VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Vili Nurmi

ANDROID-ASKELMITTARISOVELLUS

Tekniikka
2020

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Vili Nurmi
Opinnäytetyön nimi	Android-askelmittarisovellus
Vuosi	2020
Kieli	suomi
Sivumäärä	47
Ohjaaja	Pirjo Prosi

Tämän opinnäytetyön tarkoituksena oli luoda askelmittarisovellus Android-käyttöjärjestelmää tukeville laitteille. Kehittämällä sovelluksia Android-käyttöjärjestelmälle voi tavoittaa jopa kolme neljäsosaa kaikista mobiililaitteista.

Sovelluksen kehitys on toteutettu Android Studio-kehitysympäristössä, joka on Android-käyttöjärjestelmän virallinen ohjelmointiympäristö. Android Studio tarjoaa kehittäjälle muun muassa lukuisia valmiita kirjastoja, jotka helpottavat sovelluksen kehittämistä sekä sisäänrakennetun emulaattorin, jolla ohjelmia pystyy suorittamaan. Työn toteuttamiseen on käytetty yleisiä Android-kehityksen tekniikoita, joita voidaan soveltaa myös muiden Android-sovellusten kehittämiseen. Työ on toteutettu Java-kielellä.

Työn lopputuloksena saatiin kehitettyä toimiva askelmittarisovellus. Sovelluksen kehitys on vielä kesken ja työtä tehdessä opittua teoriaa voi hyödyntää sovelluksen jatkokehitykseen sekä muiden Android-sovellusten kehittämiseen. Android Studio osoittautui erittäin hyväksi kehitysympäristöksi ja sen monipuolisia ominaisuuksia pystyttiin hyödyntämään hyvin työtä tehdessä.

Avainsanat Android Studio, askelmittari, Java, sovelluskehitys

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Vili Nurmi
Title	Android Pedometer Application
Year	2020
Language	Finnish
Pages	47
Name of Supervisor	Pirjo Prosi

The objective of this thesis was to create a pedometer application for devices that use the Android operating system. By developing applications for Android operating system the developer can reach up to three quarters of all mobile device users.

The application was developed in the Android Studio integrated development environment, which is the official development environment for Android applications. Android Studio offers the developer, among other things, multiple libraries to aid the application development and an in-built emulator which can be used to execute programs. The development of this application was carried out using common techniques of Android development which can be applied in the development of other applications.

The final result of this thesis is a functioning pedometer application. The development of this application is still in progress and the theory learned in the process of development can be utilized in further development of this application and other Android applications. Android Studio proved to be a very good development environment and its versatility could be utilized well in the development process.

Keywords Android Studio, pedometer, Java, application development

SISÄLLYS

KUVALUETTELO

TAULUKKOLUETTELO

1 JOHDANTO

Tämä opinnäytetyö tehdään Vaasan ammattikorkeakoulun tietotekniikan sulautettujen järjestelmien linjalle. Työ on omavalintainen ja sen tärkeimpänä valintaperusteena oli oman osaamisen kasvattaminen Android-sovelluskehityksessä sekä Java-ohjelmoinnissa.

Tässä opinnäytetyössä on tarkoituksena ohjelmoida Java-ohjelmointikielellä askelmittarisovellus Android-käyttöjärjestelmää tukeville mobiililaitteille. Työn toteuttamisen apuna on käytetty Android Studio ohjelmointiympäristöä, joka on Android-käyttöjärjestelmän virallinen ohjelmointiympäristö.

Tulevaisuutta silmällä pitäen, Android sovelluskehityksen taitamista voidaan pitää merkittävänä hyötynä tietotekniikan alalla, sillä arviolta 75 % kaikista mobiililaitteista käyttää Android-käyttöjärjestelmää. Lisäksi Java-ohjelmointikielen hallitsemisesta on suurta hyötyä työmarkkinoilla, sillä se on yksi suosituimmista ohjelmointikielistä. /1,2/

Opinnäytetyön alussa tutustutaan työssä käytettyihin tekniikoihin ja käydään läpi sovelluksen rakenne. Myöhemmässä vaiheessa puretaan kirjoitettua koodia osiin helpommin ymmärrettäväksi. Työn tavoitteena on antaa kokonaisvaltainen kuva Android-sovelluksen kehittämisestä ja selittää mistä osista tyypillinen Android-sovellus ja sovelluksen koodi koostuvat.

2 ANDROID KEHITYKSEN KÄSITTEITÄ

Android sovellusten kehitykseen liittyy useita tekniikoita ja käsitteitä, jotka ovat ominaisia vain Android-sovelluskehitykselle. Tässä luvussa käydään läpi yleisimpiä tässä työssä käytettyjä Android kehityksen käsitteitä sekä Java-kehitykseen liittyvä metodin korvaaminen.

2.1 Service - Palvelu

Palvelu on sovelluksen komponentti, joka suorittaa pitkäkestoisia operaatioita taustalla. Palvelut eivät ole näkyvissä käyttäjille, mutta ne voivat olla käynnissä silloinkin kun käyttäjä on sulkenut sovelluksen tai avannut toisen sovelluksen käytetyn laitteen näytölle. Tässä sovelluksessa käytetään palvelua, joka laskee taustalla käyttäjän askelia silloinkin kun sovellus ei ole auki laitteella. /15/

2.2 Aktiviteetit

Aktiviteetit ovat Android-sovellusten komponentteja jotka luovat ikkunan, jossa sovelluksen käyttöliittymä näytetään. Suurin osa sovelluksista koostuu useista ikkunoista ja täten myös useista aktiviteeteista, joiden välillä käyttäjä liikkuu. Aktiviteetit voivat käynnistää toisia aktiviteetteja sovelluksen sisällä tai usein jopa toisissa sovelluksissa. /16/

2.3 SQLite

SQLite on Android-käyttöjärjestelmään sisäänrakennettu tietokannan hallintajärjestelmä. Android Studio-kehitysympäristö sisältää lisäksi valmiin SQLiteOpenHelper-apuluokan, jota voidaan käyttää apuna tietokannan hallitsemiseen. Tässä työssä käytetään SQLite-tietokantaa, johon käyttäjän ottamat askeleet tallennetaan päiväkohtaisesti.

