Tampereen ammattikorkeakoulu

Flutter-ohjelman toteuttaminen Google Play -palveluun

Samuli Riihikoski

OPINNÄYTETYÖ
Kesäkuu 2023

Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

RIIHIKOSKI, SAMULI:
Flutter-ohjelman toteuttaminen Google Play -palveluun

Opinnäytetyö 27 sivua, joista litteitä 1 sivu
Kesäkuu 2023

Opinnäytetyön tavoitteena oli rakentaa sovellus Android-mobiililaitteille. Sovellus toteutettiin Flutterin avulla, joka on Googlen julkaisema avoimeen lähdekoodiin perustuva kehitysympäristö. Sovellukseen lisättiin NoSQL-tietokanta Firebase-palveluun. Lopuksi ohjelma julkaistiin Google Play -palvelussa ja lähdekoodit jaettiin julkisesti.

Toteutetun sovelluksen nimi on Kutsu ja sen avulla käyttäjä voi löytää uusia ystäviä luomalla kutsuja, jotka ovat toisten käyttäjien nähtävissä. Kutsuista voi tykätä, ja lopuksi käyttäjä valitsee henkilön, jonka kanssa voi jatkaa tutustumista keskusteluhuoneessa.

Opinnäytetyössä tarkasteltiin, kuinka sovelluksen rakentaminen kannattaa tehdä niin, että se olisi helposti päivitettävissä. Työssä käsiteltiin Flutter-komponenttaja, joiden avulla koodi saadaan pidettyä siistinä. Lisäksi käytiin läpi seuraavia ohjelman ominaisuuksia: käyttäjän rekisteröityminen vahvalla tunnisteella, NoSQL-tietokannan luomisen Firebase-pilvipalveluun ja käyttäjän sijainnin hakeminen.

Työn tavoitteet saavutettiin, ja ohjelma saatiin julkaistua ajallaan. Flutterista on kehittynyt lyhyessä ajassa luotettava kehitysympäristö, joka sopii erityisesti mobiililaitteille toteutettaviin sovelluksiin. Flutteria käytettäessä on tärkeää hyödyntää sen omia kirjastoja ja komponentteja. Flutterilla kirjoitetut koodit on hyvä pitää mahdollisimman pieninä, koska liian suureksi paisuneet tiedostot tekevät koodista vaikeasti luettavan.

Asiasanat: flutter, firebase, google play

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Embedded Systems and Electronic

RIIHIKOSKI, SAMULI:
Creating Flutter Application to a Google Play Service

Bachelor's thesis 25 pages, appendices 1 page
June 2023

The purpose of this thesis was to create an application for Android devices. The application was created by using a Flutter development tool. Flutter is made by Google and it's based on an open-source environment. The application is using the NoSQL database from the Firebase service. Finally the application was published on the Google Play service and the code was shared with everyone.

The application which was created is called Kutsu. With the Kutsu application the user is able to send invitations to people located near to them. Others are able to like these invitations. The application includes a chatroom where the users can continue to get to know each other better.

The thesis focused on how to build the application would be easy to maintain and update. The thesis investigated different Flutter components that were best used to build an effective application. The thesis investigated deeper these features: the user authentication with a strong identifier, creating a NoSQL database with the Firebase service and fetching the user location.

The goals of this thesis were achieved and the application was published at the Google Play service. Flutter has grown a very powerful development environment and it suits very well developing applications for mobile phones. It's important to use Flutter's own libraries and components to get the best coding results. The code made with Flutter should be kept as small as possible because too long a code makes it harder to read.

Key words: flutter, firebase, google play

SISÄLLYS

LYHENTEET JA TERMIT

Android	Googlen kehittämä käyttöjärjestelmä mobiililaitteille.
API	Application Programming Interface. Ohjelmointirajapinta.
Dart	Googlen kehittämä ohjelmointikieli.
Flutter	Googlen kehittämä avoimeen lähdekoodiin pohjautuva kehitysympäristö.
Firebase	Googlen ylläpitämä tietokantapalvelu.
Github	Verkkosivusto, joka tarjoaa git-versionhallintaa ohjelmistoprojekteille.
Google Play	Googlen omistama digitaalinen sisältöpalvelu.
iOS	iPhone OS, Applen kehittämä käyttöjärjestelmä.
JavaScript	Ohjelmointikieli, jota käytetään nettisivujen luomiseen.
JSON	Javascript Object Notation. Avoimeen standardiin pohjautuva tiedostomuoto tiedonvälitykseen.
Linux	Alunperin Linux Torvaldsin kehittämä käyttöjärjestelmä. Ensimmäinen versio julkaistiin vuonna 1991.
NoSQL	Not only SQL. Tietokanta, joka poikkeaa normaalista relaatiotietokannasta.
URL	Uniform Resource Locator. Yksilöllinen verkkotunniste.

Web	Maailmanlaajuinen verkko, jonka avulla voi hakea sivustoja palvelimelta.
Widget	Flutter ympäristössä oleva käyttöliittymä elementti, jolla on ennalta määrätty toiminnallisuus.

