli kolme Google Play Console -palvelun tarjoamaa testikanavaa avoin, suljettu tai sisäinen kanava. Tässä päädyttiin valitsemaan sisäinen testaus, koska kanavalle ladattavat sovelluksen päivitykset tulivat saataville muutamassa minuutissa, kun taas muiden testikanavan päivittämiseen voi kulua huomattavasti kauemmin.

Testikanavalle lisättiin käyttäjiä luomalla sähköpostilista, johon lisättiin käyttäjien Google-tilin sähköposti. Kun sähköposti oli lisätty, käyttäjä pystyi liittymään testaukseen linkin kautta ja lataamaan sovelluksen Google Play -kaupasta (kuva 11).



KUVA 11. Sisäisen testikanava sovelluksen näkymä Google Play -kaupassa

4 YHTEENVETO

Työn tarkoituksena oli kehittää tietokoneversio aikaisemmin kehitetystä Android-mobiilisovelluksesta Windows-käyttöjärjestelmälle, testata molemmat mobiili- sekä tietokoneversio, jotta voitiin olla varmoja sovelluksien toimivuudesta, ja valmistella tavat jakaa sovellus käyttäjille. Lopputuloksena oli tietokoneversio, johon saatiin toteutettua samat pääominaisuudet kuin aikaisemmin kehitettyyn mobiilisovellukseen, ja se toimi Windows-käyttöjärjestelmällä työpöytäsovelluksena. Siinä myös helpotettiin tiettyjä toimintoja verrattuna mobiilisovellukseen, kuten uusien tuotteiden ja asiakkaiden tietojen lisäämistä. Molemmat versiot testattiin ja löydetyt puutteet saatiin korjattua ongelmitta. Lopuksi kumpikin sovellus saatiin valmisteltua helposti jaettavaksi käyttöönottoa varten.

Sovelluksen avulla saatiin helpotettua toimeksiantajan vuokrauksien tilauksiin liittyvien, asiakkaiden ja vuokrattavien tuotteiden tietojen tallettamista sekä seuraamista.

Tietokoneversion toteuttaminen oli helppo aloittaa, koska sovelluksen toiminnallisuudet olivat jo tiedossa, mutta XAML-kielen ja WPF-sovelluksen käytänteiden opettelu tarjosi paljon haasteita kehittämisen aikana.

Manuaalinen testaaminen oli hyvin aikaa kuluttava prosessi tehdä, mutta sen avulla löytyivät helposti puutteet ja virheelliset toiminnallisuudet. Manuaalinen testaaminen pakotti myös analysoimaan sovellusta huomattavasti tarkemmin kuin kehityksen aikana tehdyn testaamisen aikana, jotta pienetkin virheet löytyivät kuten kirjoitusvirheet. Mobiilisovelluksen testaamiseen käytetty Test Lab oli myös hyvin hyödyllinen ja antoi hyvää tietoa mahdollisista ongelmista eri puhelimilla, joita ei manuaalisen testaamisen aikana havaittu. Sovelluksien testaamiseen harkittiin myös yksikkötestaamista, mutta sen avulla sovelluksen kattava testaaminen osoittautui hyvin työlääksi toteuttaa verrattuna manuaaliseen testaamiseen.

Tietokoneversion jakaminen haluttiin tehdä mahdollisimman yksinkertaiseksi, jonka takia tietokoneversion jakamisessa päädyttiin valitsemaan Dropbox. Valinta oli hyvin yksinkertainen mutta toimiva.

Jatkokehityksenä sovellukseen olisi mahdollista kartoittaa ominaisuuksia, joiden avulla sen käytettävyyttä voitaisiin parantaa. Esimerkiksi useiden tuotteiden lisääminen ja muokkaaminen tietokantaan täytyy nykyisessä toteutuksessa tehdä yksi kerrallaan, joten mahdollisuus lisätä ja muokata useita tuotteita kerralla säästäisi aikaa.