2.4 Metodin korvaaminen @Override-komennolla

Java-koodissa metodin korvaaminen toteutetaan @Override-komennolla. Metodin korvaaminen tarkoittaa toisesta luokasta perityn metodin korvaamista uudella metodilla, joka on samantyyppinen, samanniminen ja sillä on samat parametrit.

Android kehityksessä suuri osa luokista sisältää aktiviteettien elinkaarten määritykseen käytettäviä metodeja, joten niitä joudutaan usein syrjäyttämään (override), kun sama metodi periytyy toisesta luokasta. Esimerkkejä näistä metodeista ovat muun muassa onCreate() ja onStartCommand(). /3/

3 SOVELLUKSEN TOIMINTAMALLI

Tämän askelmittarisovelluksen toteutus koostuu kolmesta eri osasta ja samalla myös kolmesta Java-luokasta. Näiden kolmen luokan tärkeimpinä tehtävinä ovat taustalla käynnissä olevan palvelun luonti askelten laskemiseen, askelten tallentaminen tietokantaan sekä käyttöliittymän esittäminen käyttäjälle.

ServiceClass-luokan käyttötarkoituksena on käynnistää taustalla käynnissä oleva palvelu, joka vastaanottaa käytetyn laitteen STEP_DETECTOR-sensorilta tapahtumia, eli askeleita. Palvelu vastaanottaa sensorilta askelia taustalla silloinkin, kun sovellus ei ole auki laitteen näytöllä. Kun sensorilta vastaanotetaan tapahtuma, kutsutaan DBhelper-luokan addData-metodia.

DBhelper-luokassa suoritetaan askelten tallentamiseen tarvittavan SQLite-tietokannan luominen laitteen muistiin sekä askelten tallentaminen kyseiseen tietokantaan. ServiceClass-luokassa tehdyn addData-metodin kutsun jälkeen DBhelper-luokassa sijaitseva addData-metodi suorittaa askelten lisäämisen kuluvan päivän askelmäärään tai luo uuden rivin tietokantaan, mikäli kyseessä ovat päivän ensimmäiset askeleet. Samassa luokassa sijaitsee lisäksi getdataforCalendar-metodi, joka noutaa halutun päivän askelmäärän tietokannasta ja palauttaa kyseisen arvon käytettäväksi Calendaractivity-luokassa.

Calendaractivity-luokka on aktiviteettiluokka, joka vastaa sovelluksen käyttöliittymästä, kun sovellus avataan käytetyn laitteen näytölle. Käytännössä tämän sovelluksen käyttöliittymä koostuu kalenterinäkymästä, josta voidaan valita haluttu päivämäärä, ja tekstinäkymästä johon valitun päivän askelmäärä tulee näkyviin. Askelmäärä noudetaan tekemällä kutsu DBhelper-luokan getdataforCalendar-metodille, joka palauttaa halutun päivän askelmäärän esitettäväksi laitteen näytölle.

Sovelluksen ulkoasu on suunniteltu erillisessä activity_calendaractivity.xml-tiedostossa. Xml-tiedostossa suunniteltu ulkoasu saadaan käyttöön Calendaractivity-luokassa suorittamalla setContentView-komento, joka linkittää xml-tiedoston Java-tiedostoon.

4 SERVICECLASS-LUOKKA

ServiceClass-luokan käyttötarkoituksena on luoda taustalla käynnissä oleva palvelu, joka vastaanottaa step_detector-sensorilta tapahtumia eli askeleita ja kutsuu DBHelper-luokan addData-metodia, jolla askeleet lisätään käytettävän laitteen SQLite-tietokantaan. ServiceClass-luokka on Service-luokan aliluokka eli se perii Service-luokan tiedot, joita käytetään apuna palvelun luomiseen. Lisäksi ServiceClass-luokka toteuttaa (implements) SensorEventListener-luokan, joka sisältää onAccuracyChanged- sekä onSensorChanged-metodit.

4.1 STEP_DETECTOR-sensori

Android-käyttöjärjestelmän versiosta 4.4 (koodinimi ”KitKat”) eteenpäin on ollut mahdollista käyttää laitteen step_detector- ja step_counter-sensoreita. Step-sensorien käyttö perustuu sisäisesti kiihtyvyysanturiin, mutta Android kohtelee niitä loogisesti erillisinä sensoreina. Sensoria rekisteröidessä tulee koodissa tarkistaa onko käytetyllä laitteella valmius käyttää step-sensoreita. /4/

Step_detector-sensorin toimintaperiaate on yksinkertainen. Kun sensori havaitsee askeleen, se laukaisee tapahtuman (event) ja palauttaa arvon 1 jokaista otettua askelta kohden. Kun uusia askeleita ei ole arvo on 0. Otettujen askeleiden kokonaismäärän laskeminen joudutaan toteuttamaan itse sovelluksen koodissa. Tämä johtuu siitä, että sensori ei kykene palauttamaan kasvavaa lukua, vaan ainoa sensorin palauttama arvo on 1. Askelten kokonaismäärän laskeminen ja tallentaminen on toteutettu DBHelper-luokassa. /4/

