Tampereen ammattikorkeakoulu

Kimppakyyti-mobiilisovelluksen toteuttaminen Flutterilla

Timo Eskola

OPINNÄYTETYÖ
Maaliskuu 2024

Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

ESKOLA, TIMO:
Kimppakyyti-mobiilisovelluksen toteuttaminen Flutterilla

Opinnäytetyö 29 sivua
Maaliskuu 2024

Opinnäytetyössä toteutettiin Android-käyttöjärjestelmälle sovellus kimppakyytien järjestämisen helpottamiseksi alustariippumattomalla Flutter-ohjelmistoteknologialla. Sovelluksessa hyödynnetään Firebasen autentikointi- ja tietokantapalveluita sekä Googlen karttapalveluita.

Kimppakyydeillä torjutaan ilmastonmuutosta ja vähennetään polttoainekustannuksia, ruuhkia ja pysäköintialueiden tarvetta. Sovelluksen tarkoituksena on luoda helpompi tapa järjestää kimppakyytejä nykyisin yleisimmin käytetyille kanaville ja näin lisätä kimppakyytien suosiota ja tehdä kimppakyydeillä matkustamisesta houkuttelevampaa.

Sovellus mahdollistaa kyydin lisäämisen tietokantaan ja kyytien hakemisen tietokannasta. Sovellus laskee reitin käyttäjältä saadun lähtöpaikan, määränpään ja mahdollisten välietappien perusteella. Lähtöpaikka, määränpää ja välietapit voidaan määrittää joko valitsemalla sijainnin kunta tai valitsemalla piste kartalta. Hakutoiminto löytää kaikki tietokantaan lisätyt kyydit, joiden lähtöaika vastaa haussa määritettyä päivämäärää ja reitti kulkee haussa määritettyjen lähtöpaikan ja määränpään kautta.

Opinnäytetyössä ei kehitetty valmista, julkaisukelpoista sovellusta, vaan monia ominaisuuksia jätettiin jatkokehitykseen. Sovellus sisältää kuitenkin oleellisimmat ominaisuudet: autentikoinnin Google-tunnuksilla, kimppakyydin ilmoittamisen ja kimppakyytien hakemisen ja toimii siten hyvänä pohjana valmiin sovelluksen kehittämiseen. Flutter osoittautui hyväksi valinnaksi sovelluksen kehittämiseen, eikä kehityksessä ei tullut vastaan suuria ongelmia vaan kaikki tavoitteenmukaiset ominaisuudet saatiin valmiiksi.

Asiasanat: android, firebase, flutter, kimppakyyti

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

ESKOLA, TIMO:
Creating a Carpool Mobile Application with Flutter

Bachelor's thesis 29 pages
March 2024

Carpooling combats climate change and reduces fuel costs, traffic jams and the need for parking areas. The purpose of this thesis was to create a new and better way to organize carpooling instead of carpooling groups on Facebook, which is currently the most popular way of organizing carpooling Finland.

In the thesis, an Android mobile application was developed using Flutter technology. The maps for the application were provided by Google Maps. The application also utilised Firebase’s authentication and database services.

In the application users can add new rides by entering the departure point and destination, as well as any intermediate stops, or by selecting the respective locations on the map. The application’s search tool finds all rides that pass through the entered departure point and the destination on the desired date.

The application is not yet ready for deployment, but it contains the most important features: authentication and ability to add and search for rides. It is a good basis for further development.

Key words: android, carpooling, firebase, flutter

SISÄLLYS

LYHENTEET JA TERMIT

Android	Mobiililaitteilla käytettävä käyttöjärjestelmä.
API-avain	Avain ohjelmistorajapinnan käyttämiseen.
collection-if	Mahdollistaa Dart-kielessä if-lauseen kirjoittamisen taulukon sisään.
Dart	Googlen kehittämä ohjelmointikieli.
Facebook	Suosittu yhteisöpalvelu.
Flutter	Googlen kehittämä, Dart-kieleen perustuva alusta-riippumaton ohjelmistoteknologia.
Flutterfire CLI	Komentorivityökalu Firebasen käyttämiseen Flutterissa
Firebase	Googlen ohjelmistokehityspaketti mobiili- ja webbikehitykseen.
Firebase Console	Webbiselaimella toimiva konsoli Firebase projekteille
GeoJSON	JSON-tiedosto, joka sisältää paikkatietoja.
Google Cloud Console	Konsoli Googlen ohjelmistorajapintojen hallintaan.
Gradle	Androidissa käytetty käännösautomaation työkalu
HTTP	Tiedonsiirto protokolla.
iOS	Applen kehittämä käyttöjärjestelmä iPhone-laitteille.

JavaScript	Pääasiassa webbiohjelmoinnissa käytettävä kieli.
JSON	Yksinkertainen tiedostomuoto datan välitykseen ja tallennukseen.
Käänteinen geokoodaus	Sijainnin nimeäminen koordinaattien perusteella.
Linux	Enimmäkseen avoimen lähdekoodin käyttöjärjestelmä.
macOS	Käyttöjärjestelmä Macintosh-tietokoneisiin.
Maps SDK For Android	Ohjelmistorajapinta, joka tarjoaa kartat Android sovelluksille.
Material Design	Googlen kehittämä muotokieli.
NoSQL	Perinteisestä relaatiomallista poikkeava tietokanta.
null	Tyhjä arvo
null safety	Estää ohjelmointivirheet, joissa tahattomasti käytetään muuttujaa, jonka arvo on null.
POST	HTTP-protokollan metodi, jossa vastaanottaja käsittelee lähetetyn datan sisällön.
React Native	Metan kehittämä alustariippumaton ohjelmistoteknologia.
Realtime Database	Reaaliaikainen NoSQL Firebase-pilvitietokanta, jossa data on JSON-muodossa.

regex	Säännöllinen lauseke. Määrittää joukon merkkijonoja.
Routes API	Ohjelmistorajapinta, joka tarjoaa reitin annettujen pisteiden välille.
SHA-1	Hajautusalgoritmi
spread-operaattori	Jakaa taulukon elementteihin.
stateful widget	Widget, jolla on oma tila.
stateless widget	Muuttumaton widget.
URL-osoite	Verkkosivuston osoite Internetissä
view	Natiivin Androidin käyttöliittymäkomponentti.
Visual Studio Code	Suosittu ohjelmoijille tarkoitettu tekstieditori.
widget	Flutterissa käyttöliittymän peruskomponentti.
Windows	Microsoftin luoma käyttöjärjestelmä.
XML	Android-sovelluksissa käytetty tiedostoformaatti