1 JOHDANTO

Yhä useammalla ihmisellä on kännykkä. Tämä luo markkinat, missä kehittäjät voivat luoda sisältöä näille laitteille ja jakaa omia pelejä, ohjelmia tai muuta digitaallista materiaalia maailmanlaajuisesti. Kehittäjillä on mahdollista valita työkalut ja kehitysympäristöt, millä sovelluksen toteuttaa. Onkin tärkeää, että kehittäjät osaavat hahmottaa projektiin liittyvät kokonaisuudet ja valita siihen parhaiten sopivat työkalut.

Flutter on noussut yhdeksi suosituimmaksi työkaluksi erityisesti projekteissa, jotka suunnitellaan mobiilialustoille. Se on Googlen ylläpitämä ja perustuu avoimeen lähdekoodin. Se julkaistiin ensimmäistä kertaa 2018 (Flutter 2023a) ja on hyvin lyhyessä ajassa saavuttanut suosiota kehittäjien keskuudessa. Sille on julkaistu kattava määrä apukirjastoja, jotka nopeuttavat entisestään kehitystyötä.

Aihe tekemääni opinnäytetyöhöni syntyi koulun kurssilta, jossa tehtiin projekti yritykselle. Projektiin tehty käyttöliittymä toteutettiin Flutterin avulla. Yllätyin, kuinka helposti lähestyttävä kyseinen ympäristö on kehittäjän näkökulmasta ja halusin tutustua aiheeseen lisää. Työn tarkoituksena on rakentaa Kutsu-ohjelma Flutter-kehitysalustalla ja julkaista se Google Play -kaupassa. Ohjelman voi ladata ilmaiseksi Android-käyttöjärjestelmillä toimiville mobiililaitteille ja lähdekoodit julkaistaan kaikkien saataville Github-palveluun (Liite 1).

Kutsu-ohjelmalla käyttäjä voi lähettää kutsuja lähellä oleville ihmisille. Jos henkilö haluaa löytää kaverin, esimerkiksi sulkapallon pelaamiseen tai kesäfestareille, voi hän luoda avoimen kutsun, josta toiset käyttäjät voivat tykätä. Markkinoilla on paljon samantyyppisiä sovelluksia, mutta ne perustuvat pääosin ihmisten kuviin ja niistä tykkäämiseen. Tässä työssä toteutettavassa ohjelmassa asiaa lähestytään hieman eri näkökulmasta. Henkilön kuva on edelleen esillä, mutta hakukriteeri tehdään yhteisen aktiviteetin kautta. Ohjelman sisälle rakennetaan keskusteluhuone, missä käyttäjät voivat tutustua paremmin toisiinsa.

Työssä tutkitaan erityisesti ohjelman arkkitehtuuria ja sitä, kuinka ohjelma kannattaa rakentaa, siten että nopea päivitettävyys on mahdollista. Tarkemmin käsitellään myös: käyttäjän rekisteröityminen vahvalla tunnisteella, NoSQL (Not only SQL) -tietokannan toteuttaminen Firebase-pilvipalveluun ja käyttäjän sijainnin hakeminen. Koska testaus on tärkeä osa ohjelmistokehitysprosessia, tarkastellaan työssä myös Flutterin testimahdollisuuksia.

2 TYÖKALUT JA YMPÄRISTÖT

2.1 Flutter

Flutter on Googlen rakentama kehitysympäristö modernien käyttöliittymien rakentamiseen. Flutter-sovellukset koodataan Dart-kielellä. Se on olio-ohjelmointikieli ja se käännetään, joko konekielelle tai JavaScript-koodiksi. (Napoli 2019)

Yksi Flutterin vahvuuksista on sen alustariippumaton kehitysympäristö. Kehittäjä voi rakentaa yhdellä koodilla sekä Android, Linux, Web ja iOS sovelluksen. Viime aikoina Flutteriin on lisätty päivityksiä, mitkä auttavat sovellusten kehittämiseen myös sulautettuihin järjestelmiin. Ominaisuus, että voidaan käyttää samaa koodia useammissa järjestelmissä, nopeuttaa ohjelman kehitystyötä. Toinen hyvä puoli on se, että Flutter on täysin ilmainen. Flutteriin on myös vuosien varrella rakennettu kirjastoja, joiden avulla sovelluksen kehittäminen helpottuu entisestään.

Flutterin haittapuolena voi pitää sitä, että se on täysin Googlen ylläpitämä. Jos Google yllättäen päättää lopettaa Flutterin kehittämisen, niin se voi vahingoittaa yrityksiä, jotka käyttävät kyseistä kehitysympäristöä omissa projekteissaan. Tämän hetkisen suosion kasvu ei tosin näytä, että Flutter olisi katoamassa. Isoista nimistä BMW ja Toyota käyttävät Flutteria omissa tuotteissaan. (Flutter 2023b)

2.2 Firebase

Firebase on Googlen pilvipalveluna tarjoama sovellusalusta. Se kattaa laajat työkalut datan tallentamiseen ja analysointiin. Työkaluihin kuuluu mm. käyttäjän profiilin todennus ja tunnistus, sekä erilaiset pilvitallennusmahdollisuudet. Firebase-alustan avulla kehittäjien ei tarvitse noudattaa normaalia front- ja backend mallista sovelluskehitystä, missä luodaan API (Application Programming Interface) rajapinta backend-koodille. Tämän sijaan käyttäjän laite voi suoraan ottaa yhteyttä pilvessä olevaan tietokantaan. (Biswas 2021)

Firebase-alustan hinnoittelu on tehty houkuttelevaksi. Kehittäjä saa ilmaiseksi käyttöönsä Firebase-alustan monipuoliset työkalut ja maksut alkavat kertymään vasta kävijämäärän kasvaessa tarpeeksi isoksi. Tietokannasta saa esimerkiksi tehdä 50 000 hakua päivässä täysin ilmaiseksi (Firebase 2023a). Tallennustilaa saa käyttöön ennen lisämaksua viisi gigatavua. Tämän takia Firebase sopii hyvin kehittäjille, jotka vasta testaavat ideaansa toimivuutta tai aloitteleville yrityksille.