4.2 Sensorin käytön alustus onCreate-metodin sisällä

Kuvassa 1 näkyy ServiceClass- luokan OnCreate-metodin sisällä toteutettavat asetukset, jotka on tarkoitus toteuttaa vain kerran silloin kun palvelu luodaan. Askelmittari-sovelluksessa tarkoituksena on, että sensorin tapahtumia (askeleita) kuunteleva palvelu on käynnissä jatkuvasti niin pitkään, kunnes sovelluksen asennus poistetaan. Tästä syystä myös sensorin tapahtumia kuunteleva ”listener” eli kuuntelija määritetään onCreate-metodin sisällä.

```
@Override
public void onCreate() {

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    if (sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR) != null) {

        stepdetector =
sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);
        sensorManager.registerListener(this, stepdetector,
SensorManager.SENSOR_DELAY_NORMAL);
    }
```

Kuva 1. Sensorin käytön alustus

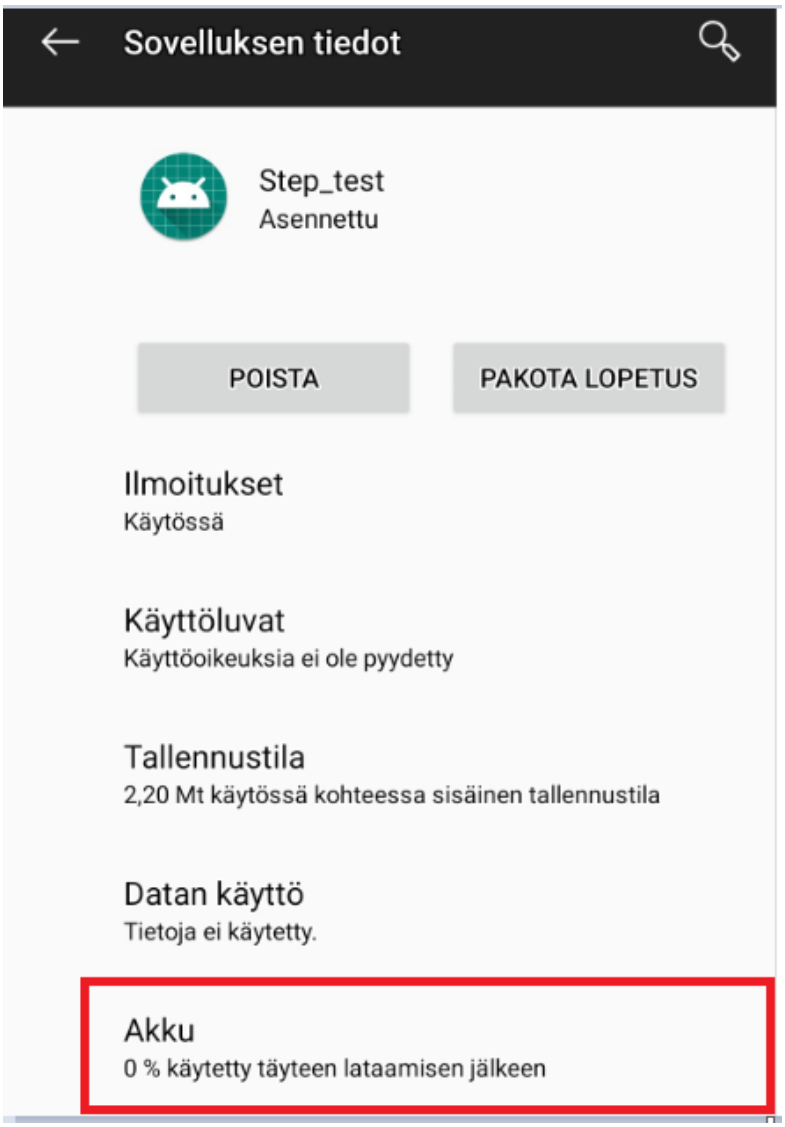
SensorManager-luokkaa apuna käyttämällä sovellus pääsee käsiksi laitteen sensoreihin. Kuvassa 1 if-ehtolauseen sisällä tarkistetaan onko käytössä olevalla laitteella TYPE_STEP_DETECTOR nimistä sensoria. Mikäli tarkistuksen tulos on erisuuri kuin null, eli sensori löytyy, toteutetaan ehtolauseen sisällä oleva koodi.

SensorManager.getDefaultSensor-komennolla valitaan sensori, jonka tapahtumia halutaan alkaa ”kuuntelemaan” eli vastaanottamaan. Tämän jälkeen registerListener-metodilla rekisteröidään kuuntelija, joka seuraa sensorilta saapuvia tapahtumia. Tässä tapauksessa registerListener-metodille annetaan parametriksi stepdetector, joka on getDefaultSensor-metodin sisällä määritetty STEP_DETECTOR-sensoriksi.

4.3 Sensorin viive ja virrankulutus

SENSOR_DELAY-parametri määrittää sensorin viiveen, eli kuinka nopeasti sensori lähettää tietoja sovellukselle kutsuttaessa onSensorChanged-metodia. Oletusarvo SENDOR_DELAY_NORMAL on sopiva suurimmassa osassa tapauksista ja tarkoittaa 200 000 mikrosekunnin (0,2 sekuntia) viivettä. Muita vaihtoehtoisia määrityksiä ovat esimerkiksi SENSOR_DELAY_FASTEST (0 mikrosekuntia) ja SENSOR_DELAY_UI (20000 mikrosekuntia). Määritetty viive (delay) on kuitenkin lopulta vain ehdotus, sillä Android-käyttöjärjestelmä ja muut sovellukset voivat vaikuttaa viiveeseen. Tästä syystä yleisenä tapana on määrittää mahdollisimman suuri

viive, koska usein järjestelmä käyttää tästä huolimatta pienempää viivettä kuin määritetty. Suuremman viiveen määritys aiheuttaa vähemmän lastia käytössä olevan laitteen prosessorille ja täten kuluttaa vähemmän virtaa. /5/