Jos harkittaisiin, voisiko sovelluksia pienin muutoksin soveltaa joihinkin muihin käyttötarkoituksiin, niin vastaus olisi, että ei. Sovellukset on kehitetty hyvin tarkoin määritelmin, jos sovelluksien toiminallisuutta muutettaisiin tai Firestore-tietokantaan tallennettavien tietoja muutettaisiin, niin se vaatisi sovelluksessa suuria muutoksia. Sovellusta voitaisiin harkita muille yrityksille vuokrattavien tuotteiden, asiakastietojen tai vuokrauksien tilausten tietojen tallettamiseen pienin muutoksin, jos yrityksellä ei ole omaa järjestelmää tietojen tallettamista varten tietokantaan. Tulisi myös harkita, kuinka paljon yritys käyttäisi sovellusta, jos haluttaisiin käyttää Firestore-tietokantaa sen ilmaisen version rajoituksilla.

LÄHTEET

1. Meet Android Studio. 2021. Android. Saatavissa: https://developer.android.com/studio/intro. Hakupäivä 23.4.2021.

2. XML. 2021. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Xml. Hakupäivä 26.4.2021.

3. Get Started With WPF. 2018. Microsoft. Saatavissa: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019. Hakupäivä 9.5.2021

4. XAML. 2021. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Extensible_Application_Markup_Language. Hakupäivä 26.4.2021.

5. Cloud Firestore. 2021. Firebase. Saatavissa: https://firebase.google.com/docs/firestore. Hakupäivä 23.4.2021

6. Cloud Firestore Data model. 2021. Firebase. Saatavissa: https://firebase.google.com/docs/firestore/data-model. Hakupäivä 21.5.2021

7. Pricing plans. 2021. Firebase. Saatavissa: https://firebase.google.com/pricing . Hakupäivä 6.5.2021.

8. Introducing the new Google Play Console. 2021. Google. Saatavissa: https://support.google.com/googleplay/android-developer/answer/9859062?hl=en&ref_topic=3450769. Hakupäivä 23.4.2021.

9. What is Dropbox?. 2021. Dropbox. Saatavissa: https://www.dropbox.com/features. Hakupäivä 23.5.2021

10. Getting started with authentication. 2021. Google. Saatavissa: https://cloud.google.com/docs/authentication/getting-started. Hakupäivä 30.4.2021

11. Manual testing. 2021. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Manual_testing. Hakupäivä 21.5.2021.

Firestore-tietokanta koostui neljästä kokoelmasta, jotka olivat käyttäjät- (Users), asiakkaat- (Customers), tuotteet- (Products) ja tilaukset-kokoelma (Receipts). Alapuolella on lista sovelluksen toiminnallisuuksista ja sen alapuolella on määrittelyt jokaisen kokoelman dokumenteille 1-4.

Firestore koostuu kokoelmista ja dokumenteista. Kokoelmat sisältävät dokumentteja, joihin tallennettava tieto sijoitetaan dokumenttien kenttiin.

Mobiilisovelluksen päätoiminnallisuudet

Uuden käyttäjätilin luonti
Lisätään Firebase-palveluun uusi käyttäjä sähköpostin ja salasanan avulla. Salasanan pituus minimi 6 merkkiä.
Lisätään myös Firestore-tietokannan Users-kokoelmaan dokumentti, joka sisältää uuden käyttäjän nimen, sähköpostin, roolin, jonka arvo on dummyUser, ja Firebase-käyttäjätilin UID-koodi asetettiin dokumentin nimeksi. Dokumentin tarkat tiedot (1.).
Sisäänkirjautuminen
Sähköpostin ja salasanan avulla autentikoidaan käyttäjä Firebase-palveluun ja noudetaan Firestore-tietokannasta Users-kokoelmasta käyttäjätilin UID-koodin sisältävä dokumentti ja otetaan käyttäjän nimi ja rooli.
Salasanan uusiminen
Lähetetään Firebase-palvelun avulla syötettyyn sähköpostiin viesti, jonka avulla salasana voidaan uusia, jos sähköpostilla on käyttäjätili Firebase-palvelussa.
Asiakkaiden lisääminen, muokkaaminen, poistaminen ja hakeminen.
Lisätään Firestore-tietokantaan Customers-kokoelmaan uusi dokumentti (2.), joka koostuu asiakkaan yhteystiedoista.
Muokataan Customers-kokoelmassa olevan asiakas dokumentin kentissä olevia arvoja.