1 JOHDANTO

Kimppakyydillä tarkoitetaan kuljettajan lisäksi yhden tai useamman matkustajan yhteistä auton käyttöä. Kimppakyytien hyödyt ovat moninaiset: kuljettaja säästää ainekustannuksissa, kimppakyydit vähentävät ruuhkia ja pysäköintialueiden tarvetta sekä vaikuttavat positiivisesti ilmastonmuutoksen torjuntaan. (Kallio 2013)

Kirjoitushetkellä suosituimpia kimppakyytien järjestämiskanavia ovat Facebookissa toimivat kimppakyyti-ryhmät. Yleisen kimppakyyti ryhmän (Facebook2023a) käyttäjämäärä on yli 60 000. Tämän lisäksi eri reiteille on omia ryhmiä. Esimerkiksi Tampere-Helsinki välin ryhmässä käyttäjiä on yli 10 000 (Facebook 2023b). Ryhmissä kyytien järjestäminen tapahtuu ryhmän seinälle kirjoitetuilla ilmoituksilla.

Työn tarkoituksena on luoda vaihtoehtoinen tapa järjestää kimppakyytejä Facebook-ryhmille. Tarkoitukseen rakennetulla mobiilisovelluksella saadaan kaikki kimppakyydit keskitetysti yhteen paikkaan. Lisäksi voidaan toteuttaa lukuisia käyttäjiä helpottavia ominaisuuksia, tärkeimpänä kyytien hakeminen reitin ja päivämäärän perusteella.

Sovellus toteutetaan alustariippumattomalla teknologialla suurimman mahdollisen käyttäjämäärän saavuttamiseksi. Teknologiaksi valitaan Flutter, jolla sovellukseen saadaan helposti integroitua Googlen karttapalvelut. Sovelluksesta kehitetään Android-versio. iOS-version luominen onnistuu pienillä muutoksilla. Tämä jätetään jatkokehitykseen. Työssä ei laadita julkaisukelpoista versiota sovelluksesta mutta se sisältää tärkeimmät ominaisuudet: autentikoinnin, kyydin lisäämiset tietokantaan ja kyydin haun.

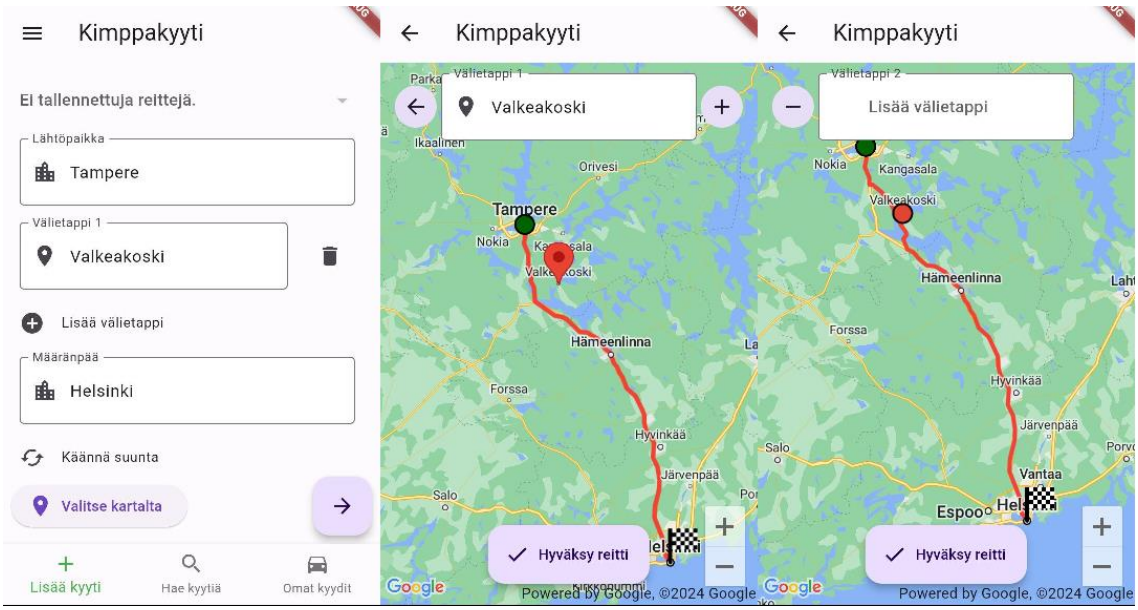
2 SOVELLUKSEN ESITTELY

Käyttäjän kirjauduttua sisään avautuu sovelluksen navigointinäkymä näytön alareunaan (kuva 1), josta käyttäjä voi valita uuden kyydin lisäämisen, kyytien etsimisen tai listauksen omista kyydeistä.

2.1 Kyydin lisääminen

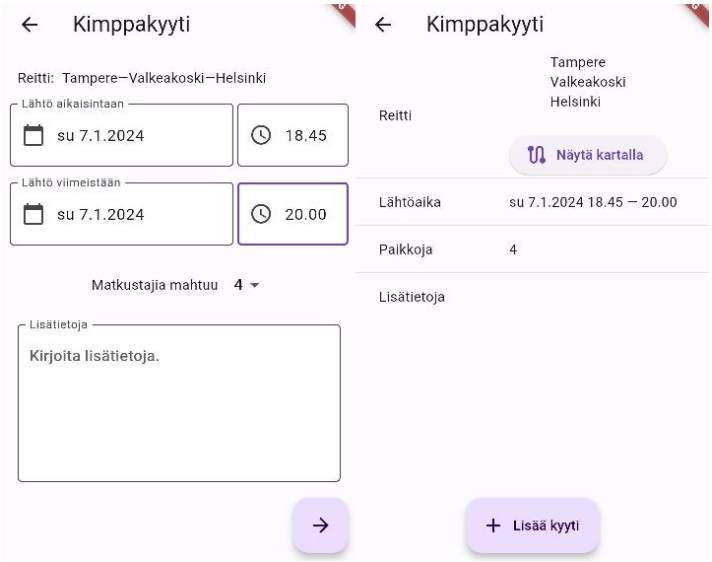
Käyttäjä voi valita reitin valitsemalla lähtöpaikan, määränpään ja mahdolliset välietapit kuvan 1 valikosta tai karttanäkymän avulla. Karttanäkymässä kohteet valitaan kirjoittamalla tekstikenttään tai painamalla karttaa ja painamalla oikean yläkulman lisäyspainiketta.