Kuva 2. Sovelluksen akun käyttö

Step-sensorit on suunniteltu toimimaan erittäin pienellä virrankulutuksella. Sovellus on käytössä omalla OnePlus 6 puhelimellani ja kuvassa 2 näkyvistä sovelluksen tiedoista pystyin tarkistamaan, että vaikka edellisestä täyteen lataamisesta kuvankaappauksen ottamisen aikaan oli kulunut 11 tuntia ja 2 minuuttia, oli sovelluksen akun käyttö silti alle 1 %. Tämä siitäkin huolimatta, että sovelluksen taustapalvelu,

joka vastaanottaa sensorilta askeleita ja tallentaa niitä tietokantaan, on ollut käynnissä jatkuvasti.

4.4 Taustapalvelun käynnistäminen etualalle

Android-ohjemointirajapinnan (API) versiosta 26 eteenpäin taustapalveluiden jatkuvaa käyttöä on rajoitettu huomattavasti käyttökokemuksen parantamiseksi. Ohjelmointirajanpinnan versio 26 tarkoittaa myös Android-käyttöjärjestelmän versiota 8.0, joka on nimeltään Oreo. Uusien rajoitusten kiertämiseksi sovellusten tulee jatkossa käyttää taustalla toimivien töiden vuorottajaa (JobScheduler) tai käynnistää sovelluksen taustapalvelu etualalla (Foreground). /7/

Alun perin sovellusta luodessa tein taustapalvelun (background service), mutta huomasin, että sovelluksen käynnistämisen jälkeen se laski askelia vain noin 10 sekuntia, jonka jälkeen askelmäärä ei enää kasvanut. Löydettyäni artikkelin uusista ohjelmointirajapinnan rajoituksista ymmärsin, että ongelma johtuu uusista ohjelmointirajanpinnan rajoituksista, jotka estävät taustapalveluita pysymästä päällä määräämättömiä aikoja. Lisäsin tämän jälkeen ohjelman koodiin funktion, joka käynnistää taustapalvelun etualalle ja pitää palvelun käynnissä. /7/

```
private void StartForeground()
{
    String NOTIFICATION_CHANNEL_ID = "com.example.step_test.ServiceClass";
    String channelName = "My Background Service";
    NotificationChannel chan = new NotificationChannel(NOTIFICATION_CHANNEL_ID,
channelName, NotificationManager.IMPORTANCE_LOW);
    chan.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
    NotificationManager manager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
    assert manager != null;
    manager.createNotificationChannel(chan);

    NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this, NOTIFICATION_CHANNEL_ID);
    Notification notification = notificationBuilder.setOngoing(true)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentTitle("App is running in background")
            .setPriority(NotificationManager.IMPORTANCE_MIN)
            .setCategory(Notification.CATEGORY_SERVICE)
            .build();
    startForeground(2, notification);


}
```

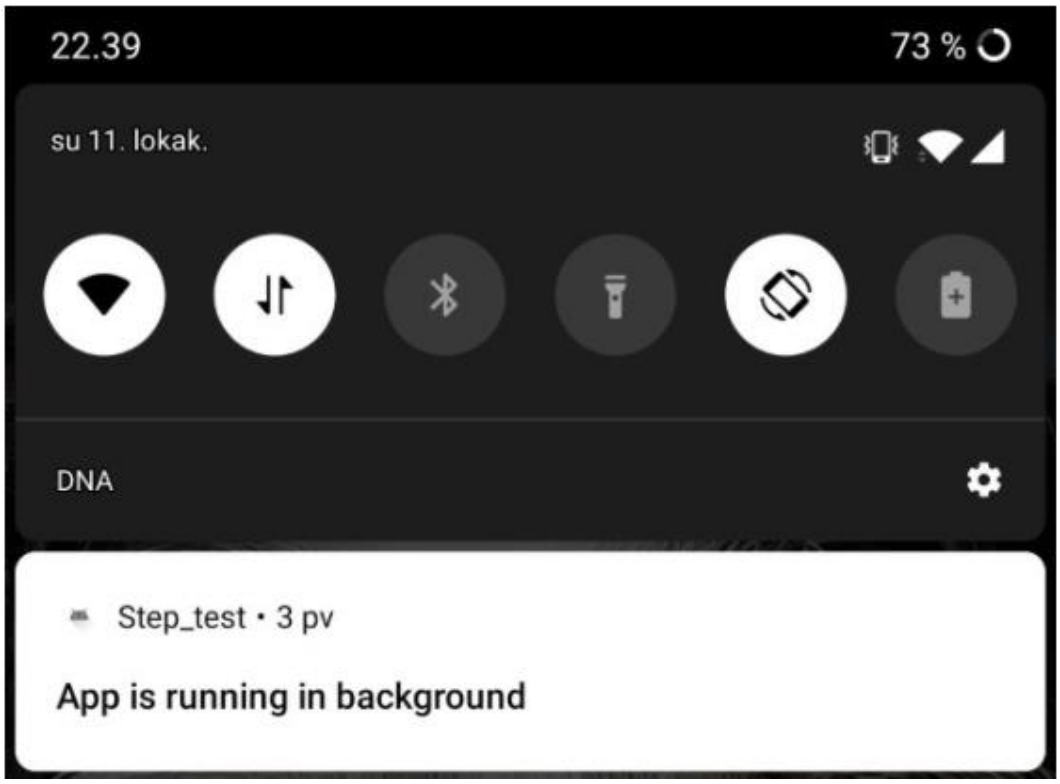
Kuva 3. Palvelun käynnistäminen etualalle

Kun palvelu halutaan käynnistää etualalla, sille tulee luoda ”Notification channel” eli ilmoituskanava. Kuvassa 3 näkyy etualalla toimivan palvelun ja ilmoituskanavan luontiin tarvittavaa koodia. Ilmoituskanavan avulla käytetyn laitteen näytölle saadaan ilmoitus siitä, että haluttu palvelu on käynnissä laitteella vaikka sovellus, joka palvelua käyttää, ei välttämättä ole avoinna.