Poistetaan Customers-kokoelmasta valittu dokumentti.
Noudetaan asiakkaiden dokumentteja vaihtoehdoilla, business-kenttä on true vai false ja lisänä mahdollisuus hakea yrityksen/yksityishenkilön nimen perusteella.
Vuokrattavien tuotteiden lisääminen, muokkaaminen, poistaminen ja hakeminen tietokannasta.
Lisätään Firestore-tietokantaan dokumentti, joka sisältää tuotteen tiedot kuten nimen, painon, hinnan euroina, painon kiloina, tuotteen kokonaismäärän, tuotteen sen hetkisen määrän, kategorian numeron 1-22 ja dokumentin ID on tuotteen koodi VK01001 (3.)
Muokataan Firestore-tietokannassa olevan tuotteen dokumentin kentissä olevia arvoja, mutta ei sallita muokata kategoriaa.
Poistetaan Firestore-tietokannan Products-kokoelmasta valitun tuotteen dokumentti.
Noudetaan Firestore-tietokannasta tuotteita kategorioiden tai tuotteen koodin perusteella.
Uuden tilauksen luonti
Valitaan Firestore-tietokannasta asiakkaan dokumentti, valitaan tietokannasta vuokrattavat tuotteet ja luodaan Receipts-kokoelmaan dokumentti (4.), joka sisältää valitun asiakkaan yhteystiedot, valittujen tuotteiden painot, hinnat, nimet, koodit, valittujen tuotteiden kokonaismäärät, lainassa olevien tuotteiden määrä, tilauksen luonti päivämäärä, arvioitu asennusaika, osoitetiedot johon tuotteet kuljetetaan ja lisätään myös loput dokumentin kentät (4.).
Vähennetään myös tilauksessa valittujen tuotteiden määrät Products-kokoelman dokumenttien amount-kenttien määristä.
Tilauksen hyväksyminen
Asetetaan valitun tilauksen dokumentin requestPending-kenttään arvoksi false, requestAcceptDate-kentän arvoksi asetetaan sen hetkisestä ajasta unix-aikaleima ja productGiver-kenttään asetetaan sen hetkisen käyttäjän nimi.

Tilauksen peruuttaminen
Jos tilausta ei ole hyväksytty, niin tilauksen voi perua eli tilauksessa varattujen tuotteiden määrät lisätään takaisin Products-kokoelman tuotteiden dokumentteihin ja tilauksen dokumentti poistetaan.
Tuotteiden palauttaminen
Lisätään palautettavan tuotteen dokumentin amount-kentän arvoon +1. Vähennetään tilauksen dokumentin productAmountInCartList-listasta palautetun tuotteen kohdalta yksi, lisätään returnProductsDateList-listaan palautuksen päivämäärä unix-aikaleimana, ja lisätään returnsAcceptNameList-listaan käyttäjän nimi, joka asetti tuotteen palautetuksi.
Tilauksen sulkeminen
Kun tilauksen dokumentin kaikki tuotteet ovat palautettu eli productCurrentlyLoanedList-listan kaikki arvot ovat nolla, niin asetetaan productsReturned-kentän arvoksi true.
Tilauksen tietojen esittely
Näytetään tilauksen dokumentissa olevat tiedot eli asiakkaan yhteystiedot, tilauksen luontipäivämäärä ja tuotteiden tiedot, kuten nimi, hinta, koodi, paino, jos tuote on palautettu, niin palautuksen päivämäärä sekä vastaanottajan nimi.
Jos tilaus on hyväksytty, niin näytetään myös tilauksen hyväksymispäivämäärä
Jos tilaus on suoritettu, niin näytetään hinta-arvio kuluneiden päivien ja lainattujen tuotteiden hintojen perusteella. Lisänä myös viimeisen tuotteen palautus päivämäärä.
Tilaus dokumenttien haku Firestore-tietokannasta useilla eri hakuvaihtoehdoilla
Noudetaan Firestore-tietokannan Receipts-kokoelmasta dokumentteja asiakkaan nimen perusteella, aikavälin perusteella milloin tilauksen dokumentti on luotu, tilauksessa olevan tuotteen koodin perusteella ja sen perusteella onko tilaus hyväksymättä, hyväksytty vai suoritettu.

Alla olevat listat sisältävät Firestore-tietokannan eri kokoelmien dokumenttien kenttien nimet, mitä niihin tallennetaan ja missä muodossa mobiilisovelluksessa.