2.3 Google Play

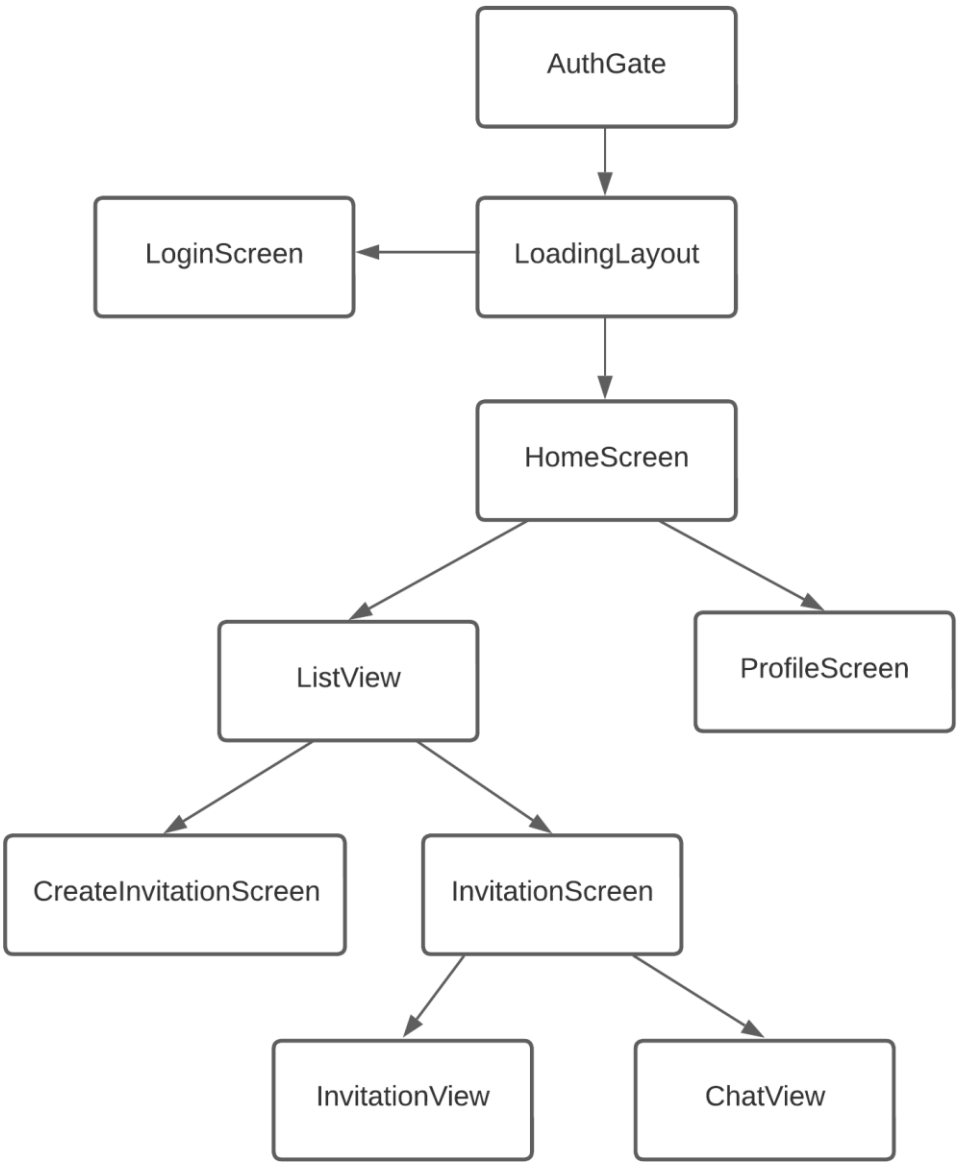
Google Play on Googlen hallinnoima digitaalinen sisältöpalvelu. Palvelussa voi ladata ilmaisia tai maksullisia elokuvia, kirjoja tai musiikkia. Palveluun kuuluu myös Android-käyttöliittymälle tarkoitetut pelit ja sovellukset.