Ilmoituskanavaa luotaessa Notificationchannel-objektille annetaan yksilöllinen ID, käyttäjälle näkyvä kanavan nimi ja ilmoituksen tärkeystaso. Kanavan nimen käyttäjä voi nähdä sovelluksen ilmoitusasetuksista.

Kuva 4. Android-tilarivi (notification bar)

Kuva 4 on kuvaruutukaappaus omalta OnePlus 6-puhelimeltani, jossa näkyy punaisella ympyröitynä askelmittarisovelluksen ikoni. Tämä ikoni kertoo, että sovellus tekee työtä, eli laskee askelia taustalla. Kyseisen ikonin ulkonäkö määritetään ilmoituskavanaa luotaessa komennolla ”.setSmallIcon(R.drawable.ic_launcher_foreground”. Komennon loppuosa ”ic_launcher_foreground” on käytetyn kuvatiedoston nimi ja kyseinen tiedosto löytyy valmiiksi Android Studion kirjastosta.



Kuva 5. Askelmittarisovelluksen ilmoitus

Kuvassa 5 näkyy askelmittarisovelluksen lähettämä ilmoitus. Ilmoituksen teksti määritetään ilmoituskanavaa luotaessa komennolla ”.setContentTitle”. Tällä kertaa sovelluksen ilmoitukseksi on asetettu geneerinen teksti joka kertoo, että palvelu on käynnissä taustalla. Ilmoituksilla pystyy tekemään paljon muutakin kuin esittämään

pelkkää tekstiä ja tulevaisuudessa sovellukseen voisi lisätä ilmoituksen, joka näyttää otetut askeleet reaaliaikaisesti ilman, että koko sovellusta tarvitsee avata.

4.5 OnStartCommand ja onSensorChanged metodit

Palvelua käynnistettäessä käytetään apuna Android-sovelluksen elinkaaren määrittelyyn käytettyjä metodeja. Palvelun käynnistys tapahtuu kuvassa 6 näkyvällä OnStartCommand-metodilla ja se pysähtyy vasta, kun Context.stopService()- tai stopSelf()-metodeja kutsutaan. Metodi palauttaa arvon START_STICKY, joka on tarpeellinen vasta tilanteessa, jossa käytetyn laitteen muisti lakkaa ja palvelu pysähtyy. Tällöin START_STICKY käynnistää palvelun uudelleen. /8/

Kuvassa 6 näkyy myös OnSensorChanged-metodin alustus ja metodia kutsutaan, kun onCreate-metodin sisällä määritetyltä STEP_DETECTOR-tyyppiseltä sensorilta tulee ”SensorEvent” eli tapahtuma. OnSensorChanged-metodi puolestaan kutsuu DBHelper-luokassa sijaitsevaa addData-metodia, joka tallentaa askeleet SQLite-tietokantaan.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId){

    return Service.START_STICKY;

}


@Override
public void onSensorChanged(SensorEvent e) {

    dBhelper.addData(); //kun sensori havaitsee muutoksen kutsutaan DBhelper
luokan addData() metodia

}
```

Kuva 6. OnStartCommand- ja onSensorChanged-metodien alustus

5 DBHELPER-LUOKKA

Tässä askelmittarisovelluksessa DBhelper-luokassa toteutetaan tiedon tallentaminen SQLite-tietokantaan sekä tiedon noutaminen tietokannasta Calendaractivity-luokassa esitettävään muotoon. DBhelper-luokka on SQLiteOpenHelper-luokan aliluokka. /6/

5.1 SQLiteOpenHelper Ja DBhelper luokka

SQLiteOpenHelper-luokka on DBhelper-luokan yliluokka (super class), jota käytetään apuna SQLite-tietokannan luomiseen sekä hallinnoimiseen. Tämä luokka huolehtii tietokannan avaamisesta sekä tietokannan luomisesta ja päivittämisestä tarvittaessa. /3/