1. **Käyttäjä-dokumentin tiedot.**

Käyttäjä-dokumenttiin tallennettava tieto	**Tallennettavan tiedon tietotyyppi**	**Kentän nimi Firestore Users- kokoelman dokumentissa**
Käyttäjätilin sähköposti	String	email
Käyttäjän nimi	String	name
Luodun Firebase-käyttäjätilin UID-koodi	String	UID
Käyttäjän rooli	String	clearance

2. Asiakas-dokumentin tiedot.

Asiakas-dokumenttiin tallennettava tieto	**Tallennettavan tiedon tietotyyppi**	**Kentän nimi Firestore Customers- kokoelman dokumentissa**
Asiakkaan nimi	String	name
Yrityksen/yksityishenkilön puhelinnumero	String	phoneNumber
Yrityksen/yksityishenkilön sähköposti	String	email
Asiakkaan alennusprosentti	String	discount
Yrityksen/yksityishenkilön osoite	String	address
Yrityksen/yksityishenkilön postinumero	String	postalcode
Yrityksen/yksityishenkilön postitoimipaikka	String	postalServiceLocation
Onko asiakas yritys Jos asiakas on yritys: true muuten false.	boolean	business
Yrityksen kontaktihenkilön puhelinnumero	String	contactPhoneNumber
Yrityksen kontaktihenkilön nimi	String	businessContact
Yrityksen Y-tunnus	String	businessYCode

3. Tuote-dokumentin tiedot.

Tuote-dokumenttiin tallennettava tieto	**Tallennettavan tiedon tietotyyppi**	**Kentän nimi Firestore Products- kokoelman dokumentissa**
Dokumentin ID on tuotteen koodi		
Tuotteen nimi	String	name
Kategoria tuotteen koodin VK01001 kahden ensimmäisen numeron perusteella 1-22	int	category
Tuotteen hinta per päivä euroina	double	price
Tuotteiden kokonaismäärä	int	totalAmount
Tuotteiden määrä varastossa	int	amount
Paino kiloina	double	weight

4. Tilaus-dokumentin tiedot.

Tilaus-dokumenttiin tallennettava tieto	**Tallennettavan tiedon tietotyyppi**	**Kentän nimi Firestore Receipts- kokoelman dokumentissa**
Asiakkaan sähköposti	String	yritys: businesEmail Yksityishenkilö: email
Yrityksen kontaktihenkilön nimi	String	contactName
Kontaktihenkilön puhelinnumero	String	contactPhone
Asiakaskohtainen alennus	String	customerDiscount
Asiakkaan Firestore-dokumentin ID	String	customerDocID
Yksityishenkilön/yrityksen nimi	String	customerName
Onko asiakas yritys Jos yritys: true muuten false	boolean	isBusiness
Kuljetuksen osoite	String	deliveryAddress
Postitoimipaikka	String	postLocation
Postinumero	String	postNumber
Arvioitu tuotteiden asennusaika	String	installTime

Valittujen tuotteiden kokonaismäärät listassa	List (String)	productAmountInCartList
Lainassa olevien tuotteiden määrä	List (String)	productCurrentlyLoanedList
Tuotteiden nimilista	List (String)	productNameList
Tuotteiden koodilista	List (String)	productIDList
Lainattujen tuotteiden hintalista	List (String)	productPriceList
Tuotteiden painolista	List (String)	productWeightList
Tilauksen hyväksyjän nimi. Käyttäjätilin nimi.	String	productGiver
Tuotteiden palautus päivämäärä lista Unix-aikaleima	List (Int)	returnProductsDateList
Tuotteiden palautuksien vastaanottajan nimilista sisältää oman indeksin jokaiselle tuotteelle	List (String)	returnsAcceptNameList
Tilauksen luonnin päivämäärä Unix-aikaleima	int	requestDate
Tilauksen hyväksymisen päivämäärä Unix-aikaleima	int	requestAcceptDate
Viimeisen tuotteen palautuspäivämäärä Unix-aikaleima	int	finalProductReturnDate
Odottaako tilaus hyväksyntää	boolean	requestPending

Jos tilaus hyväksytty: true muuten false		
Onko tuotteet palautettu Jos kaikki tilauksen tuotteet ovat palautettu: true muuten false	boolean	productsReturned