Valittu piste näkyy punaisella merkitsijällä (kuvassa 1 Valkeakosken kohdalla), lähtöpaikka vihreänä pisteenä, välietappi punaisena pisteenä ja määränpää ruutulippuna. Kuvassa 1 punainen viiva on sovelluksen generoima reitti, joka muodostuu ruudulle, kun määränpää on valittu ja generoidaan uudestaan välietappeja lisätessä tai poistaessa.



KUVA 1. Kyydin lisäys, reitin valinta.

Reitin valittuaan käyttäjä ohjataan seuraavalle sivulle (kuva 2), jossa valitaan lähtöaika, kyytiin mahtuvien matkustajien määrä ja mahdolliset lisätiedot. Nämä täytettyään siirtyy käyttäjä kuvan 2 vahvistus sivulle, josta painamalla ”Lisää kyyti”-painiketta, uusi kyyti tallentuu tietokantaan.

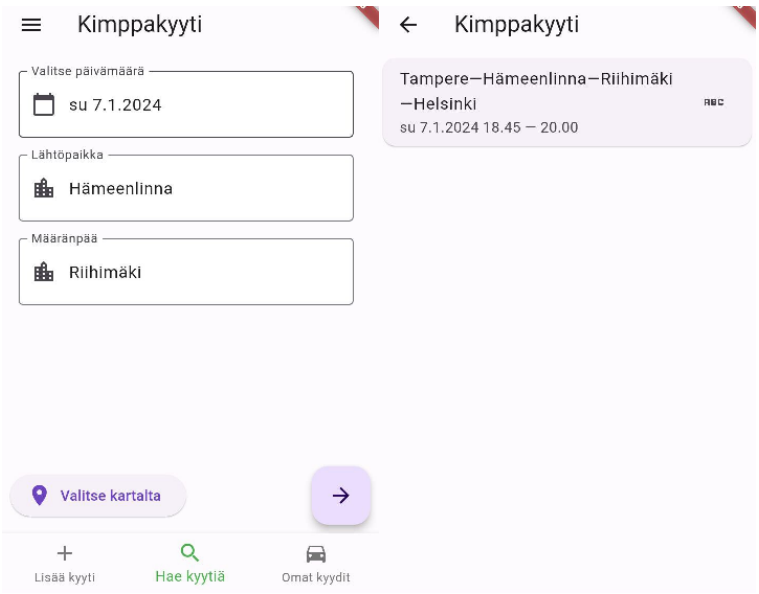


KUVA 2. Kyydin lisäys, muut tiedot.

2.2 Kyydin haku

Kyytejä voidaan hakea antamalla lähtöpaikka, määränpää ja päivämäärä (kuva 3). Samaan tapaan kuin kyydin reitinvalinnassa (kuva 1) voidaan myös lähtöpaikka ja määränpää määrittää karttanäkymästä.

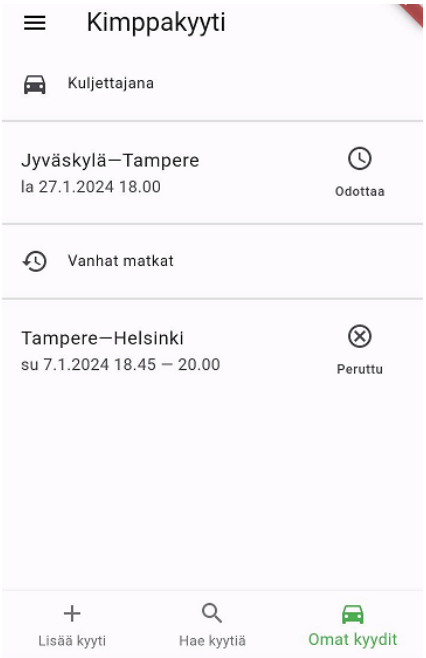
Annettuaan hakuparametrit, käyttäjä siirtyy haun tulossivulle, jossa näytetään kaikki kyydit, joiden reitti kulkee haussa määritetyn lähtöpaikan ja määränpään kautta annettuna päivämääränä. Kuvassa 3 näkyy, että sovellus antaa hakutuloksena kuvassa 1 ja 2 lisätyn kyydin Tampereelta Helsinkiin, kun haetaan kyytiä Hämeenlinnasta Riihimäelle.



KUVA 3. Kyydin haku

2.3 Omat kyydit

Omat kyydit -sivulla näkyy listaus käyttäjän lisäämistä kyydeistä otsikon: kuljettajat alla (kuva 4). Otsikon: menneet kyydit alapuolelta löytyy historia vanhoista kyydeistä.



KUVA 4. Omat kyydit

3 TEKNOLOGIAT JA YMPÄRISTÖ

3.1 Dart

Dart on alun perin JavaScriptille vaihtoehdoksi luotu ohjelmointikieli (Perry 2011). Dart tarjoaa olio-ohjelmoinnista tutut luokat ja monia ohjelmointia helpottavia operaattoreita ja notaatioita, joita esitellään kuvissa 5 ja 6. Kuvassa 5 nähdään, kuinka JavaScriptistä tutun spread-operaattorin avulla saadaan yhdellä koodirivillä lista kunnista, joiden kautta reitti kulkee.

```
List<String> get locations => [
      start.municipality,
      ...waypoints.map((e) => e.municipality),
      destination.municipality
    ];
```

KUVA 5. Esimerkki spread-operaattorin käytöstä

Dartin yksi merkittävistä ominaisuuksista on null safety, jonka avulla estetään yleiset koodausvirheet, joissa käytetään tahattomasti muuttujaa, jonka arvo on null. Kuvassa 6 MyLocations-luokan muuttujat (home, work, custom) voivat saada arvoksi null ja tämä merkitään käyttämällä tyypin perässä kysymysmerkkiä. Kuvassa näkyy myös collection if -ominaisuuden käyttö, eli if-lause on kirjoitettu listan sisään sekä cascade notaatio (..) listan järjestämiseen.

```
class MyLocations {
  Point? home;
  Point? work;
  Map<String, Point>? custom;

  List<Point> get locations {
    return [
      if (home != null) home!,
      if (work != null) work!,
      ...?custom?.entries.map((e) => NamedPoint.fromPoint(e.key, e.value)),
    ]..sort((a, b) => a.id.compareTo(b.id));
  }

  MyLocations({this.home, this.work, this.custom});
```

KUVA 6. Esimerkki MyLocations-luokasta

3.2 Flutter

Flutter on Googlen kehittämä alustariippumaton ohjelmistoteknologia, jonka pohjana on Dart-kieli. Se on viime vuosina noussut React Nativen ohi suosituimmaksi alustariippumattomaksi teknologiaksi mobiilikehitykseen (Statista 2023). Mobiilialustoiden (Android ja iOS) lisäksi Flutter tarjoaa täyden tuen Windows, Linux ja macOS -alustoille (Sneath 2022).

