OULUN AMMATTIKORKEAKOULU

Johannes Törmänen

RYHMÄNHALLINTA MOBIILISOVELLUKSEN KEHITTÄMINEN FLUTTER-TEKNIIKALLA

RYHMÄNHALLINTA MOBIILISOVELLUKSEN KEHITTÄMINEN FLUTTER-TEKNIIKALLA

Johannes Törmänen
Opinnäytetyö
Syksy 2022
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, Ohjelmistokehitys

Tekijä: Johannes Törmänen
Opinnäytetyön nimi: Ryhmänhallinta mobiilisovelluksen kehittäminen Flutter-tekniikalla
Työn ohjaaja: Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Syksy 2022 Sivumäärä: 24

Tämän opinnäytetyön tarkoituksena oli perehtyä Googlen kehittämään avoimen lähdekoodin käyttöliittymäohjelmistokehityspakettiin Flutter, sekä luoda perehtymisen jälkeen kyseisellä ohjelmistokehityspaketilla mobiilisovellus, joka hyödyntää myös MySQL-tietokantaa ja Node.js -palvelimen tarjoamia ohjelmointirajapintoja.

Opinnäytetyö keskittyi Flutter-käyttöliittymäohjelmistokehityspakettiin ja tämän eri ominaisuuksien hyödyntämiseen sovelluksessa. Opinnäytetyössä käytettiin useita työkaluja työn eri osa-alueisiin. Mobiilisovelluksen kehityksessä käytössä Android Studio sekä Flutter SDK, MySQL-tietokannan luomisessa käytettiin MySQL Workbench -ohjelmaa ja Node.js -palvelimen luonnissa käytössä oli Visual Studio Code.

Kehitetty mobiilisovellus oli ryhmänhallintasovellus, jolla käyttäjä voi luoda ryhmiä, lisätä niihin jäseniä ja luoda ryhmälle tapahtumia, joihin ryhmän jäsenet voivat ilmoittaa osallistumisestaan. Tärkeimpiä ominaisuuksia olivat käyttäjän luominen, sisäänkirjautuminen, ryhmän luominen, tapahtumien luominen sekä tapahtumien näyttäminen listana sekä kalenterinäkymässä.

Asiasanat: Flutter, Node.js, Express.js

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author: Johannes Törmänen
Title of thesis: Group management mobile application development using Flutter technology
Supervisor: Jouni Juntunen
Term and year when the thesis was submitted: Autumn 2022
Number of pages: 24

The purpose of this thesis was to learn about the open-source user framework Flutter developed by Google, and to create a mobile application with this framework. The mobile application also uses MySQL database and Node.js backend.

The thesis focused on the Flutter framework and the use of its various features in the application. The thesis used several tools for several aspects of the work. Android Studio and Flutter SDK were used for the development of the mobile application, MySQL Workbench was used to create the MySQL database and Visual Studio Code was used to create the Node.js server.

The mobile application developed was a group management application that allows the user to create groups, add members to them and create events for the group in which group members can register their participation. The major features were user creation, login, creating a group, creating events, and displaying events as a list and in a calendar view.

Keywords: Flutter, Node.js, Express.js

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on perehtyä Googlen kehittämään avoimen lähdekoodin käyttöliittymäohjelmistokehityspakettiin Flutter, sekä luoda perehtymisen jälkeen kyseisellä ohjelmistokehityspaketilla mobiilisovellus, joka hyödyntää myös MySQL-tietokantaa ja Node.js -palvelimen tarjoamia ohjelmointirajapintoja. Tavoitteena on saada aikaan toimiva sovellus, sekä oppia kehittämään mobiilisovelluksia Flutterilla.

Kehitettävä mobiilisovellus tulee olemaan ryhmänhallintasovellus, jolla käyttäjä voi luoda ryhmiä, lisätä niihin jäseniä ja luoda ryhmälle tapahtumia, joihin ryhmän jäsenet voivat ilmoittaa osallistumisestaan. Tärkeimpiä ominaisuuksia tulevat olemaan käyttäjän luominen, sisäänkirjautuminen, ryhmän luominen, tapahtumien luominen sekä tapahtumien näyttäminen ensiksi listana, ja myöhemmin mahdollisesti kalenterinäkymässä. Sovellukseen tulee myös mahdollisesti ryhmänsisäinen keskusteluominaisuus, mikäli muut ominaisuudet saadaan tehtyä valmiiksi.

Opinnäytetyö tulee keskittymään Flutter-käyttöliittymäohjelmistokehityspakettiin ja tämän eri ominaisuuksien hyödyntämiseen sovelluksessa. Flutter valikoitui käytettäväksi teknologiaksi mielenkiinnosta alustariippumatonta sovelluskehitystä kohtaan, sekä halusta oppia kyseistä teknologiaa. Node.js sekä MySQL ovat jo tuttuja työkaluja, joten niiden opiskeluun ja käyttöönottoon ei kulu ylimääräistä aikaa.

Opinnäytetyössä tullaan käyttämään useita työkaluja työn eri osa-alueisiin. Mobiilisovelluksen kehityksessä käytössä tulee olemaan Googlen ja JetBrainsin luoma mobiilisovelluskehitysalusta Android Studio sekä Flutter SDK (Software Development Kit) (1). Lisäksi sovellusta testataan Android Studion Android Emulaattorilla, joka simuloi Android laitetta tietokoneella ilman fyysistä laitetta (2). Node.js palvelimen luontiin käytössä tulee olemaan avoimen lähdekoodin monialustainen tekstieditori Visual Studio Code. MySQL-tietokannan suunnitteluun ja toteutukseen käytetään MySQL Workbenchia, joka on visuaalinen tietokannan suunnittelutyökalu.

2 KÄYTETTÄVÄT TEKNOLOGIAT JA TYÖKALUT

2.1 Flutter ja Dart

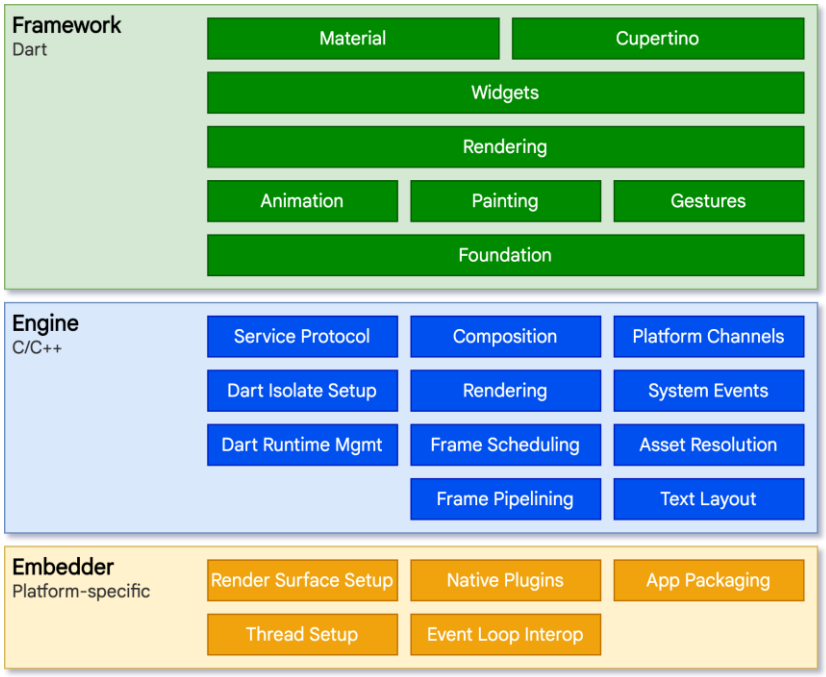
Flutter on Googlen kehittämä avoimen lähdekoodin käyttöliittymäohjelmistokehityspaketti, jonka ohjelmointikielenä toimii Dart. Sitä käytetään sovellusten kehittämiseen eri alustoille, kuten Androidille, iOS:lle tai Windowsille, yhdellä koodikannalla. Yhden koodikannan merkittävä etu on se, ettei samaa sovellusta tarvitse tehdä useasti eri alustoille. Flutterin ensimmäinen versio julkaistiin toukukuussa 2017 ja Flutter 1.0 julkaistiin joulukuussa 2018. (3.)