```
public class DBhelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "stepdb";
    public static final String TABLE = "steptable";
    public static final String ID = "ID";
    public static final String DATE = "Date";
    public static final String TOTAL_STEPS = "total_steps";
    public static final String DATABASE_CREATE = "CREATE TABLE " + TABLE + "(" +
ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + DATE + " TEXT," + TOTAL_STEPS + "
INTEGER" + ")";
```

Kuva 7. DBhelper-luokan muuttujien alustus

DBhelper-luokan alussa määritellään tarvittavat muuttujat tietokannan luomiseen. Kuvassa 7 näkyy tässä luokassa käytettyjen muuttujien alustus. Tietokannalle annetaan nimi DATABASE_NAME ja kaikki tietokannan taulut nimetään. Tässä tapauksessa tietokannassa on vain yksi taulu. Seuraavaksi nimetään taulun sarakkeet. Nimeämisille ei ole määritelty erityisiä sääntöjä, joten tietokannan, taulut ja sarakkeet voi nimetä haluamallaan tavalla. Taulun ensimmäiseen sarakkeeseen ID, tulee automaattisesti kasvava yksilöllinen luku jokaiselle riville eli ID:tä ei tarvitse erikseen päivittää tietokantaan, vaan luku kasvaa itsestään yhdellä aina kun tietokantaan lisätään uusi rivi. Taulun toiseen sarakkeeseen DATE tallennetaan jokaisen päivän päivämäärät, kun askelia on otettu. Viimeiseen sarakkeeseen TOTAL_STEPS tallennetaan päivämäärän perusteella päivän aikana otetut askeleet. Päivämäärien ja

askeleiden tallennus suoritetaan addData-metodissa, joka on selitetty seuraavassa luvussa.

Lopuksi alustetaan vielä SQL-lauseella DATABASE_CREATE-muuttuja, jota käytetään tietokannan luomiseen. SQL-lauseen alussa annetaan CREATE TABLE -komento, joka luo uuden taulun stepdb-tietokantaan. Taulun määrityksiksi annetaan taulun nimi sekä taulussa käytettävät sarakkeet. /6/

ID-sarakkeelle annetaan tässä vaiheessa määritys INTEGER PRIMARY KEY AUTOINCREMENT, joka vastaa siitä, että ID-sarakkeen arvo kasvaa yhdellä aina kun tauluun lisätään uusi rivi. Käytännössä jokaisessa SQL-taulussa tulee olla yksi sarake, joka on primary key. Tämä pitää huolen siitä, että taulun jokaisella rivillä on ainakin yksi sarake, joka varmasti erottaa rivit toisistaan. /6/

DATE-sarakkeelle annetaan määritys TEXT, joka vastaa string-muuttujan tyyppiä, koska päivämäärät tallennetaan tauluun string-muodossa. Lopuksi TOTAL_STEPS-sarakkeelle määritetään tyypiksi INTEGER, joka vastaa integer-tyypin muuttujaa. Päivän askeleet tallennetaan tähän sarakkeeseen numeromuodossa.

```
public DBhelper(Context context) {
    super(context, DATABASE_NAME, null, 1);
}


@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(DATABASE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

}
```

Kuva 8. DBhelper-luokan konstruktori ja onCreate-metodi

Kuvassa 8 näkyy DBhelper-luokan konstruktorin sekä onCreate-metodin alustus. DBhelper-luokan konstruktorilla on neljä parametria: Context, name, factory ja ver-

sion. Tässä tapauksessa Context-muuttuja tarjoaa pääsyn tietokantaan ja DATABASE_NAME-muuttuja on tietokannan nimi. Factory-parametria käytetään cursor-objektien luomiseen, mutta se voidaan jättää oletuksena arvoon null. Lopuksi vielä version parametrin arvoksi asetetaan 1, joka tarkoittaa tietokannan versiota 1. Tämä on tietokannan versio, joka tulee sovelluksen asennuksen mukana. Tätä tietoa voidaan käyttää tulevaisuudessa esimerkiksi jos sovellusta päivitetään ja tietokantaan tehdään muutoksia, esimerkiksi lisätään uusia sarakkeita. Tällöin onUpgrade-metodissa voidaan tarkistaa versionumeron perusteella, mikä versio tietokannasta käyttäjällä on asennettuna ja päivittää tietokanta uudempaan versioon. Myös tietokannan päivittäminen takaisin vanhempaan versioon onnistuu käyttämällä onDowngrade-metodia. /6/

OnCreate-metodia kutsutaan vain silloin kun aikaisempaa tietokantaa ei ole olemassa ja se toteuttaa tietokannan luonnin käyttäen aikaisemmin määritettyä DATABASE_CREATE-muuttujaa. Tietokannan luonti tapahtuu kuitenkin todellisuudessa vasta siinä vaiheessa kun getReadableDatabase- tai getWritableDatabase-metodia kutsutaan ensimmäisen kerran.

OnUpgrade-metodia käyteään tietokannan versioiden päivittämiseen. Käytännössä tämä tulee tarpeeseen vain, kun sovelluksesta julkaistaan uusi versio, jossa esimerkiksi tietokantaan on lisätty tauluja tai kolumneja. Yleisesti onUpgrade-metodin sisällä käytetään ”DROP TABLE” SQL-komentoa, joka poistaisi vanhan tietokannan taulun ja korvaisi sen uudella. Tämän sovelluksen käyttötarkoituksiin onUpgrade-metodia ei kuitenkaan tarvita, koska vanhan taulun poistaminen poistaisi myös kaiken tiedon aikaisemmista askelista ja päivämääristä. Tästä syystä metodi on jätetty tyhjäksi, mutta SQLiteOpenHelper-luokan käyttö vaatii, että metodi on kuitenkin olemassa. /6/

5.2 addData()-metodi

AddData-metodin tehtävänä on lisätä SQLite-tietokantaan askelia sekä päivämääriä. Metodin ensimmäisessä osassa alustetaan tarvittavat muuttujat sekä tarkistetaan onko tietokantaan tallennettu jo kuluvana päivänä askelia. Tätä tietoa tarvitaan

myöhemmin määrittämään onko tarpeena lisätä uusia askelia kuluvan päivän askeliin vai luoda uusi rivi uudelle päivälle tietokantaan.

```
public boolean addData() { //metodi jolla lisätään askelia tietokantaan

    int stepsToday = 0;
    boolean dateCheck = false;
    boolean result = false;
    Calendar calendar = Calendar.getInstance();
    String today = String.valueOf(calendar.get(Calendar.DAY_OF_MONTH)) + "." +
String.valueOf(calendar.get(Calendar.MONTH) + 1) + "." +
String.valueOf(calendar.get(Calendar.YEAR));
```

Kuva 9. AddData-metodin muuttujien alustus

Kuvassa 9 näkyy addData-metodin käyttöön alustetut muuttujat. StepsToday-, dateCheck- ja result-muuttujia käytetään apuna kuluvan päivän askelien tarkistamiseen. Lisäksi määritellään calendar-muuttuja, jota tarvitaan Calendar-luokan käyttämiseen sekä String today-muuttuja, joka kertoo kuluvan päivän, kuukauden ja vuoden.