3.2.1 Widget

Widget on Flutter-sovelluksessa käyttöliittymän peruskomponentti. Widgetit ovat itsenäisiä ja voivat sisältää isäntä- ja lapsiwidgettejä muodostaen hierarkkisen puurakenteen. Widgettejä on kahta tyyppiä, stateless (kuva 7) ja stateful widget (kuva 8). Stateless widgetillä ei ole omaa tilaa vaan se on muuttumaton.

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

  @override
  Widget build(BuildContext context) {
    return StreamBuilder(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.active) {
            return snapshot.hasData
                ? const NavigationPage()
                : const LoginPage();
          }
          return const LoadingSpinner();
        }); // StreamBuilder
  }
}
```

KUVA 7. Autentikointitilan kuuntelija

Stateful widgetillä on oma sisäinen tilansa, joka säilötään State-olioon. Tila sisältää muuttujia, jotka voivat vaihtua ajon aikana. Kun State-olion tila muuttuu ja se kutsuu setState-metodia, kyseinen widget piirretään uudelleen. (Add interactivity to your Flutter app n.d.)

```
class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  bool _isLoading = false;

  @override
  Widget build(BuildContext context) {
    final authProvider = context.watch<AuthProvider>();
    return Scaffold(
      body: _isLoading
          ? const LoadingSpinner()
          : Center(
```

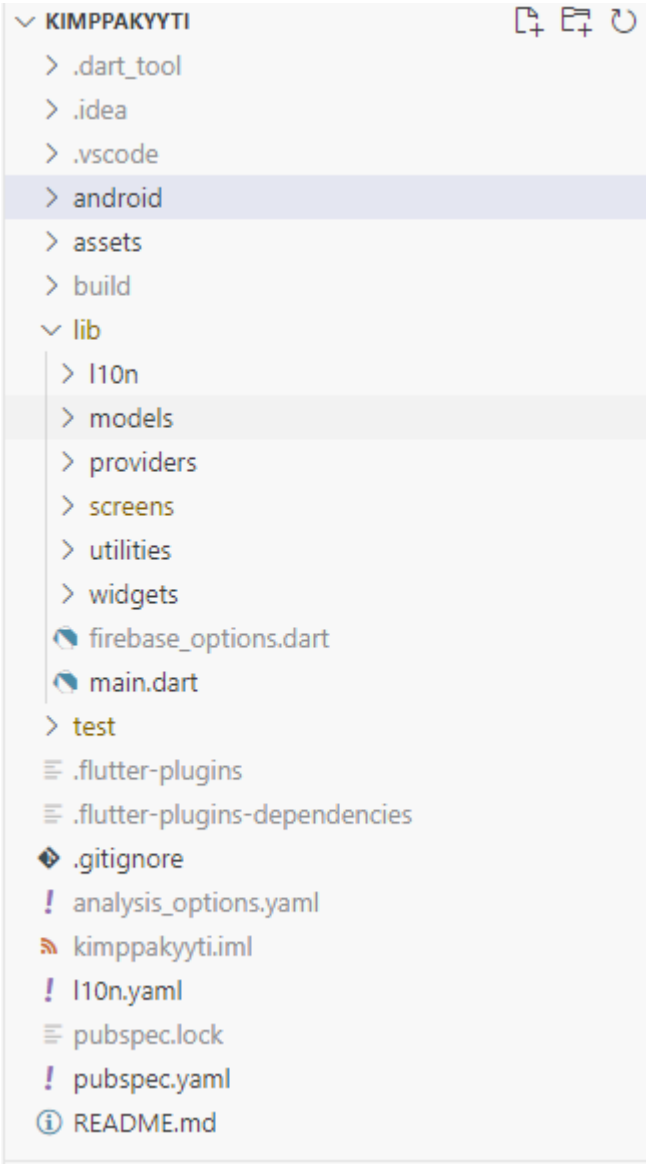
KUVA 8. Stateful widget

3.2.2 Material UI

Flutter käyttää Material design -muotokieltä, jonka avulla saadaan kehitettyä näyttävä käyttöliittymä vaivattomasti (Material n.d.). Toisin kuin React Native, Flutter ei käännä käyttöliittymäkomponentteja Androidin natiiveiksi vieweiksi vaan käyttää omia widgettejään (The Droids on Roids 2023).

3.2.3 Flutter projektin rakenne

Kuvassa 9 on kuvattu projektin rakennetta. Dart-koodi on projektin lib-kansiossa ja sen alikansioissa. Jokaiselle alustalle on oma kansionsa, joka sisältää alusta-spesifiset koodit. Työssä tehdään sovellus vain Androidille, joten kuvassa 9 näkyy android-kansio. Tämä kansio sisältää Android-sovelluksen vaatimat gradle-tiedostot, xml-tiedostot (esim. AndroidManfiest.xml) ja MainActivity.kt tiedoston, joka on Android-sovelluksen ensimmäisen näyttö, joka avautuu käyttäjän käynnistäessä sovelluksen.



KUVA 9. Projektin rakenne

3.3 Firebase

Firebase on Googlen ohjelmistokehityspaketti mobiili- ja webbikehitykseen alustariippumattomille sovelluksille. Firebase tarjoaa monenlaisia työkaluja esim. testaukseen, mainontaan ja kaatumisraportteihin. Tässä projektissa käytetään Firebasen autentikointia ja Realtime Database -tietokantaa.

3.3.1 Autentikointi

Sovelluksen käyttäminen vaatii autentikoinnin. Kirjautumistapana käytetään kirjautumista Google-tilillä. Firebase mahdollistaa myös kirjautumisen mm. Facebook-tilillä ja puhelimella. Näiden toteuttaminen jätetään jatkokehitykseen.

3.3.2 Tietokanta

Firebasen Realtime Database on NoSQL-pilvitietokanta, jossa kaikki data sisältyy yhteen JSON-objektiin. Se ei siis sisällä SQL-tietokannoista tuttuja tauluja. Solmuihin voidaan määrittää luku- ja kirjoitusoikeuksia. Tietokanta sallii vain nopeita operaatioita ja mahdollistaa useiden käyttäjien palvelemisen tinkimättä nopeudesta. Realtime Databasen avulla data saadaan tehokkaasti synkronoitua reaaliajassa käyttäjien laitteiden välillä. (Firebase n.d.)

3.4 Googlen karttapalvelut

Flutteriin on tarjolla liitännäinen, jolla saadaan Googlen karttapalvelut integroitua sovellukseen (Google Maps for Flutter n.d.). Liitännäisen avulla saadaan karttanäkymä, johon voidaan lisätä merkitsijöitä. Merkitsijä voidaan esimerkiksi asettaa käyttäjän painamaan kohtaan kartalla ja rajapinnalta voidaan lukea painetun kohdan koordinaatit.

Sovellus käyttää myös Googlen Routes ohjelmistorajapintaa, joka tarjoaa reitin annettujen pisteiden välille. Rajapinta tarjoaa myös reitin etäisyyden ja arvioidun matka-ajan. Pisteitä voi olla useita eli lähtöpaikan ja määränpään väliin voidaan asettaa välietappeja. Rajapinnan käyttö maksaa viisi Yhdysvaltain dollaria tuhannelta kyselyltä (Google Maps Platform Pricing n.d.).

4 TOTEUTUS

4.1 Firebase

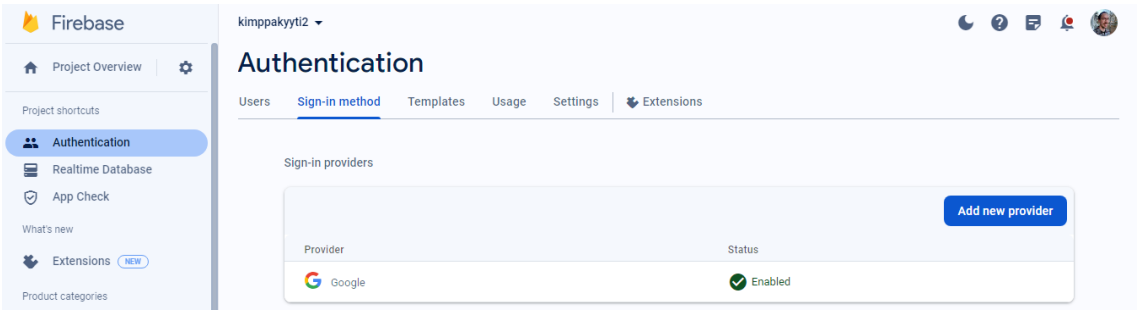
Firebasen asentaminen ja konfigurointi Flutter projektiin onnistuu helpoiten Flutterfire CLI komentorivityökalulla (Add Firebase to your Flutter app). Työkalu generoi projektin lib-kansioon firebase_options.dart tiedoston, johon tallentuu tarvittavat konfiguraatiot autentikoinnin ja tietokannan käyttämiseen. Firebase otetaan käyttöön kuvan 10 koodilla.

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
  runApp(const MyApp());
}
```