Tämä palvelu tarjoaa kehittäjille väylän jakaa heidän tuotteitaan. Ohjelma tuodaan palveluun Google-konsolisivuston kautta. Sivuston käyttö maksaa 25 euroa, mikä on kertamaksu ja maksun jälkeen kehittäjältä ei peritä muita makuja, vaikka kehittäjä julkaisisi useampia tuotteita tulevaisuudessa. (Play 2023)

3 KUTSU-OHJELMAN TOTEUTUS

3.1 Ohjelman rakenne

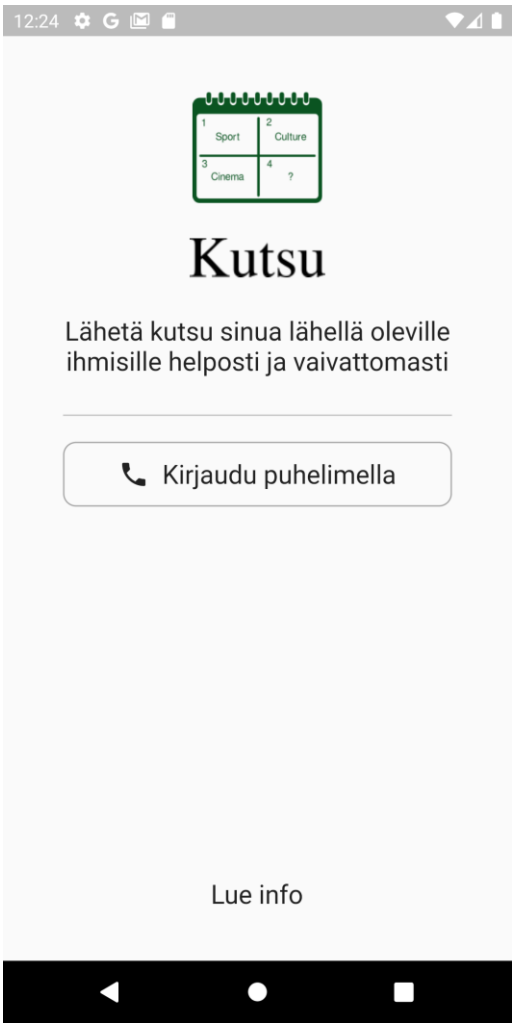
Kuviossa 1 on lohkokaavio Kutsu-ohjelmasta. Kaaviossa olevat elementit ovat ohjelman sivuja lukuun ottamatta AuthGate-komponenttia, joka hoitaa käyttäjän tunnistautimisen ja LoadingLayout-komponentti, jonka avulla hoidetaan pidempi aikaiset lataukset ohjelman sisällä.



Kuvio 1. Kutsu-ohjelman lohkokaavio.

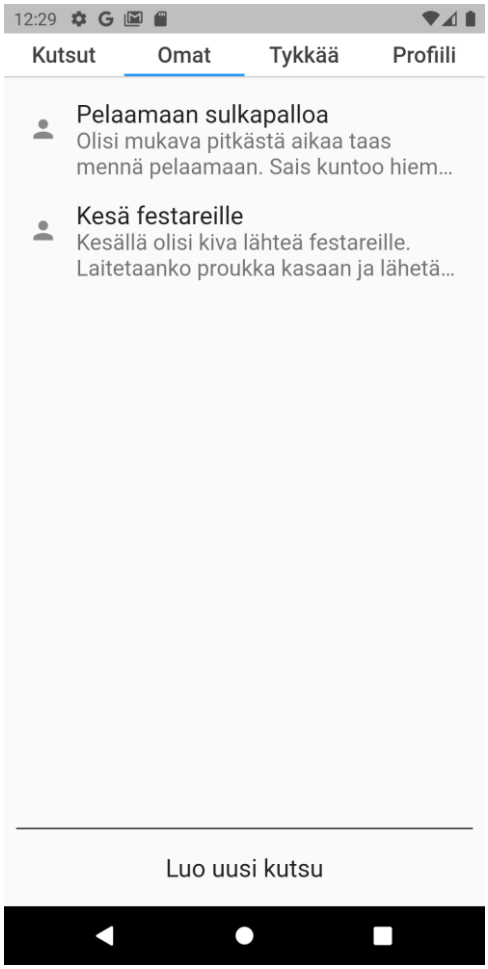
Ohjelma käynnistää ensimmäisenä AuthGate-komponentin. Sen tarkoituksena on kuunnella, onko käyttäjä suorittanut vahvan tunnistautumisen. Jos käyttäjä ei ole tunnistautunut, niin hänet ohjataan tekemään vahva tunnistautuminen puhelinnumeron avulla. Näkymä tästä sivusta on kuvassa 1. Tunnistautumisen jälkeen käyttäjä ohjataan kotisivulle.

On hyvä huomioida, että ennen kotisivua ladataan järjestelmään LoadingLayout-komponentti. Flutterin yksi vahvimpia ominaisuuksia on se, että sen tuottamat ohjelmat koostuvat komponenteista (Widget) ja nämä puolestaan voivat sisältää muita komponentteja (Windmill 2020). Tällä periaatteella voidaan rakentaa hyvin monipuolisia komponentteja ja pitää koodi siistinä ja luettavana.



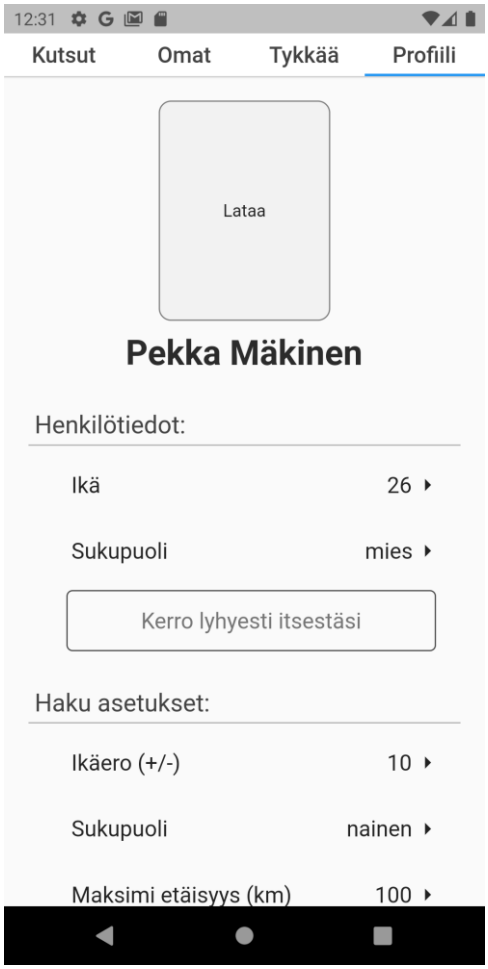
Kuva 1. Ohjelman tunnistautumissivu.