Dart on Googlen kehittämä ilmainen ja avoimen lähdekoodin asiakasohjelmaoptimoitu ohjelmointikieli, jonka tarkoitus on tarjota tuottoisin ohjelmointikieli monialustakehitykseen. Dart on kielenä tyyppisuojattu, se hyödyntää staattista tyypintarkistusta varmistaakseen, että muuttujan arvo vastaa aina sen staattista tyyppiä. Dart on myös null-safe kieli, eli esimerkiksi muuttujat eivät voi saada arvoa null, ellei sitä erikseen määritellä mahdolliseksi. Dart sisältää laajan valikoiman eri kirjastoja, jotka tarjoavat olennaiset asiat monille jokapäiväisille ohjelmointitehtäville. (4.)

Flutterin suurimpia etuja kehittäjälle ovat nopeampi koodin kirjoittaminen ja jakaminen, joustavat pienoisohjelmat (widgets), yksinkertainen käytettävyys, työkalujen ja resurssien laaja valikoima, tehokas tuki sekä tarve vähemmälle testaukselle. Yhden koodikannan hyödyntäminen usealla eri alustalla, kuten Android ja iOS, mahdollistaa nopeamman kehityksen, sekä runsas ja koko ajan kasvava pienoisohjelmakirjasto tarjoavat käyttövalmiita ratkaisuja sekä perus- että edistyneisiin ominaisuuksiin. Flutterin haittapuolena sovelluksista tulee melko suurikokoisia, sovelluksen vähimmäiskoko on yli 4 megatavua, eli suurempi kuin natiivi Java sovelluksen (539 kilotavua) tai Kotlin sovelluksen (550 kilotavua). (5. ja 6.)

Flutter on rakenteeltaan suunniteltu laajennettavaksi, kerroksittaiseksi järjestelmäksi, joka koostuu sarjoista itsenäisiä kirjastoja, jotka nojautuvat alempaan kerrokseen. Yhdelläkään kerroksella ei ole etuoikeutettua pääsyä alempaan kerrokseen ja jokainen ohjelmistokehyksen taso on suunniteltu olemaan välttämätön ja vaihdettava. (7.)

Kehittäjät hyödyntävät Flutteria tavallisesti Flutter-ohjelmistokehyksen kautta, joka tarjoaa modernin ja reaktiivisen ohjelmistokehyksen. Se sisältää laajan kirjaston eri toiminnallisuuksille, alustoille ja asetelmille, jotka on koottu eri kerroksista. Ohjelmistokehyksen kerroksittainen rakenne pohjalta pinnalle on perustukselliset luokat, tärkeimmät rakenteelliset palvelut kuten animaatiot ja eleet, renderöinti kerros (rendering), pienoisohjelmakerros sekä Material- ja Cupertino-kirjasto kerros (kuva 1).

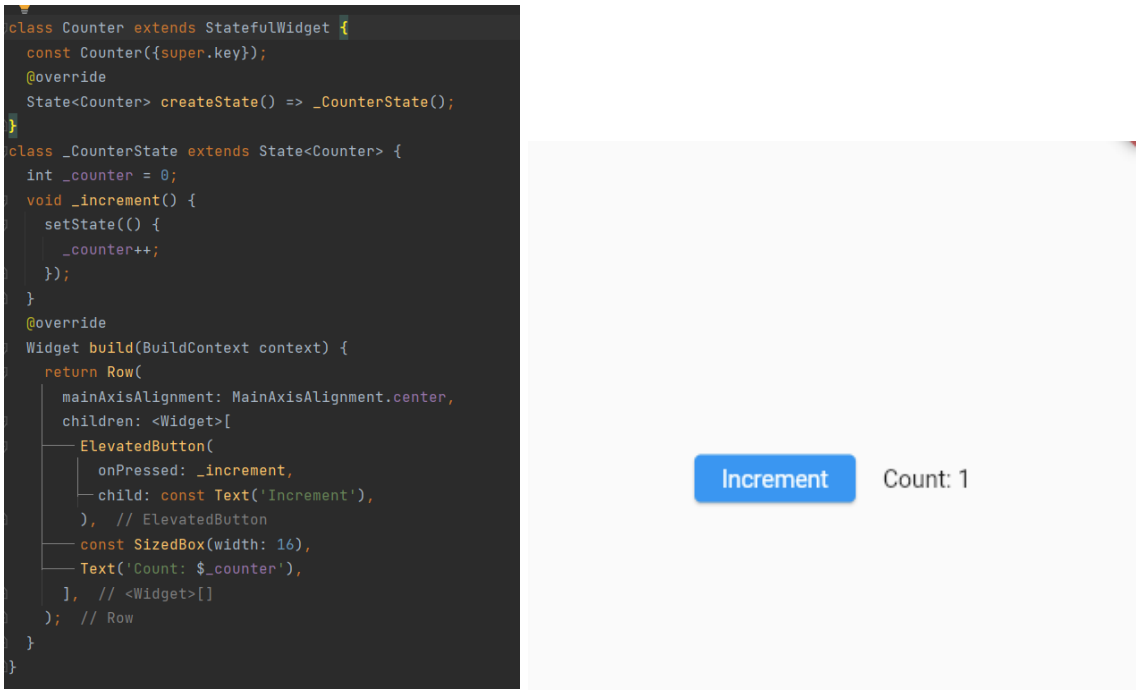


KUVA 1. Flutterin arkkitehtuurinen rakenne (8).

Ohjelmistokehyksen alin kerros, foundation-kirjasto, tarjoaa alimman tason yleisluokat ja –ohjelmat, joita kaikki muut tasot hyödyntävät. Seuraavan tason animation-kirjasto tarjoaa perustan animaatioiden toteuttamiselle Flutterissa. Muut ylemmän tason kirjastot hyödyntävät tätä perustaa tarjotakseen edistyksellisempiä ja monimutkaisempia animaatioita sovellukselle. Flutter esittää animaatiot arvoina, jotka muuttuvat ajan myötä. Tämä arvo voi olla tyypiltään mitä tahansa, kuten esimerkiksi sovelluksen taustaväri Color-luokasta, joka vaihtuu alkuperäisestä väristä sujuvasti toiseksi väriksi (9). Saman tason gestures-kirjasto tarjoaa työkalut käyttäjän eri eleiden tunnistamiseen (10). Painting-kirjasto sisältää valikoiman eri luokkia, jotka tarjoavat eri tapoja värjätä asioita (11). Widgets-kirjasto hyödyntää rendering-kirjaston RenderObject-luokan hierarkiaa muun muassa toteuttaakseen sen asetelman (layout). Yleensä ainut hetki, jolloin kehittäjä joutuu olemaan vuorovaikutuksessa RenderObject-luokan kanssa, on kun hän korjaa asetelmavirheitä (12.) Material- ja Cupertino-kirjastot tarjoavat laajan valikoiman eri

pienoisohjelmia. Cupertino-kirjasto on erikoistunut iOS -suunnitelukieleen. Sovelluksissa, joita käytetään usealla eri alustalla, suositellaan käytettävän Material-kirjastoa (13).