KUVA 10. Firebasen alustaminen

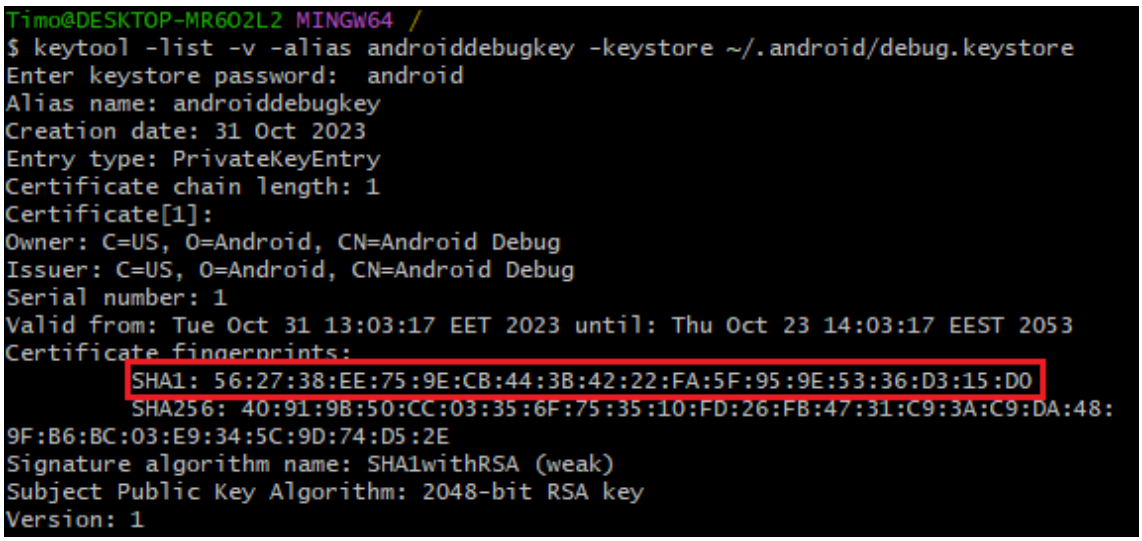
4.2 Autentikointi

Google-tilillä autentikoimista varten täytyy ensin aktivoida Google Firebase consolen kirjautumisvaihtoehdoista (kuva 11).



KUVA 11. Google-tilillä autentikoinnin aktivointi Firebase consolesta.

Tämän lisäksi täytyy Firebase consolessa käydä lisäämässä projektin asetuksista sovelluksen SHA-1 sertifikaatti. Sertifikaatin saa haettua kuvan 12 komennolla.