Kotisivu (HomeScreen) koostuu kahdesta pääelementistä. Ensimmäinen elementeistä on kutsulistaukset. Käyttäjä näkee listamuodossa hänen lähellä olevien kutsujen lisäksi hänen luomansa ja tykkäämänsä kutsut. Kaikki kolme eri listaa ovat samanlaisia käyttöliittymä-komponentteja ja vain listojen sisältö vaihtelee. Kuvassa 2 on kuvakaappaus henkilön itsensä luomista kutsuista.



Kuva 2. Listaus käyttäjän omista kutsuista.

Kotisivun toinen elementti on käyttäjän profiilisivu (ProfileScreen). Sen avulla käyttäjä voi ladata profiilikuvan, muuttaa henkilötietojaan ja hakukriteerejä, millä lähellä olevia kutsuja haetaan. Kuvassa 3 on kuvakaappaus kyseiseltä sivulta.



Kuva 3. Käyttäjän profiilin muokkaus sivu.

3.2 Tunnistautuminen

Tunnistautuminen on yksi tärkeimmistä ohjelman ominaisuuksista. Tunnistautumisen toteuttaminen kannattaa tehdä huolellisesti. Sen avulla voidaan varmistaa, että järjestelmään kirjautuvat käyttäjät ovat oikeasti ihmisiä, eikä botteja, jotka yrittäisivät toiminnoillaan häiritä muita käyttäjiä.

3.2.0 Tunnistesivun lisääminen

Kuviossa 1 olevassa lohkokaaviossa AuthGate-komponentti kuuntelee muutoksia sisäänkirjautumisessa. Kuuntelemiseen käytetään StreamBuilder-komponenttia. Sen avulla voidaan tuoda asynkronisesti datapaketteja. Sivu reagoi näihin paketteihin muuttamalla ulkonäköään. StreamBuilder-komponentti kuuntelee

muutoksia, jotka tapahtuvat käyttäjän sisään- ja uloskirjautumisessa (Flutter 2023c). Koodissa 1 on AuthGate-komponentti kokonaisuudessaan.

```
class AuthGateScreen extends StatelessWidget {
  const AuthGateScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return StreamBuilder(
        stream: FirebaseAuth.instance.userChanges(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return LoadingLayout(
              child: LoginScreen(),
            ); // LoadingLayout
          }
          return LoadingLayout(
            child: HomeLayout(),
          ); // LoadingLayout
        }); // StreamBuilder
  }
}
```

Koodi 1. AuthGate komponentti kuuntelee käyttäjän sisäänkirjautumista.

Kyseinen komponentti varmistaa, onko käyttäjä kirjautunut sisään. Jos käyttäjä on kirjautunut sisään, ohjataan hänet kotisivulle, muussa tapauksessa käyttäjä kehotetaan suorittamaan tunnistautuminen uudelleen.

3.2.1 Tunnisteen valitseminen Firebase-palvelusta

Firebasissa on erilaisia tapoja käyttäjän suorittaa tunnistautuminen. Niiden lisääminen ohjelmaan käy kätevästi niille tarkoitettujen apukirjastojen avulla. Ensimmäiseen versioon tunnistautuminen rakennettiin Google- sähköpostin avulla. Gmail-sähköpostin luominen on helppo tehdä ja näin ollen huijareidenkin on helppo alkaa pommittaa palvelua tekaistuilla sähköposteilla. Tietokantaan alkoikin ilmestyä aivan outoja sähköposteja, vaikka ohjelma oli sisäisessä testauksessa.

Tavoitteena on luoda ohjelma, johon käyttäjät voivat luotettavasti kirjautu sisään, koska sähköpostitunnisteesta luovuttiin ja tilalle lisättiin puhelintunnistautuminen. Tunnistus toimii siten, että käyttäjä syöttää puhelinnumeronsa ohjelmaan ja sen jälkeen käyttäjälle lähetetään 6-numeroinen koodi tekstiviestillä.