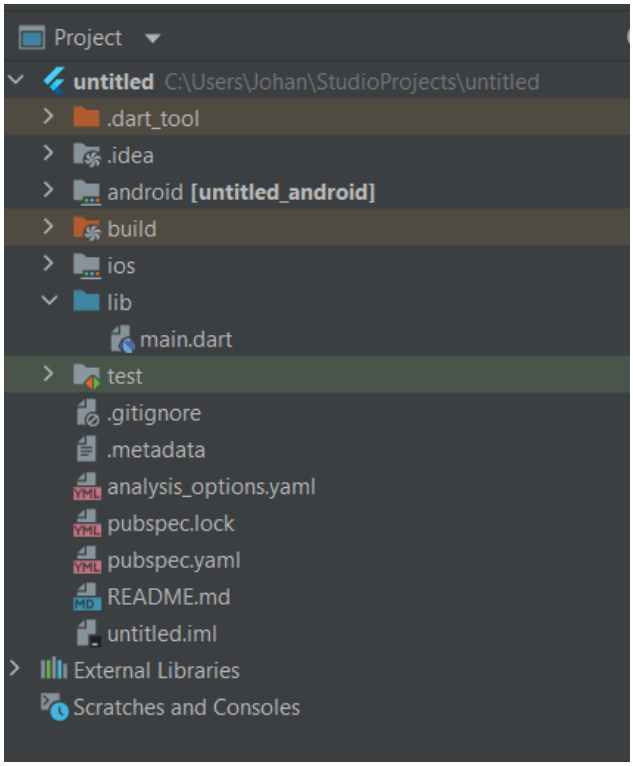
Flutterin keskeinen idea on rakentaa sovelluksen käyttöliittymä pienoisohjelmista. Pienoisohjelmat kuvaavat sen, miltä niiden näkymän kuuluisi näyttää hallitsevan konfiguraation ja tilan (state) mukaan. Pienoisohjelman tilan muuttuessa se kokoaa kuvauksensa uudelleen. Flutterin peruspienoisohjelmia ovat muun muassa Text, Row, Column ja Stack -pienoisohjelmat. Text antaa nimensä mukaisesti kehittäjän luoda tyyliteltyjä tekstejä sovellukseen. Row ja Column mahdollistavat joustavan asetelman luomisen sekä horisontaalisesti että vertikaalisesti. Stack mahdollistaa eri pienoisohjelmien asettamisen toisten pienoisohjelmien päälle. Edellä mainitut peruspienoisohjelmat ovat tilattomia pienoisohjelmia (stateless widget). Monimutkaisempien ominaisuuksien luomiseksi, kuten esimerkiksi tekstin muuttamisen nappia painamalla toiseksi tekstiksi, sovelluksilla on usein joku tila. Tämän toteuttamiseksi Flutterissa on tilalliset pienoisohjelmat (StatefulWidgets), jotka osaavat luoda olioita tilan säilyttämiseen. Kuvassa 2 on esitelty yksinkertainen sovellus, jossa painiketta painamalla tekstin luku kasvaa aina yhdellä. Sovelluksen _increment() funktio kutsuu setState() funktiota, joka kertoo Flutter ohjelmistokehykselle, että sovelluksen tilassa on tapahtunut muutos ja sovelluksen build()-menetelmä on ajettava uudelleen. Käyttäjälle tämä näkyy tekstin muuttumisena. (14.)



KUVA 2. Yksinkertainen StatefulWidget -sovellus.

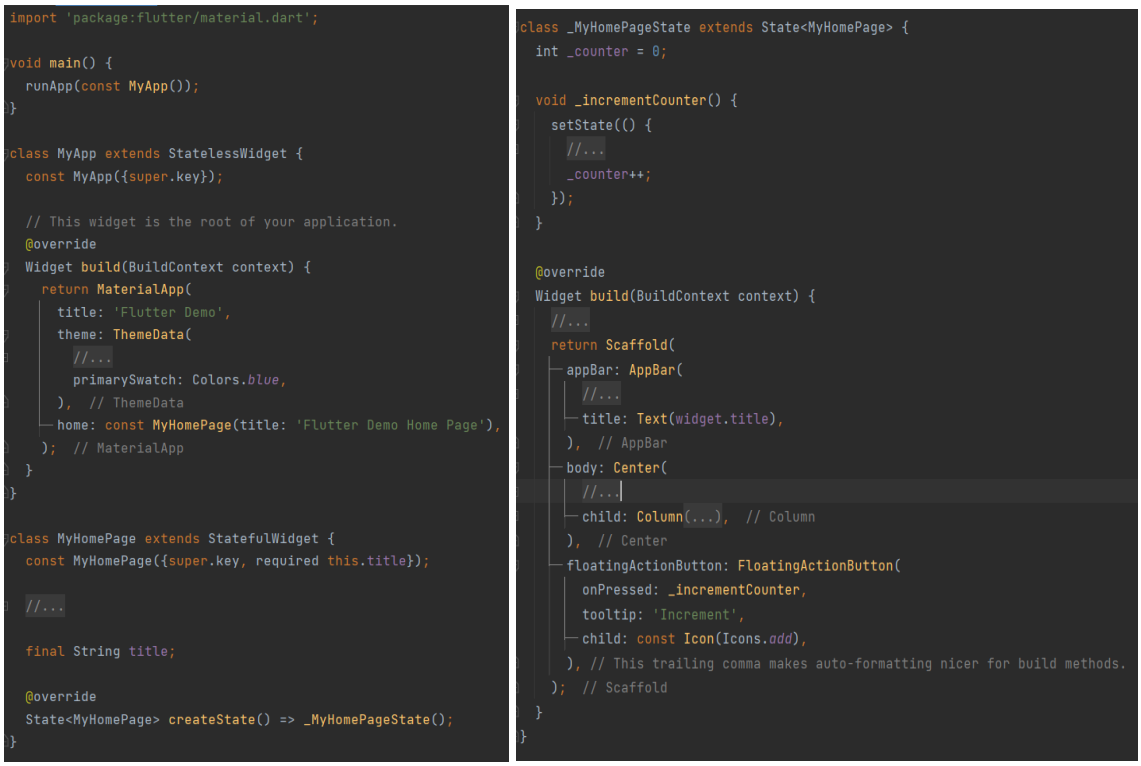
2.2 Flutter-projekti Android Studiossa

Android Studion käyttäminen Flutter-sovelluksen kehittämisessä vaatii Flutter- ja Dart-laajennuksien asentamisen Android Studioon. Android Studion generoima Flutter-projekti sisältää useita eri tiedostoja (kuva 3). Suurin osa sovelluksen muokattavasta koodista sijaitsee lib-kansion .dart-tiedostoissa, kuten esimerkiksi main.dart -tiedostossa (kuva 4).



KUVA 3. Yksinkertaisen Flutter-projektin rakenne.