KUVA 12. SHA-1 sertifikaatti

Kirjautuminen toteutetaan kuva 13 metodilla. Käyttäjän kirjautuessa saadaan käyttäjän uniikki tunniste eli id, käyttäjän nimi ja URL-osoite käyttäjän profiilikuvaan. Jos kirjautunut käyttäjä on uusi käyttäjä, nämä tiedot tallennetaan tietokantaan. Autentikointitilan muutoksia kuunnellaan authStateChanges() metodin avulla (kuva 7). Käyttäjän ollessa kirjautuneena avataan navigointisivu. Jos käyttäjä ei ole kirjautunut, avataan kirjautumissivu.

```
Future<void> signInWithGoogle() async {
  try {
    final gUser = await GoogleSignIn().signIn();
    if (gUser == null) {
      return;
    }
    final gAuth = await gUser.authentication;
    final authCredential = GoogleAuthProvider.credential(
        accessToken: gAuth.accessToken, idToken: gAuth.idToken);
    final credential = await _auth.signInWithCredential(authCredential);
    if (credential.additionalUserInfo!.isNewUser) {
      _addUserToDatabase(credential.user!);
    }
  } on PlatformException catch (error) {
    if (error.code == "network_error") {
      throw SignInException(error: Errors.networkError);
    } else {
      throw SignInException();
    }
  } on FirebaseAuthException catch (error) {
    throw SignInException(message: error.message);
  }
}
```

KUVA 13. signInWithGoogle() metodi

4.3 Tietokanta

4.3.1 Tietokannan rakenne

Kuvassa 14 on kuvattu tietokannan rakennetta. Tietokanta jakautuu kolmeen päähaaraan rides, search ja users. Käyttäjien tiedot tallennetaan users/uid/public polkuun, jossa uid on käyttäjän uniikki tunnus. Tallennettavat tiedot ovat käyttäjän nimi ja URL-osoite käyttäjän profiilikuvaan.

Kyydin tiedot: reitti (Route-luokka luvussa 4.4.3), lähtöaika, kyytiin mahtuvien matkustajien määrä (capacity) ja lisätiedot (info) lisätään polkuun rides/uid/rideKey, jossa uid on kuljettajan tunniste.

KUVA 14. Tietokannan rakenne: haku ja käyttäjät

4.3.2 Validointi, luku- ja kirjoitusoikeudet

Firebase tietokantaan voidaan määrittää sääntöjä eli validoinnin ja luku- ja kirjoitusoikeudet. Validoinnin voi lisätä tietokannan jokaiseen solmuun. Luku- ja kirjoitusoikeudet tietyssä solmussa määräytyy ylimmän haaran oikeuksien mukaan. Kuvassa 15 on esitetty säännöt tietokannan hakuosioon.

Kuvasta nähdään, että lukuoikeudet on myönnetty autentikoiduille käyttäjille search/vuosi/kuukausi/päivä polkuun. Tietokannasta ei saa siis kerralla haettua esim. yhden kuukauden kaikkia hakutuloksia.

Kirjoitusoikeudet on myönnetty polkuun search/vuosi/kuukausi/päivä/rideId, jossa rideId on Firebasen generoima tunniste. Kirjoittaminen onnistuu vain, jos rideId on oikean mittainen, kyseistä rideId:tä ei löydy ennestään tietokannasta samalta päivältä ja tietoparin arvoksi annetaan käyttäjän oma tunniste. Lisäksi validoidaan vuosi, kuukausi ja päivä käyttämällä regex-lausekkeita.

```
"search": {
  "$year": {
    ".validate": "$year.matches(/^[2][0][2-9][0-9]/)",
    "$month": {
      ".validate": "$month.matches(/^[1-9]|1[0-2]/)",
      "$day": {
        ".validate": "$day.matches(/^[1-9]|[12][0-9]|3[01]/)",
        ".read": "auth !== null",
        "$rideId": {
          ".write": "auth !== null",
          ".validate": "!data.exists() && $rideId.matches(/^.{20}$/) && newData.val() === auth.uid"
        }
      }
    }
  }
},
```

KUVA 15. Tietokannan säännöt

4.3.3 Kyydin lisääminen tietokantaan

Kyytiä lisätessä tietokantaan (kuva 16) käytetään Firebasen push() metodia luomaan kyydille uusi 20-merkkinen tunniste (rideKey). Hakutoimintoa varten lisätään polkuun search/vuosi/kuukausi/päivä/ uusi tietopari, jossa avain on rideKey ja arvo kuljettajan uid. Koska lähtöaika on mahdollista olla usealla päivällä (kuva 2), lisätään arvot tarvittaessa myös usealle päivälle.

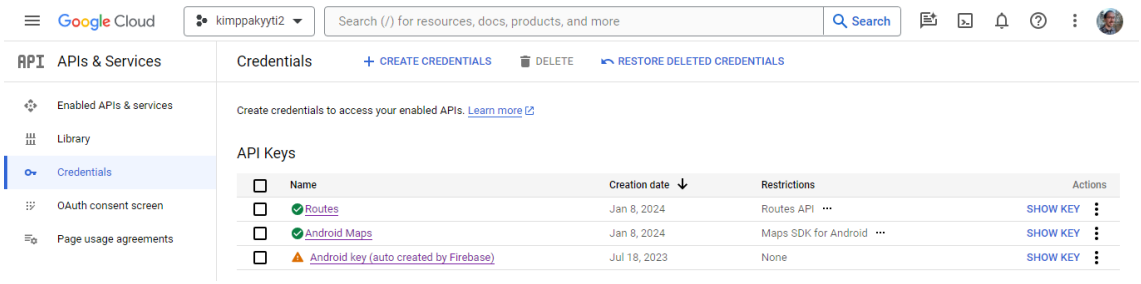
```
Future<String?> addRideToDatabase(Route route, DepartureTime time,
    int capacity, String? info, Function onError) async {
  final uid = FirebaseAuth.instance.currentUser!.uid;
  final Map<String, dynamic> values = {};
  final rideData = {
    'capacity': capacity,
    'info': {'value': info, 'timestamp': ServerValue.timestamp},
    'time': {'data': time.toJson(), 'timestamp': ServerValue.timestamp},
    'route': {'data': route.toJson(), 'timestamp': ServerValue.timestamp}
  };
  final rideKey =
      FirebaseDatabase.instance.ref().child('/rides/$uid/').push().key;
  values['/rides/$uid/$rideKey/'] = rideData;
  for (var day in time.dates!) {
    values['/search/${day.year}/${day.month}/${day.day}/$rideKey/'] = uid;
  }
  values['/users/$uid/private/rides/$uid/$rideKey/'] =
      ServerValue.timestamp;
  String? result;
  await FirebaseDatabase.instance.ref().update(values).then((_) {
    result = rideKey;
  }).catchError((err) {
    onError(err);
    result = null;
  });
  return result;
}
```

KUVA 16. Kyydin lisääminen tietokantaan

4.4 Kartat

4.4.1 Google Maps liitännäinen

Googlen karttapalvelun lisäämiseen sovellukseen tarvitaan API-avain, jonka voi generoida Google Cloud Consolessa (kuva 17). Lisäksi konsolista täytyy lisätä käyttöön Routes API ja Maps SDK For Android ohjelmistorajapinnat. API-avaimia voidaan rajoittaa käytettäväksi esim. vain Android-sovelluksesta ja vain tiettyä ohjelmistorajapintaa.