3.3 Paikannus

Paikannus on yksi ominaisuus kännyköissä, jota hyvin useasti käytetään hyödyksi sovelluksissa. Kutsu-ohjelma käyttää hyödyksi mobiililaitteen paikannuksen mittausta. Flutter tarjoaa tähän tarkoitukseen valmiin paketin nimeltä Location (Flutter 2023d). Kyseinen kirjasto hoitaa automaattisesti luvan kysymisen käyttäjältä sijaintitiedon käyttämiseen ohjelman sisällä. Koodissa 2 nähdään, kuinka helppoa Location-kirjaston käyttäminen on käytännössä.

```
Future<LocationData?> getLocation() async {
  Location location = Location();
  bool _serviceEnabled;
  PermissionStatus _permissionGranted;
  LocationData _locationData;

  _serviceEnabled = await location.serviceEnabled();
  if (!_serviceEnabled) {
    _serviceEnabled = await location.requestService();
    if (!_serviceEnabled) {
      return null;
    }
  }

  _permissionGranted = await location.hasPermission();
  if (_permissionGranted == PermissionStatus.denied) {
    _permissionGranted = await location.requestPermission();
    if (_permissionGranted != PermissionStatus.granted) {
      return null;
    }
  }

  _locationData = await location.getLocation();
  return _locationData;
}
```

Koodi 2. Sijaintitiedon hakeminen Android-kännykällä.

Kutsu-ohjelma vertaa käyttäjien sijaintitietoja toisiinsa ja kerää kutsut vain käyttäjän läheltä olevilta ihmisiltä. Kahden pisteen etäisyyden laskentaan on laskukaavat, mutta tässä ohjelmassa käytetään karkeampaa laskentatapaa, koska ohjelma sallii pienen heiton etäisyyden laskennassa.

Koodissa 3 on funktio ohjelmasta, jota käytetään kahden käyttäjän etäisyyden laskemisessa. Siinä käytetään hyväksi tietoa, että yksi asteen kymmenys on suurin piirtein 111 kilometriä kartalla.

```
bool userInDistance(double latitude, double longitude) {

  double distance = user!.filters.distance * 0.00111;

  if (user!.location.latitude > (latitude + distance)) return false;
  if (user!.location.latitude < (latitude - distance)) return false;
  if (user!.location.longitude < (longitude - distance)) return false;
  if (user!.location.longitude > (longitude + distance)) return false;

  return true;
}
```

Koodi 3. Sijainnin laskemiseen käytetty funktio.

3.4 Tietokannan rakennus

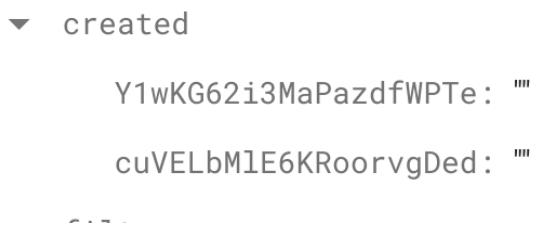
Mobiililaitteille tyypillistä on, että datapaketteja haetaan usein ja pienissä paketeissa. Tietokanta kannattaa suunnitella tätä periaatetta tukien. Kuten aiemmin mainittiin, niin Firebase sisältää NoSQL-tietokannan. Tietokannan dokumentit ovat JSON (JavaScript Object Notation) -tyyppinen ja ne tukevat 34 syvyyteen meneviä tasoja.

Tietokantaa rakentaessa kannattaa JSON-tiedostojen tasojen määrä pitää mahdollisimman pienenä. Mitä enemmän on tasoja, sitä monimutkaisimmiksi dokumentit muuttuvat ja niiden lataaminen ja läpikäyminen vie pidemmän ajan (Firebase 2023b). Kutsu-ohjelmaa varten luomme tietokannan, mikä koostuu kolmesta eri kokoelmasta: käyttäjistä, kutsuista ja keskusteluista. Kuvassa 4 on käyttäjä-dokumentin rakenne.

Kuva 4. Käyttäjä-dokumentin rakenne.

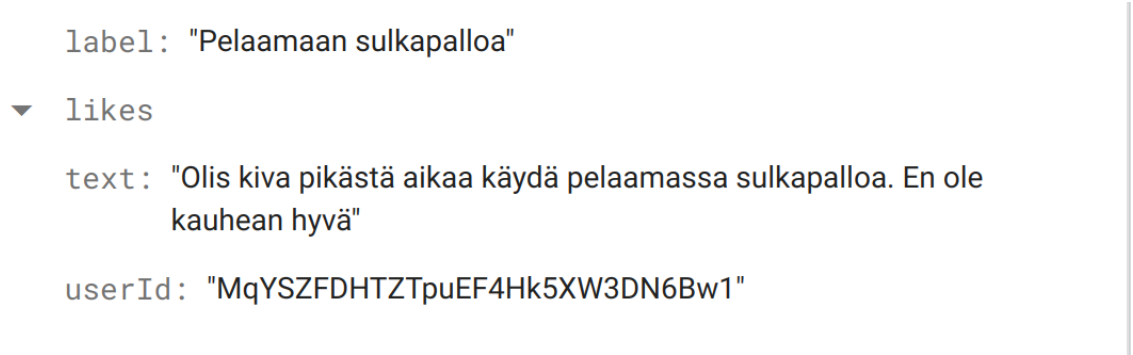
Käyttäjä-dokumentti sisältää henkilötietojen lisäksi tiedon paikkasijainnista, luoduista kutsuista ja tykätyistä kutsuista. Käyttäjä-dokumentti luodaan, kun henkilö kirjautuu ensimmäistä kertaa sovellukseen ja tuhotaan, kun käyttäjä valitsee *”poista profiili”* -vaihtoehdon.