Kuvan 4 esimerkkiprojekti luo Material-applikaation, joka perii StatelessWidget-luokan ja näin itse applikaatiosta tulee pienoisohjelma. Material on visuaalinen suunnittelu kieli, josta Flutter tarjoaa laajan valikoiman eri Material-pienoisohjelmia. Flutterissa lähes kaikki muodostuu pienoisohjelmista, kuten kuvan 4 esimerkkiprojektin pienoisohjelma Scaffold, joka puolestaan koostuu muun muassa AppBar-, Text- ja Center-pienoisohjelmista. Pienoisohjelmien tärkein tehtävä on tarjota build()-menetelmä, joka määrittelee kuinka pienoisohjelma näytetään muiden alemman tason pienoisohjelmien suhteen. (15.)



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        //...
        primarySwatch: Colors.blue,
      ),  // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );  // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  //...

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
```

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      //...
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    //...
    return Scaffold(
      appBar: AppBar(
        //...
        title: Text(widget.title),
      ),  // AppBar
      body: Center(
        //...
        child: Column(...),  // Column
      ),  // Center
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );  // Scaffold
  }
}
```

KUVA 4. Yksinkertaisen Flutter projektin main.dart -tiedosto.

Flutter on kielenä erittäin joustava ja projektin organisoimiseen on useita eri tapoja ja vaihtoehtoja. Yleisiä tapoja ja suunnittelumalleja organisoida sovelluksen arkkitehtuuri ovat muun muassa Model-View-ViewModel -malli (MVVM), Model-View-Controller -malli (MVC) ja BloC -malli, joka on MVC-mallin variaatio. (16.)

2.3 Node.js REST API

Node.js on ilmainen avoimen lähdekoodin monialustainen Googlen V8 JavaScript -moottorilla rakennettu JavaScript-ajoympäristö (17). Se on suunniteltu rakentamaan skaalautuvia verkkoapplikaatioita . Useat kolmannen osapuolen kirjastot hyödyntävät Node.js ja helpottavat sen käyttöä, kuten Express, joka tarjoaa yhden helpoimmista ja tehokkaimmista tavoista luoda verkkopalvelimen (18). Express-kirjastoa käytetään verkko- tai mobiilisovellusten taustapalveluiden (backend) luomiseen, kuten ohjelmistorajapintojen luomiseen tietokantaa varten. Kuvassa 5 on esitelty Express-kirjastoa hyödyntävän REST API -ohjelmistorajapintapalvelimen alustaminen. Express-kirjaston lisäksi kyseinen esimerkki hyödyntää muun muassa MySQL-kirjastoa tietokannan kanssa kommunikoimiseen.

```
require('dotenv').config()
const express = require('express')
const app = express()
const mysql = require('mysql')
const bodyParser = require('body-parser')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const async = require('async')

/*--------------- SETUP ---------------------------*/
const PORT = 3001

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
app.use(express.json())

// MySQL connection
const connection = mysql.createConnection({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_DATABASE,
    charset : 'utf8mb4'
})
connection.connect(function (err) {
    if (err) throw err
    console.log("Connected to database")
})

app.listen(PORT, () => {
    console.log(`Server running at ${PORT}`)
})
```

KUVA 5. Yksinkertainen Node.js express sovellus.

Uusien kolmannen osapuolien kirjastojen ja ohjelmistopakettien lisääminen omaan Node.js sovellukseen on helppoa eri paketinhallintatyökalujen avulla. Näistä paketinhallintatyökaluista suurin ja suosituin on vuonna 2014 perustetun ja 2020 Githubin hankkiman yrityksen npm, Inc:n ylläpitämä npm (19). Se on myös Node.js:n oletuspaketinhallintatyökalu. Npm on komentorivityökalu, jonka avulla käyttäjä voi ladata ja asentaa halutun ohjelmistopaketin omaan Node.js sovellukseensa. Esimerkiksi Express.js paketin lisääminen tapahtuisi komentoriviltä komennolla npm install express.

REST API on kirjainlyhenne sanoista REpresentational State Transfer Application Programming interface. Se on HTTP-protokollaa ja sen standardimetodeja hyödyntävä ohjelmistorajapinta

arkkitehtuurityyli. Ohjelmointirajapinta mahdollistaa eri sovellusten kommunikoinnin keskenään. Se voi olla joko sovelluksen sisäiseen käyttöön tarkoitettu tai muille sovelluksille tarkoitettu ulkoinen rajapinta (20). REST arkkitehtuurityylissä datasta ja toiminnallisuuksista puhutaan resursseina, ja niihin pääsee käsiksi Uniform Resource Identifierien (URI) kautta. Kuvassa 6 on kuvattu Express-kirjastolla luoto REST API -arkkitehtuurityylin resurssi, joka hakee HTTP-pyynnöllä toimitetun groupId-parametrin avulla tietokannasta dataa, ja palauttaa ne HTTP-vastauksella pyynnön lähettäjälle.

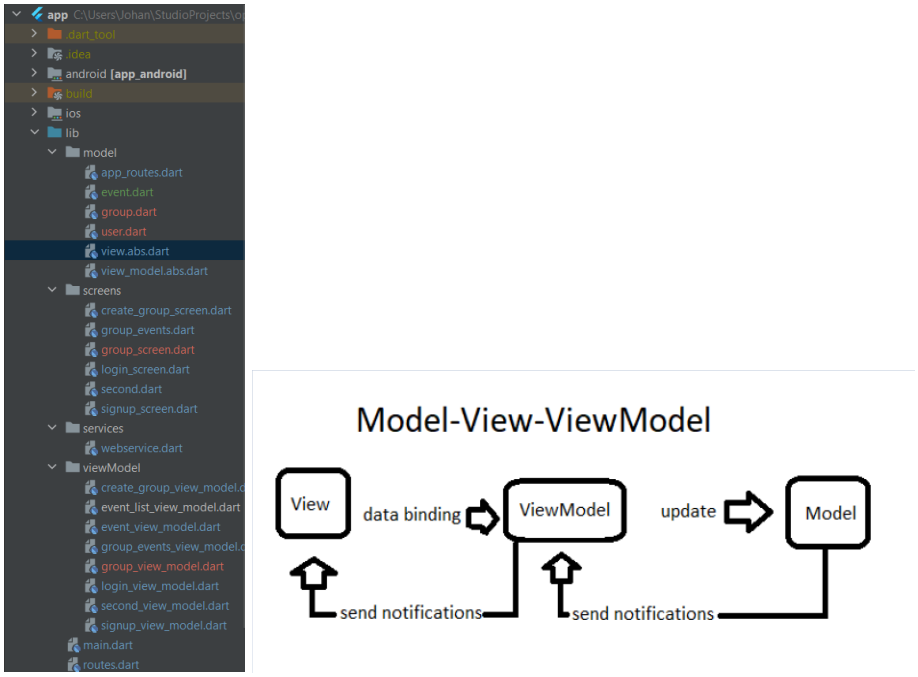
```
//sends members of a searched group
app.post('/api/group/members', (req, res) => {
    try {
        let groupId = req.body.groupId
        const sqlGetGroupMembers = 'CALL getGroupMembers(?)'



        connection.query(sqlGetGroupMembers, [groupId], (err, results, fields) => {
            if (err) throw err
            if (results.length > 0) {
                res.status(200).json({
                    "groupMembers":
                        results[0]
                })
            }
            else {
                res.status(200).send('No members found with this group')
            }
        })

    } catch {
        res.status(500).send()
    }
})
```

KUVA 6. Esimerkki REST API arkkitehtuurityylin resurssista.