```
try {

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT " + TOTAL_STEPS + " FROM " + TABLE + "
WHERE " + DATE + " = '" + today + "'", null);

    if (cursor.moveToFirst()) {
        stepsToday = cursor.getInt(cursor.getColumnIndex(TOTAL_STEPS));
        dateCheck = true;
    }
    db.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

Kuva 10. Tietokannan avaaminen luettavaan muotoon ja kuluvan päivän askelmäärän tarkistus

Muuttujien alustuksen jälkeen kuvassa 10 näkyvää getReadableDatabase-metodia kutsumalla avataan stepdb-tietokanta luettavaan muotoon. Tämän jälkeen suoritetaan rawQuery-kysely stepdb-tietokantaan, jolla saadaan selville kuluvan päivän

askelien määrä. Jos kuluvana päivänä on askelia, boolean dateCheck-muuttujan arvoksi asetetaan true ja tätä tietoa käytetään seuraavassa vaiheessa määrittämään lisätäänkö uudet askeleet kuluvan päivän askeliin vai luodaanko tietokantaan uusi rivi uudelle päivälle. Lopuksi suljetaan tietokanta db.close-komennolla.

```
try {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();

    contentValues.put(DATE, today);
    if (dateCheck) { //tarkoittaa samaa kuin if dateCheck = true
        contentValues.put(TOTAL_STEPS, ++stepsToday);

        int update = db.update(TABLE, contentValues, DATE + "= '" + today + "'",
null);

        if (update == 1) {
            result = true;
        }
        db.close();
```

Kuva 11. Askelten lisääminen tietokantaan

Kun on saatu selville onko kuluvana päivänä askelia, voidaan siirtyä lisäämään tietoja tietokantaan. Kuvassa 11 näkyy kuinka getWritableDatabase-metodia kutsumalla avataan tietokanta kirjoitettavaan muotoon. Tietoja tallennetaan käyttämällä apuna ContentValues-luokkaa. ContentValues.put-komento ottaa 2 parametria, kolumnin, johon tietoa halutaan tallentaa, ja itse arvon, joka halutaan tallentaa. Tässä tapauksessa stepdb-tietokannan DATE-kolumniin halutaan tallentaa today-muuttujan arvo, joka saatiin muuttujien alustusvaiheessa hakemalla kalenterista kuluvan päivän päivämäärä. Jos dateCheck-muuttujan arvo on true se tarkoittaa, että kuluvana päivänä on jo askelia, joten kuluvan päivän askeliin lisätään uudet askeleet sen sijaan, että luotaisiin uusi rivi. Uuden rivin lisääminen käydään läpi seuraavassa vaiheessa. Jälleen contentValues.put-komentoa käyttämällä tietokantaan lisätään tällä kertaa TOTAL_STEPS-kolumniin stepsToday-muuttujan arvo.

Muutokset viedään tietokantaan saakka kuvassa 11 näkyvällä db.update-komennolla, joka ottaa argumenteiksi taulun nimen, muutettavat arvot ja where-lauseen,

jonka perusteella tiedetään mitä rivejä ja sarakkeita tietokannassa halutaan päivittää. Lopuksi tehdään vielä tarkistus, että tietokannan päivitys onnistui. Update-komento palauttaa aina arvon siitä kuinka moneen riviin on tehty muutos. Tässä tapauksessa halutaan päivittää aina yhtä riviä, joten tarkistuksen arvoksi kelpaa vain luku 1. Tällöin result boolean-muuttujan arvoksi asetetaan true eli tosi.

```
} else { //jos kuluvalla päivällä ei ole vielä askelia
    contentValues.put(TOTAL_STEPS, 1);
    long insert = db.insert(TABLE, null, contentValues);


    if (insert != -1) {
        result = true;
    }
    db.close();
}
} catch (Exception e) {
        e.printStackTrace();
    }
    return result; //palautetaan arvo true jos riville/rivin lisääminen onnistui
}
```

Kuva 12. Askelten lisääminen tietokantaan, jos kuluvana päivänä ei ole askelia

Metodin lopuksi käytetään if-else-ehtolauseen else-osaa tapauksia varten, joissa kuluvana päivänä ei ole vielä askelia tietokannassa. Tämä ehto toteutuu kun aamulla ottaa uuden päivän ensimmäisen askeleen. ContentValues-muuttuja ottaa 2 paremetria, taulun jota päivitetään, ja itse datan. Tässä tapauksessa päivitetään TOTAL_STEPS-tauluun 1 askel. Tämän jälkeen aletaan jälleen toteuttamaan if-else ehdon ensimmäistä osaa, jossa kuluvan päivän askeliin lisätään lisää askelia. SQL-komennolla db.insert - tietokannan TABLE nimiseen tauluun lisätään contentValues muuttujan arvot, eli tässä tapauksessa 1 askel. Insert-komento palauttaa aina uuden lisätyn rivin ID:n, joka on automaattisesti kasvava luku. Jokaisella tietokannan rivillä on siis yksilöllinen ID. Tästä syystä if-lauseessa tarkistetaan, että uuden rivin ID ei ole -1. Kaikki tätä suuremmat ID:t nollasta eteenpäin on sallittuja ja uuden rivin lisäys on onnistunut. Lopuksi palautetaan vielä boolean result-muuttujan arvo true jos rivin lisäys onnistui.