KUVA 17. API-avaimet Google Cloud Consolessa

Käyttämällä Visual Studio Codea ja Dartin –dart-define ominaisuutta, voidaan API-avaimet piilottaa projektin .vscode/launch.json tiedostoon kuvan 18 tavalla. (Dartcode.org n.d.)

```
"configurations": [
    {
        "name": "Flutter",
        "type": "dart",
        "request": "launch",
        "toolArgs": [
            "--dart-define",
            "API_KEY=AIzaSyCYV_0QlYtT8GzYoCfyF7u1dCfpgJi4rd8",
            "--dart-define",
            "ROUTES=AIzaSyA6uGBTtkqhKljj1I7hXJhIod_ymRtN9zA"
        ]
    }
]
```

KUVA 18. API-avaimet launch.json tiedostossa.

Jotta API-avain saadaan käyttöön Android-sovelluksessa, täytyy muokata build.gradle (kuva 19) tiedostoa (Filho 2021) ja AndroidManifest.xml (kuva 20) tiedostoa. Kartta saadaan näkymään sovelluksessa käyttämällä GoogleMap widgettiä (kuva 21).

```
7   def dartEnvironmentVariables = []
8   if (project.hasProperty('dart-defines')) {
9       dartEnvironmentVariables = project.property('dart-defines')
10          .split(',')
11          .collectEntries { entry ->
12              def pair = new String(entry.decodeBase64(), 'UTF-8').split('=')
13              [(pair.first()): pair.last()]
14          }
15  }
63          manifestPlaceholders += [
64              API_KEY: dartEnvironmentVariables.API_KEY
65          ]
```

KUVA 19. build.gradle

```
<meta-data android:name="com.google.android.geo.API_KEY"
        android:value="${API_KEY}"/>
```

KUVA 20. AndroidManifest.xml

```
GoogleMap(
  minMaxZoomPreference:
      const MinMaxZoomPreference(4.85, double.infinity),
  initialCameraPosition: const CameraPosition(
    target: MapController.centerOfFinland,
    zoom: 4.85,
  ),              // CameraPosition
  cameraTargetBounds: CameraTargetBounds(LatLngBounds(
      southwest: MapController.southwest,
      northeast: MapController.northeast)), // LatLngBounds // CameraT
  mapToolbarEnabled: false,
  rotateGesturesEnabled: false,
  compassEnabled: false,
  myLocationButtonEnabled: false,
  buildingsEnabled: false,
  markers: _markers,
  polylines: _polylines,
  onMapCreated: (mapCon) async {
    await _onMapCreated(mapCon);
```

KUVA 21. GoogleMap widget

4.4.2 Käänteinen geokoodaus

Käyttäjän painaessa kartan pistettä, selvittää sovellus, missä kunnassa kyseinen piste sijaitsee. Kuntarajat ovat polygoneja GeoJSON mudossa (Tebest 2022). Kierroslukumenetelmällä (Sunday 2012) selvitetään, sijaitseeko piste polygonin sisällä. Iteroimalla kuvan 22 funktiota, saadaan selville, missä kunnassa piste sijaitsee.

```
static double _isLeft(LatLng x, LatLng y, LatLng z) {
  return (y.latitude - x.latitude) * (z.longitude - x.longitude) -
      (z.latitude - x.latitude) * (y.longitude - x.longitude);
}

static bool pointInPolygon(LatLng point, List<LatLng> polygon) {
  var wn = 0;
  for (var i = 0; i < polygon.length; i++) {
    var b = polygon[(i + 1) % polygon.length];
    if (polygon[i].longitude <= point.longitude) {
      if (b.longitude > point.longitude &&
          _isLeft(polygon[i], b, point) > 0) {
        wn++;
      }
    } else if (b.longitude <= point.longitude &&
        _isLeft(polygon[i], b, point) < 0) {
      wn--;
    }
  }
  return wn != 0;
}
```

KUVA 22. Kierroslukumenetelmä

4.4.3 Route-luokka

Sovelluksessa reitit talletetaan Route-olioihin (kuva 23). Route-luokka sisältää listan Leg-olioita, joiden lukumäärä on välietappien lukumäärä + 1. Leg-luokan muuttujat ovat etapin lähtöpaikka (start), etapin määränpää (destination), etapin pituus metreissä (distance), etapin arvioitu matka-aika sekunneissa (duration) ja reitti koodattuna merkkijonoksi (polyline).

```
class Route {
  final List<Leg> legs;

  Point get start {
    return legs.first.start;
  }

  Point get destination {
    return legs.last.destination;
  }

  int get distance {
    final values = legs.map((e) => e.distance);
    return values.fold(0, (previousValue, element) => previousValue + element);
  }
class Leg {
  final Point start;
  final Point destination;
  final int distance;
  final int duration;
  final String polyline;

  const Leg(this.start, this.destination, this.distance, this.duration,
      this.polyline);
```

KUVA 23. Route ja Leg luokat

4.4.4 Reitin määrittäminen

Reitti määritetään käyttämällä Routes API rajapintaa. Rajapinta toimii käyttämällä HTTP-protokollan POST metodia (kuva 24). Otsikkoon annetaan API-avain ja maski, johon rajataan, mitä arvoja halutaan serveriltä palautettavan. Kuormaksi annetaan lähtöpaikan, määränpään ja mahdollisten välietappien koordinaatit.

```
class DirectionsProvider {
  static const Map<String, String> headers = {
    "Content-Type": "application/json",
    "X-Goog-Api-Key": String.fromEnvironment('ROUTES'),
    "X-Goog-FieldMask":
        "routes.legs.duration,routes.legs.distanceMeters,routes.legs.polyline.encodedPolyline"
  };
  static Future<Route> fetchRoute({
    required Point start,
    required Point destination,
    required List<Point> waypoints,
  }) async {
    Uri uri = Uri.https("routes.googleapis.com", "directions/v2:computeRoutes");
    try {
      final body = Request(start, destination, intermediates: waypoints).body;
      final response = await http.post(uri, headers: headers, body: body);
```