3 SOVELLUS JA SEN ERI NÄKYMÄT

3.1 Sovelluksen rakenne

Sovelluksen graaffisen käyttöliittymän (frontend) arkkitehtuurin selkeyttämiseksi päädyin toteuttamaan Model-View-ViewModel (MVVM) -suunnittelumallia, jossa sovelluksen logiikka ja käyttöliittymä on erotettu toisistaan (kuva 7). Sen ovat kehittäneet Microsoftin arkkitehdit Ken Cooper ja John Gossman (21). Tässä suunnittelumallissa View, eli näkymä, sisältää sovelluksen käyttäjälle näkyvät elementit, kuten käyttöliittymä, animaatiot ja tekstit. Tämän opinnäytetyön sovelluksessa suunnittelumallin View esiintyy nimellä Screen. Model sisältää sovelluksen logiikan, jonka ViewModel noutaa tarvittaessa käyttäjän näkymässä antaman syötteen perusteella.



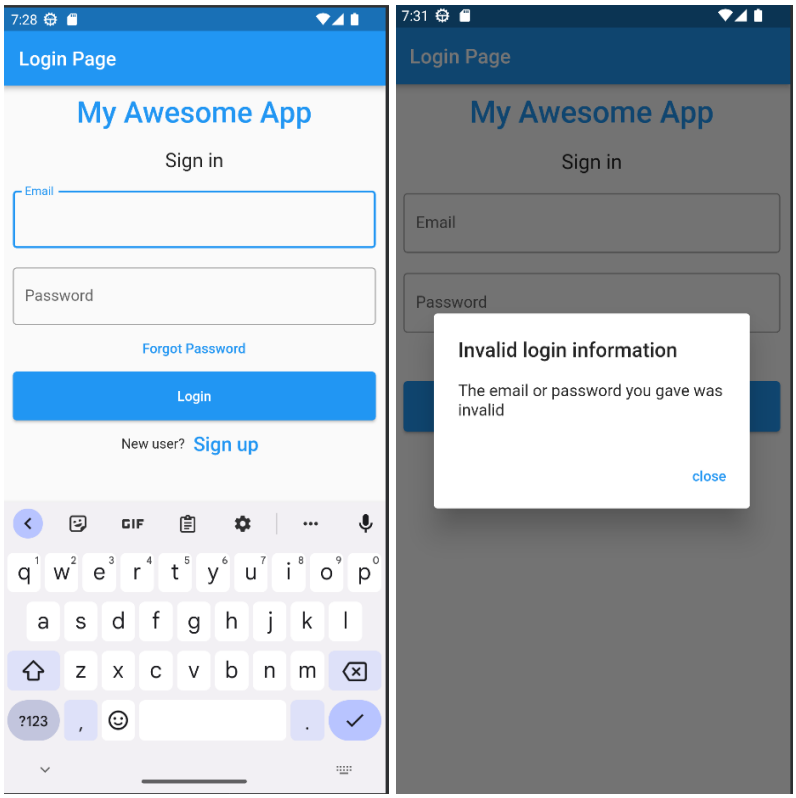
KUVA 7. Sovelluksen arkkitehtuurinen malli.

Tämän opinnäytetyön sovellus koostuu eri näkymistä ja niitä vastaavista ViewModel-luokista, Model-luokasta sekä webservice-luokasta, joka vastaa tietokannan kanssa kommunikoimisesta REST API -ohjelmistorajapinnan välityksellä. Sovelluksen ensimmäinen näkymä on sisäänkirjautumisnäkymä, josta onnistuneen sisäänkirjautumisen jälkeen avautuu sovelluksen päänäkymä. Vaihtoehtoisesti sisäänkirjautumisnäkymästä voi siirtyä käyttäjän luonti näkymään,

mikäli sovelluksen käytön vaatima käyttäjää ei vielä ole. Päänäkymässä käyttäjä näkee omat tapahtumansa ja ryhmänsä, sekä voi hallinnoida näitä.

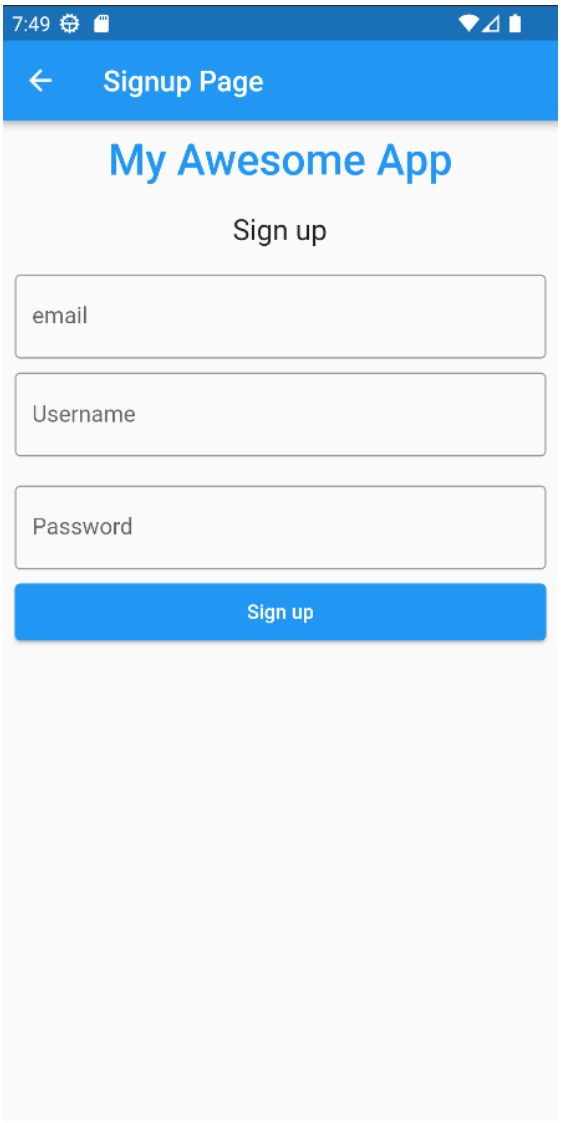
3.2 Sisäänkirjautumis- ja käyttäjän luonti näkymät

Sovelluksen käynnistyttyä ensimmäinen näkymä on sisäänkirjautumisnäkymä (kuva 8), jossa käyttäjä voi kirjautua sisään, luoda uuden käyttäjän tai palauttaa unohtuneen salasanan. Sisäänkirjautumisnäkymä mudostuu kahdesta Scaffold-pienoisohjelmasta, joka puolestaan sisältää AppBar- (Login Page -otsikkoteksti ja sen sininen taustapalkki), Center- ja ListView-pienoisohjelmista. Center-pienoisohjelma keskittää ListView-pienoisohjelman keskelle ruutua. ListView-pienoisohjelma sisältää listan muutamasta rivistä tekstejä, kaksi TextField-pienoisohjelmaa sähköpostiosoitteen ja salasanan syöttöä varten sekä painikkeita eri toiminnoille. Käyttäjän syötettyä oikean sähköpostiosoitteen ja salasanan, vertaa sovellus niitä tietokannassa oleviin ja mikäli tiedot vastaavat, siirtyy sovellus eteenpäin sovelluksen päännäkymään. Virheellisestä sisäänkirjautumisinformaatiosta käyttäjää informoidaan AlertDialog()-ponnahdusikkunalla.



KUVA 8. Sovelluksen sisäänkirjautumisnäkymä.