Kuvassa 5 on käyttäjän luomat kutsut tietokantamuodossa. Siinä on käytössä hajautustaulu, missä avaimena on kutsun id ja mahdollisena arvona on keskusteluhuoneen id. Jos arvona on tyhjä, tarkoittaa se, että kutsu on vielä avoin kaikille ja keskusteluhuonetta ei ole luotu. Kun tietokanta muodostetaan tähän tapaan, saadaan kutsut ja keskusteluhuoneet yhdistettyä luonnollisella tavalla.



Kuva 5. Luotujen kutsujen hajautustaulu tietokannassa.

Kuvassa 6 on Kutsu-dokumentin kuvaus tietokannassa. Se sisältää otsikon ja tekstikentän lisäksi, käyttäjä id:n, joka loi kutsun ja listauksen käyttäjistä, jotka ovat tykänneet kutsusta.



Kuva 6. Kutsu-tietokanta dokumentti tietokannassa.

Kutsut ovat haetuin dokumentti tietokannasta, joten on hyvä kiinnittää huomiota, kuinka voisimme pienentää tietokantahakuja tältä osin. Esimerkiksi, jos 100 käyttäjää löytää 100 kutsua lähialueeltaan, on tietokannasta tieto haettava 10 000 kertaa. Näin iso hakumäärä kuormittaa tietokantaa aivan liikaa.

Kun käyttäjä luo kutsun, sitä ei muuteta sen elinkaaren aikana. Kutsun tiedot pysyvät samana sen luomisesta aina sen tuhoamiseen asti. Näin ollen voimme väliaikaisesti varastoida kutsut käyttäjän mobiililaitteelle.

```
Future<app.Date?> getDate(String refId) async {
  if (_cacheIndex >= 1000) _cacheIndex = 0;

  app.Date? date = _cache.firstWhereOrNull(
    (e  Date? date  lement.refId == refId,
  );
      Type: Date?
  if (date == null) {
    final userRef =
        await FirebaseFirestore.instance.collection('dates').doc(refId).get();

    final data = userRef.data() as Map<String, dynamic>;
    date = app.Date(
        refId: refId,
        userId: data['userId'],
        label: data['label'],
        text: data['text']);
    storeDate(_cacheIndex, date);

    _cache[_cacheIndex] = date;
    _cacheIndex++;
  }

  return date;
}
```

Koodi 4. Funktio, millä haetaan tieto laitteelta tai tietokannasta.

Koodista 4 nähdään, kuinka kutsujen väliaikainen tallennus laitteelle on toteutettu. Yllä olevan koodin lisäyksen jälkeen, kutsu pyritään hakemaan ensijaisesti laitteen muistista. Jos kyseistä kutsua ei löydetä laitteelta, niin tämän jälkeen se haetaan tietokannasta. Tämä vähentää merkittävästi hakujen määrää ja nopeuttaa ohjelman käyttöä.

4 TESTAUS

4.1 Testaus Flutter-ympäristössä

Testaus on osa ohjelmistokehitystä ja Flutter sisältää kattavat työkalut sen tekemiseen. Flutterilla on mahdollista tehdä: yksikkötestit, komponenttitestit ja integraatiotestit (Zaccagnino 2020). Tässä työssä käsitellään vain yksikkö- ja komponenttitestejä.

4.2 Yksikkötestiesimerkki

Kaikista testeistä eniten kehittäjien pitäisi luoda juuri yksikkötestejä. Niiden avulla saadaan nopeasti testattua funktion tai luokkien toiminnallisuus. Testaamme Kutsu-ohjelmassa käyttäjän syöttämän puhelinnumeron oikeellisuutta. Testi on näkyvissä koodissa 5.

```
Run | Debug
void main() {
  Run | Debug
  test('Testing valid phonenumber', () {
    final loginScreen = LoginScreen();
    bool isValid = loginScreen.isValidNumber("044.1234567");
    expect(isValid, false);
    isValid = loginScreen.isValidNumber("044");
    expect(isValid, false);
    isValid = loginScreen.isValidNumber("0441234567");
    expect(isValid, true);
  });
}
```

Koodi 5. Yksikkötesti.

Testin luominen on yksinkertainen toimenpide. Ne rakennetaan test()-funktion sisälle. Expect()-funktion avulla luodaan ehdot, milloin testi onnistuu tai epäonnistuu. Puhelinnumeroa syöttäessä käyttäjällä on mahdollisuus vahingossa lisätä esimerkiksi piste numeroiden joukkoon, joten tällaiset tapaukset on hyvä lisätä testien joukkoon.

4.3 Komponenttitestiesimerkki

Komponenttitestien avulla saadaan testattua jotain sellaista, mikä tarvitsee graafisen käyttöliittymän testin ajaksi.

```
Widget createScreen = MaterialApp(
  home: CreateDateScreen(),
); // MaterialApp

Run | Debug | Profile
void main() {
  testWidgets("Test buttons for creating new invitation", (tester) async {
    await tester.pumpWidget(createScreen);
    expect(find.text('Luo uusi kutsu'), findsOneWidget);
  });
}
```

Koodi 6. Komponenttitesti.

Koodissa 6 olevalla komponenttitestissä lataamme ensin sivun esiin, missä luodaan uusi kutsu. Sen jälkeen find.text()-funktion avulla, etsitään käyttöliitymäkomponenttia, millä luodaan uusi kutsu. Testi onnistuu, jos kyseinen elementti löytyy.