KUVA 24. Reitin haku Routes API:n avulla

Reitti saadaan rajapinnalta merkkijonoksi koodattuna. Jotta reitti saadaan piirrettyä kartalle, täytyy reitti dekoodata taulukoksi reitin pisteitä (Adedamola 2023). Näiden pisteiden avulla GoogleMap widget saa piirrettyä reitin (kuva 25).

```
Set<Polyline> _routeToPolyline(Route route) {
  final Set<Polyline> result = {};
  for (int i = 0; i < route.legs.length; i++) {
    result.add(Polyline(
        polylineId: PolylineId('route: ${_routes.length} leg: $i'),
        points: MapUtils.decodePolyline(route.legs[i].polyline),
        width: 4,
        color: Colors.red)); // Polyline
  }
  return result;
}
```

KUVA 25. Reittiviivan määrittäminen

5 POHDINTA

Työssä Flutter osoittautui tehokkaaksi ja kehittäjäystävälliseksi teknologiaksi. Dart-kieli on nopea oppia Javasta tutun syntaksin takia ja sen modernit ominaisuudet kuten null safety ja collection if tekevät ohjelmoinnista helppoa. Flutteriin sisäänrakennetun Material UI:n avulla tyylikkään käyttöliittymän suunnittelu ja luominen on vaivatonta. Flutteriin on saatavilla myös kattavat kirjastot erilaisia widgettejä, jotka on helppo lisätä omaan projektiin. Ongelmatilanteissa Flutteriin löytyy verkosta paljon tuoreita ohjeita. Natiiviin Javaan verrattuna Flutterilla saadaan työn kaltainen projekti kehitettyä huomattavasti nopeammin ja samalla koodipohjalla saadaan iOS-versio.

Sovelluksessa on vielä paljon kehitettävää ennen kuin se on valmis julkaistavaksi. Suunnitelmissa on toteuttaa systeemi, jossa kyydin hakija voi lähettää kuljettajalle pyynnön kyytiin pääsystä. Pyyntö tulee kuljettajalle ilmoituksena ja hän voi hyväksyä tai hylätä pyynnön. Lisäksi sovellukseen tarvitaan keskustelu mahdollisuus pyynnön yhteydessä ja kyytiin hyväksyttyjen kesken. Jatkokehitykseen jää myös GPS:n integroiminen sovellukseen. Sovellukseen on tärkeää saada palautejärjestelmä, jossa käyttäjät voivat antaa palautteen kuljettajalle ja matkustajille kyytien turvallisuuden ja luotettavuuden lisäämiseksi. Muita mahdollisia lisäominaisuuksia ovat esimerkiksi polttoainekustannusten laskemien ja maksaminen mobiilimaksusovelluksella kuten MobilePaylla.

Sovellusten julkaiseminen Google Playssa on suhteellisen halpaa ja vaivatonta. Julkaisun jälkeen haasteeksi muodostuu käyttäjäpohjan luominen. Itsestään selvä sovelluksen mainontapaikka on Facebookin kimppakyytiryhmät. Jotta siirtyminen Facebookista sovelluksen käyttäjäksi olisi mahdollisimman mutkaton, tulee sovellukseen lisätä Facebook-tilillä kirjautuminen. Jotta sovellus saavuttaa kaikki mobiililaitteiden käyttäjät, täytyy sovellus julkaista myös Applen App Storessa. Sovelluksesta saadaan iOS-versio pienillä lisäyksillä. Julkaiseminen sen sijaan on Android-versioon verrattuna kallista ja haastavaa.

LÄHTEET

Kallio, S. 2013. Kïmppakyytipalvelut kestävän liikkumisen välineenä. Verkkosivu. Viitattu 19.1.2024
https://urn.fi/URN:NBN:fi:amk-2013061714254

Facebook 2023a. Kimppakyyti. Verkkosivu. Viitattu 5.12.2023
https://www.facebook.com/groups/21203240448

Facebook 2023b. Kimppakyyti Tampere – Helsinki. Viitattu 5.12.2023
https://www.facebook.com/groups/46853781203

Perry, D. 2011. Google Introduces JavaScript Alternative Called DART. Verkkosivu. Viitattu 19.1.2024
https://www.tomshardware.com/news/google-dart-programming-language-javascript,13660.html

Statista 2023. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022. Verkkosivu. Viitattu 5.12.2023
https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

Sneath, T. 2022. Introducing Flutter 3. Verkkosivu. Viitattu 5.12.2023
https://medium.com/flutter/introducing-flutter-3-5eb69151622f

Material n.d. Get Started. Verkkosivu. Viitattu 19.1.2024
https://m3.material.io/get-started

The Droids on Roids 2023. Flutter vs React Native – Which is Better for Your Project? Verkkosivu. Viitattu 19.1.2024.
https://www.thedroidsonroids.com/blog/flutter-vs-react-native-comparison

Flutter n.d. Add interactivity to your Flutter app. Verkkosivu. Viitattu 5.1.2024
https://docs.flutter.dev/ui/interactivity

Firebase n.d. Firebase Realtime Database. Verkkosivu. Viitattu 5.1.2024
https://firebase.google.com/docs/database

Pub.dev n.d. Google Maps for Flutter. Verkkosivu. Viitattu 5.1.2024
https://pub.dev/packages/google_maps_flutter

Google n.d. Google Maps Platform Pricing. Verkkosivu. Viitattu 5.1.2024
https://developers.google.com/maps/billing-and-pricing/pricing#routes-basic

Firebase. Add Firebase to your Flutter app. Verkkosivu. Viitattu 19.1.2024
https://firebase.google.com/docs/flutter/setup?platform=android

Dartcode.org n.d. Using --dart-define in Flutter. Verkkosivu. Viitattu 20.1.2024
https://dartcode.org/docs/using-dart-define-in-flutter/

Filho, G. 2021. How to setup dart-define for keys and secrets on Android and iOS in Flutter apps. Verkkosivu. Viitattu 20.1.2024 https://medium.com/flutter-community/how-to-setup-dart-define-for-keys-and-secrets-on-android-and-ios-in-flutter-apps-4f28a10c4b6c

Tebest, T. 2022. Kuntarajat 2021. Verkkosivu. Viitattu 20.1.2024 https://github.com/teelmo/geodata/blob/master/geojson/Kuntarajat%202021.geojson

Sunday, D. 2012. Inclusion of a point in a polygon. Verkkosivu. Viitattu 21.1.2024 http://profs.ic.uff.br/~anselmo/cursos/CGI/slidesNovos/Inclusion%20of%20a%20Point%20in%20a%20Polygon.pdf

Adedamola, A. 2023. Decode the google encoded string using Encoded Polyline Algorithm Format. Verkkosivu. Viitattu 21.1.2024 https://github.com/Dammyololade/flutter_polyline_points/blob/master/lib/src/utils/polyline_decoder.dart