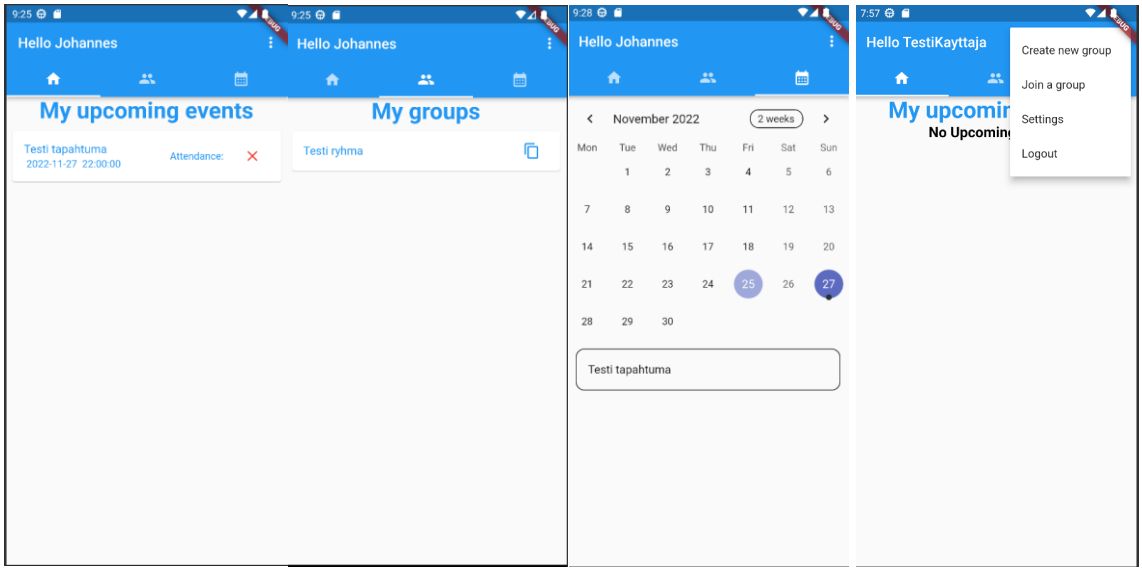
Sovelluksen käyttäminen edellyttää käyttäjän luomista, ja sitä varten sovelluksessa on oma näkymänsä (kuva 9). Tämä näkymä koostuu kahdesta tekstirivistä, kolmesta TextField()-pienoisohjelmasta sekä ElevatedButton()-pienoisohjelman painikkeesta. Käyttäjältä vaaditaan sähköpostiosoite, jota käytetään sisäänkirjautumiseen, käyttäjätunnus sekä salasana. Samalla sähköpostiosoitteella ei voi luoda kuin yhden käyttäjän. Onnistuneen käyttäjän luonnin jälkeen sovellus siirtyy eteenpäin sovelluksen päänäkymään.



KUVA 9. Sovelluksen käyttäjän luonti näkymä.

3.3 Sovelluksen päänäkymä

Sovelluksen päänäkymä (kuva 10) muodostuu MaterialApp()-pienoisohjelmasta, joka koostuu kolmesta TabBarView()-pienoisohjelman välilehdestä sekä näkymän yläosan AppBar()-pienoisohjelmasta, joka koostuu tervehdys tekstistä sekä kolmen pisteen ikonipainikkeesta. Tästä painikkeesta avautuu lisävalikko, josta voi luoda uuden ryhmän, liittyä olemassa olevaan ryhmään tai kirjautua ulos sovelluksesta.



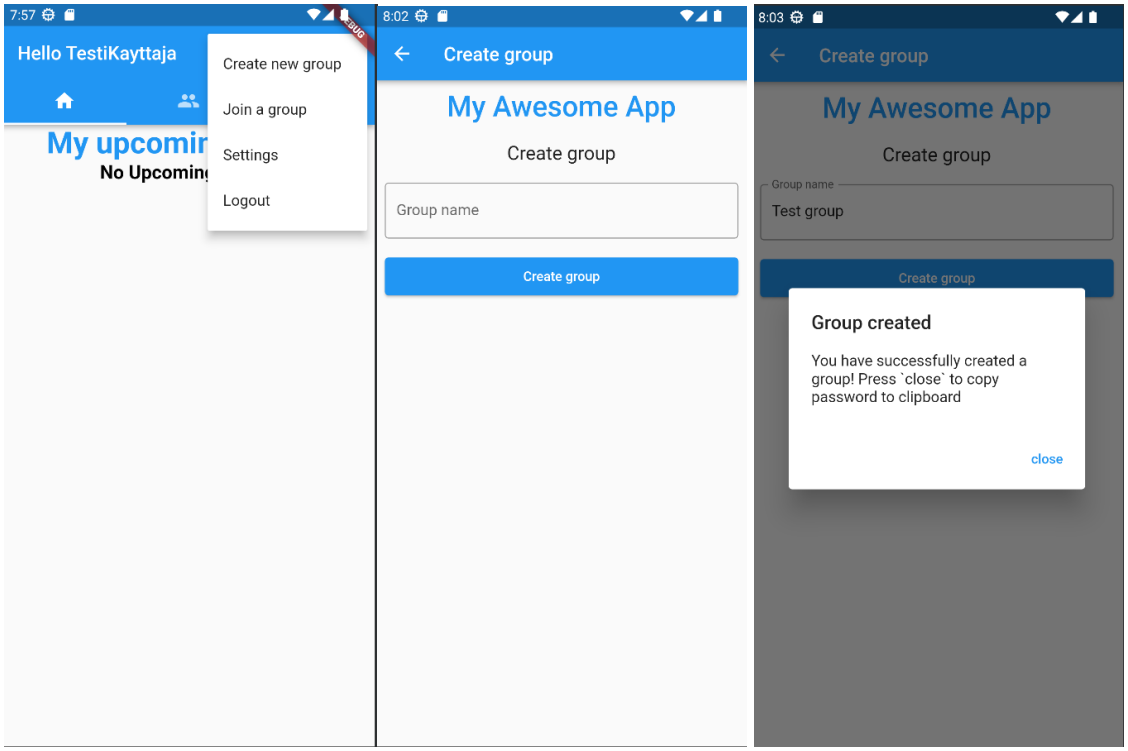
KUVA 10. Sovelluksen päänäkymän kolme välilehteä sekä lisävalikko.

Päänäkymän ensimmäisellä välilehdellä on ’My upcoming events’ otsikkorivi sekä listattuna ListView()-pienoisohjelmaan käyttäjän tapahtumat. Mikäli tapahtumia ei ole, on tapahtumalistan tilalla teksti ’No upcoming events’. Toisella välilehdellä on ’My groups’ otsikkoteksti sekä listattuna ListView()-pienoisohjelmaan käyttäjän ryhmät. Mikäli ryhmiä ei ole, on ryhmälistan tilalla teksti ’No groups’. Kolmannella välilehdellä on kalenterinäkymä, jossa näkyvät käyttäjän mahdolliset tapahtumat.