5 OHJELMAN JULKAISU

Sisäisen testin jälkeen Kutsu-ohjelma oli valmis julkaistavaksi Google Play -palveluun. Palvelu tarjoaa kattavat ominaisuudet, kuinka laajaan levitykseen ohjelma halutaan rajata. Alkuperäinen suunnitelma oli julkaista ohjelma vain kotimaassa, joten julkaisumaaksi asetetaan Suomi. Googlella kestää muutama päivä tarkastaa, että ohjelma täyttää sen asettamat julkaisukriteerit.

Julkaisun yhteydessä sovellukselle rakennetaan sivu, mikä näkyy Google Play-palvelussa. Sovelluksesta pitää näkyä muutama kuvakaappaus kyseisellä sivulla. Google haluaa myös, että annat URL (Uniform Resource Locator) -osoitteen, missä kerrotaan tietoturvapolitiikasta. Mitä tietoa keräät käyttäjästä ja kuinka tätä tietoa käytetään sovelluksen sisällä. URL ei tarvitse olla kotisivu, vaan se voi olla esimerkiksi linkki Google documents -tiedostoon.

Kun tarvittavat tiedot on annettu, ohjelma on valmis yleiseen jakeluun. Ohjelman julkaisun jälkeen Googlelta kestää muutama päivä ennen, kuin sovellus on kaikkien nähtävillä. Kuvassa 7 on Kutsu-ohjelman kuvake Google Play -palvelussa.

Kuva 7. Julkaistu Kutsu-ohjelman näkymä Google Play -palvelussa.

Julkaisun jälkeen ilmeni pieniä ongelma, jossa latausikkuna jäi pyörimään ruudulle. Ohjelma jouduttiin väliaikaisesti poistamaan Google Play -palvelusta. Ongelma saatiin korjattua ja ohjelma saatiin lisättyä uudestaan palveluun, josta sen voi ladata Android-puhelimille. Ohjelman lähdekoodi on saatavilla Github-palvelussa. (Liite 1)

6 POHDINTA

Tässä työssä pyrittiin antamaan kokonaiskuva, kuinka toteutettaa Flutter-ohjelma ja saada julkaistua se Google Play -kauppaan. Flutterilla sovellusten kehittäminen on tehty helpoksi ja kehitysympäristön tarjoamat kirjastot helpottavat kehittäjien työtä. Valmiiden komponenttien määrä on suuri ja vaarana onkin, että oikean komponentin löytäminen voi olla haastavaa. Onneksi Google on kirjoittanut kattavat Flutter-dokumentaatiot, mikä auttaa oikean komponentin löytämisessä.

Yhteen ongelmaan törmäsin useamman kerran Kutsu-ohjelmaa kirjoittaessa. Flutterissa aaltosulkeiden käyttö on hyvin runsasta. Huomasin useamman kerran laskevani aaltosulkeiden lukumäärää, koska jostain kohtaa koodia sellainen puuttui. Aaltosulkuongelmaa saa pienennettyä pitämällä komponenttien koot pieninä.

Työ oli mielenkiintoinen ja ohjelma saatiin rakennettua ajallaan. Esimerkkitestit, jotka rakennettiin ovat hyvä alku, mutta tulevaisuudessa testien lukumäärää pyritään kasvattamaan suuremmaksi. Ohjelma saatiin julkaistua Google Play -palveluun. Kyseinen palvelu auttaa kehittäjiä testaamaan ohjelmaa sisäisten testien avulla. Tällä saadaan varmistettua, että julkaisun yhteydessä tarjotaan loppukäyttäjälle sovellus, jota on ilo käyttää.

LÄHTEET

Biswas, Nabendu. 2021. Beginning React and Firebase: Create Four Beginner-Friendly Projects React and Firebase. Berkley, CA: Apress L.P

Flutter 2023a. Why Flutter is the most popular cross-platform mobile SDK. Verkkosivu. Viitattu 28.5.2023 https://stackoverflow.blog/2022/02/21/why-flutter-is-the-most-popular-cross-platform-mobile-sdk/

Flutter 2023b. Flutter apps in production. Verkkosivu. Viitattu 3.5.2023 https://flutter.dev/showcase

Flutter 2023c. StreamBuilder widget. Verkkosivu. Viitattu 10.5.2023 https://api.flutter.dev/flutter/widgets/StreamBuilder-class.html

Flutter 2023d. Flutter location plugin. Verkkosivu. Viitattu 2.5.2023 https://pub.dev/packages/location

Firebase 2023a. Pricing plans. Verkkosivu. Viitattu 4.5.2023 https://firebase.google.com/pricing

Firebase 2023b. Structure your database. Verkkosivu. Viitattu 15.5.2023 https://firebase.google.com/docs/database/ios/structure-data

Napoli, Marco L. 2019. Beginning Flutter. Wrox Press

Play 2023. How to use Play Console. Verkkosivu. Viitattu 1.6.2023 https://support.google.com/googleplay/android-developer/answer/6112435?hl=en#zippy=%2Cstep-pay-registration-fee

Zaccagnino, Carmine. 2020. Programming Flutter: native, cross-plaform apps the easy way, Pragmatic Bookshelf

Windmill, Eric, 2020. Flutter in Action, Manning Publications

LIITTEET

Liite 1. Kutsu-ohjelman lähdekoodit.

https://github.com/SamuliRiihikoski/Kutsu