3.4 Ryhmät

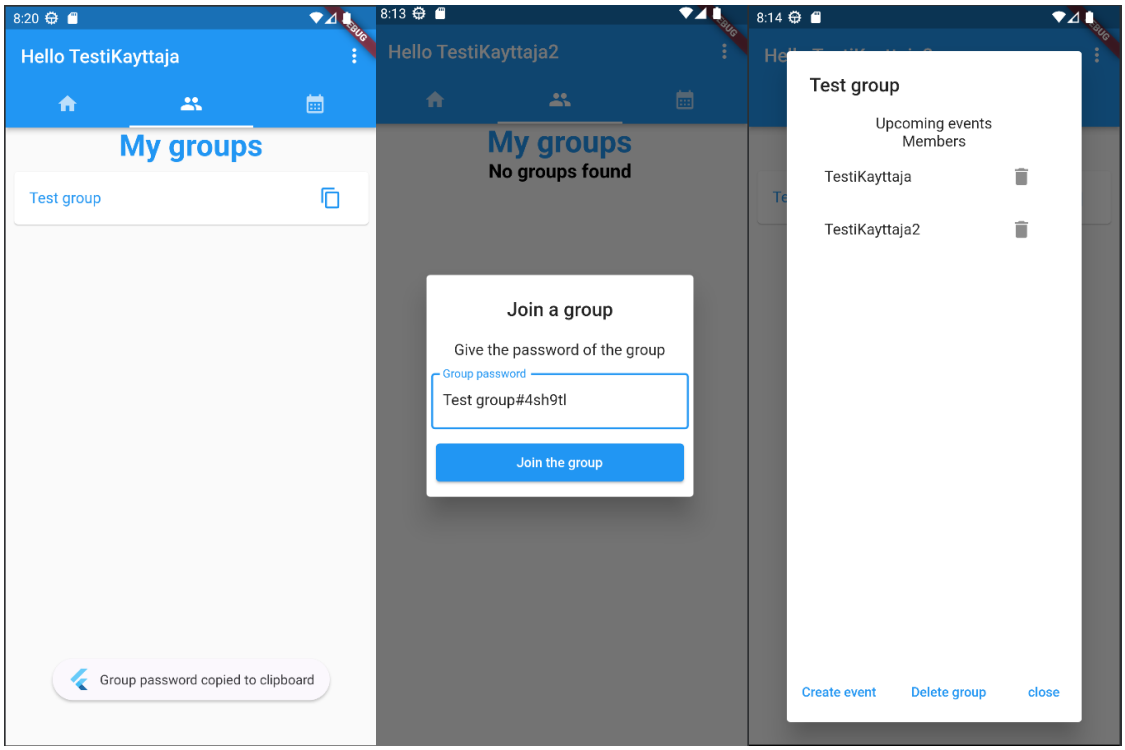
Sovelluksen tapahtumat ovat aina jonkin ryhmän tapahtumia, joten luodakseen ja nähdäkseen tapahtumia, tulee käyttäjän liittyä ryhmään tai luoda uusi ryhmä. Uuden ryhmän luominen tai olemassa olevaan ryhmään liittyminen tapahtuu päänäkymän yläosan AppBar()-pienoisohjelman lisävalikosta. Tämän valikon ’Create new group’ -valinta avaa uuden näkymän, jossa ryhmän

luominen tapahtuu (kuva 11). Ryhmän luominen vaatii vain nimen antamisen ryhmälle. Sovellus luo ryhmälle salasanan, joka koostuu ryhmän nimestä ja siihen lisätystä satunnaisgeneroidusta merkkiosiosta. Kuvan 11 esimerkkiryhmän salasanaksi muodostui ’ Test group#4sh9tl’. Ryhmän luominen lisää käyttäjän automaattisesti tähän ryhmään.



KUVA 11. Ryhmän luomisen vaiheet.

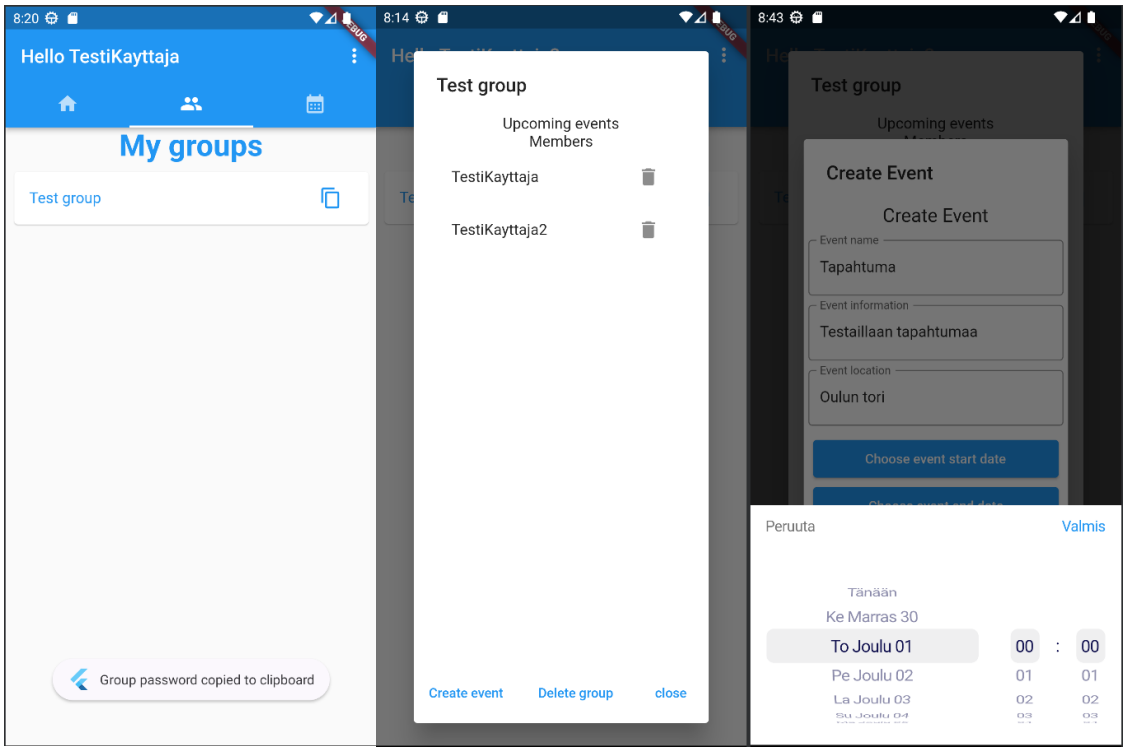
Liittyäkseen jo olemassa olevaan ryhmään käyttäjä tarvitsee ryhmän salasanan, jonka joku jo ryhmään kuuluva voi kopioda leikepöydälle ja jakaa liittyvälle jäsenelle (kuva 12). Liityttyään ryhmään voi käyttäjä luoda ryhmälle tapahtumia, poistaa ryhmästä ryhmän jäseniä tai poistaa koko ryhmän.



KUVA 12. Ryhmään liittyminen.

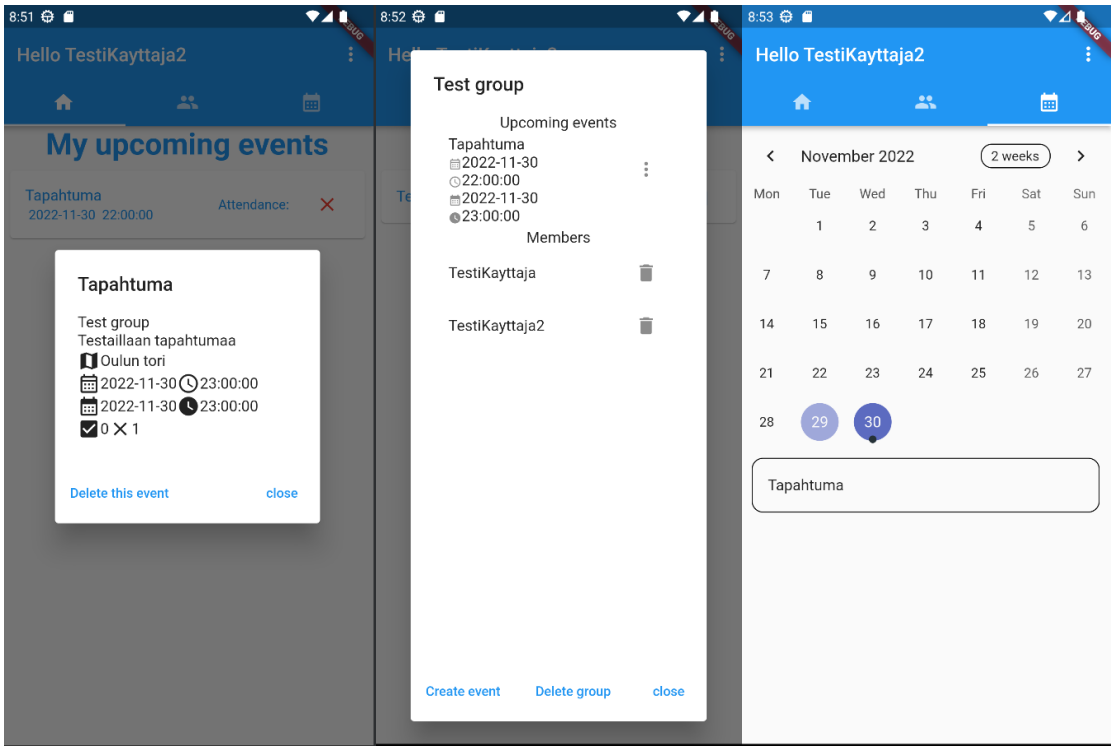
3.5 Tapahtumat

Tapahtuman luominen tapahtuu ryhmänäkymästä (kuva 13). Tapahtuman luominen vaatii tapahtuman nimen, lisäinformaation tai tarkemman kuvauksen, sijainnin sekä aloitus- ja lopetusajankohdan.



KUVA 13. Tapahtuman luominen.

Ryhmälle luodut tapahtumat näkyvät kaikille ryhmän jäsenille päänäkymän ensimmäisen välilehden listalla, toisen välilehden ryhmälistalta avatusta ryhmän näkymässä listalla, sekä kolmannen välilehden kalenterinäkymässä, joka on luotu käyttäen valmista TableCalendar -kalenteripienoisohjelmaa. Päänäkymän ensimmäisen välilehden listalta voi vaihtaa omaa osallistumistilaansa sekä poistaa kyseisen tapahtuman (kuva 14).



KUVA 14. Tapahtuman eri näkymät.

4 POHDINTA

Tämän opinnäytetyön tarkoituksena oli perehtyä Googlen kehittämään avoimen lähdekoodin käyttöliittymäohjelmistokehityspakettiin Flutter, sekä luoda perehtymisen jälkeen kyseisellä ohjelmistokehityspaketilla mobiilisovellus, joka hyödyntää myös MySQL-tietokantaa ja Node.js -palvelimen tarjoamia REST API -ohjelmointirajapintoja. Tavoitteena oli saada aikaan toimiva sovellus, sekä oppia kehittämään mobiilisovelluksia Flutterilla.

Opinnäytetyön tavoitteet saavutettiin ja sovellukselle aluksi määritellyt tärkeimmät ominaisuudet käyttäjän luominen, sisäänkirjautuminen, ryhmän luominen, tapahtumien luominen sekä tapahtumien näyttäminen listana saatiin toteutettua. Tärkeimpien ominaisuuksien lisäksi myös kalenterinäkymä saatiin luotua. Sovelluksen jatkokehityskohteita voisi olla ryhmän jäsenille roolien, kuten järjestelmänvalvojan, luominen ettei kuka tahansa voi poistaa ryhmän jäseniä ryhmästä tai poistaa koko ryhmää. Uusia ominaisuuksia voisi olla ryhmän sisäinen keskustelu- tai pikaviestipalvelu ja sovelluksen kalenterin synkronoiminen ulkopuolisen kalenterin kanssa.

Opinnäytetyötä tehdessä opin paljon Flutter-kehityksestä. Sovelluskehitys uudella ohjelmointitekniikalla sisälsi omia pieniä haasteitaan, mutta ei mitään ylitsepääsemätöntä. Ohjelmointitekniikkana Flutter vaikutti varsin helposti omaksuttavalta ja tulen käyttämään sitä mahdollisissa tulevissa mobiilisovellusprojekteissani.

LÄHTEET

1. Flutter. Set up an editor. Hakupäivä 5.10.2022. https://docs.flutter.dev/get-started/editor?tab=androidstudio
2. Android Developers 2022. Run apps on the Android Emulator. Hakupäivä 5.10.2022. https://developer.android.com/studio/run/emulator
3. Flutter. Flutter. Hakupäivä 19.9.2022. https://flutter.dev/
4. Dart. Dart. Hakupäivä 20.9.2022. https://dart.dev/overview
5. Altexsoft. 4.8.2022. The Good and the Bad of Flutter App Development. Hakupäivä 15.11.2022. https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/
6. Hannah, John 1.12.2021. 10 Flutterin etua mobiilisovellusten kehittämisessä. Hakupäivä 20.9.2022. https://www.techlila.com/fi/benefits-of-flutter-in-mobile-app-development/
7. Flutter. Flutter arcitectural overview. Hakupäivä 22.9.2022. https://docs.flutter.dev/resources/architectural-overview
8. Flutter. Architectural layers. Valokuva. Hakupäivä 22.9.2022. https://docs.flutter.dev/assets/images/docs/arch-overview/archdiagram.png
9. Flutter. animation library. Hakupäivä 31.10.2022. https://api.flutter.dev/flutter/animation/animation-library.html
10. Flutter. gestures library. Hakupäivä 31.10.2022. https://api.flutter.dev/flutter/gestures/gestures-library.html
11. Flutter. painting library. Hakupäivä 31.10.2022. https://api.flutter.dev/flutter/painting/painting-library.html
12. Flutter. rendering library. Hakupäivä 31.10.2022. https://api.flutter.dev/flutter/rendering/rendering-library.html
13. Flutter. cupertino library. Hakupäivä 31.10.2022. https://api.flutter.dev/flutter/cupertino/cupertino-library.html
14. Flutter. Introduction to widgets. Hakupäivä 24.11.2022. https://docs.flutter.dev/development/ui/widgets-intro
15. Flutter. Write your first Flutter app, part 1. Hakupäivä 17.10.2022. https://docs.flutter.dev/get-started/codelab
16. Egan, Brian 2017. Flutter Architecture Samples. Hakupäivä 31.10.2022. https://fluttersamples.com/

17. OpenJS Foundation. Introduction to Node.js. Hakupäivä 26.9.2022. https://nodejs.dev/en/learn/
18. OpenJS Foundation 2017. Express. Hakupäivä 5.10.2022. https://expressjs.com/
19. npm. About npm. Hakupäivä 5.10.2022. https://www.npmjs.com/about
20. Korhonen, Pekka 10.1.2018. Pieni API-sanakirja. Hakupäivä 15.11.2022. https://www.cgi.com/fi/fi/blogi/pieni-api-sanakirja
21. TechTarget Contributor 2019. Model-View-ViewModel (MVVM). Hakupäivä 17.10.2022. https://www.techtarget.com/whatis/definition/Model-View-ViewModel