5.3 GetDataforCalendar-metodi

GetDataforCalendar-metodia käytetään tietojen noutamiseen tietokannasta. Metodi palauttaa steps-muuttujan, jota käytetään myöhemmin CalendarActivity-luokassa päivän askelten esittämiseen näytöllä.

```
public int getdataforCalendar() {
    int steps = 0;
    SQLiteDatabase db = this.getReadableDatabase();
    Calendaractivity calendaractivity = new Calendaractivity();
    String calendartoday = calendaractivity.date;
    Cursor cursor = db.rawQuery("SELECT " + TOTAL_STEPS + " FROM " + TABLE + "
WHERE " + DATE + " = '" + calendartoday + "'", null);


    while (cursor.moveToNext()) {
steps = cursor.getInt(cursor.getColumnIndex(TOTAL_STEPS));
    }

    return steps; //palautetaan päivän askeleet
}
```

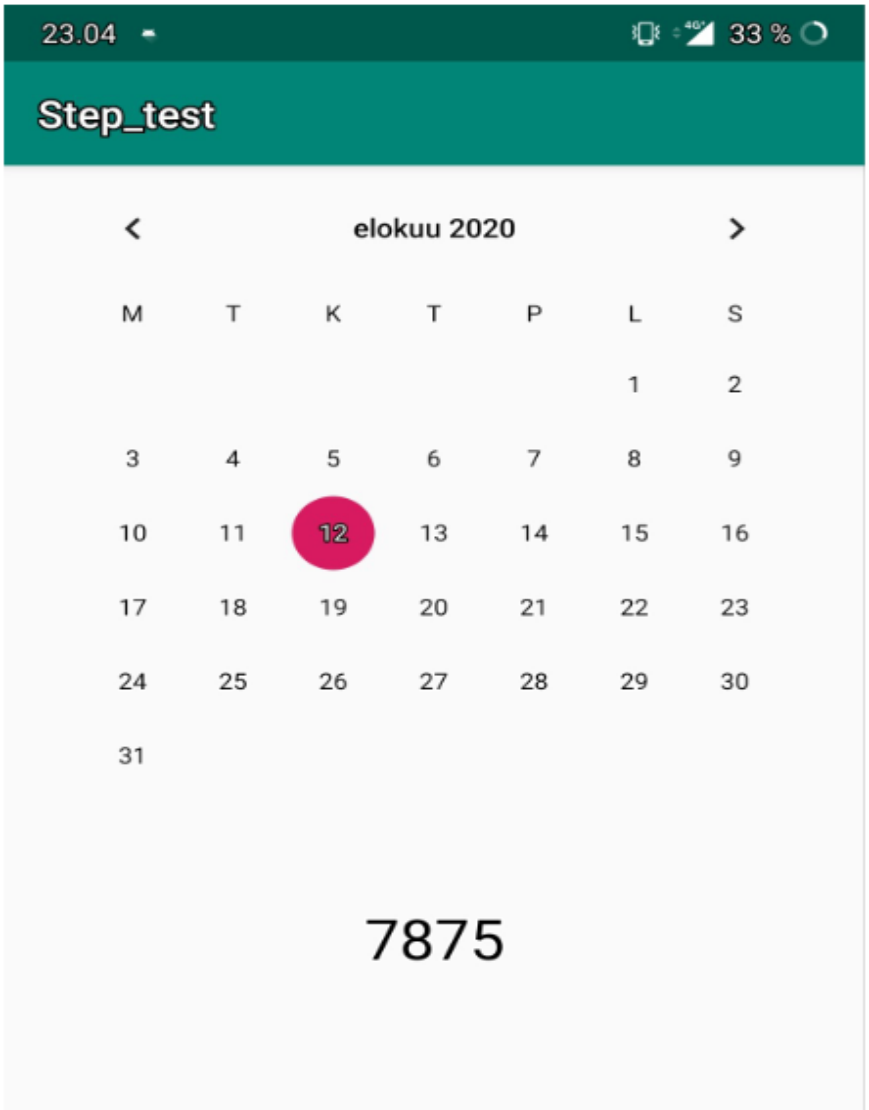
Kuva 13. GetDataforCalendar-metodin alustus

Kuvassa 13 esitetään getDataForCalendar-metodin muuttujien alustus sekä rawQuery-kysely tietokantaan. Aluksi alustetaan integer tyyppinen steps-muuttuja, jota käytetään apumuuttujana haettaessa tietoja tietokannasta. Tämän jälkeen tietokanta avataan jälleen luettavaan muotoon käyttämällä getReadableDatabase-komentoa. Lisäksi alustetaan String-muotoinen calendartoday-muuttuja ja alustetaan muuttujan arvoksi calendaractivity.date eli päivä, jonka käyttäjä on valinnut kalenterista painamalla näytöllä haluttua päivää.

Seuraavaksi suoritetaan rawQuery-kysely stepdb-tietokantaan, jolla haetaan tietokannan TOTAL_STEPS-kolumnista calendartoday-muuttujan perusteella oikean päivän askeleet. Lopuksi palautetaan vielä integer tyyppisen steps-muuttujan arvo, joka edustaa numeromuodossa kyseisen päivän askelia. Tätä arvoa käytetään CalendarActivity-luokassa esittämään valitun päivän askeleita.

6 CALENDARACTIVITY-LUOKKA

CalendarActivity-luokka on aktiviteettiluokka, jossa luodaan ikkuna, jonka sisällä sovelluksen käyttöliittymä esitetään. Kuvassa 14 näkyy sovelluksen käyttöliittymä, jossa on yksinkertainen kalenterinäkymä, tästä käyttäjä voi valita päivämäärän ja nähdä kyseisenä päivänä otettujen askelien määrän. Valitun päivän askelien määrä tulee näkyviin kalenterinäkymän alapuolelle tekstikenttään. Lisäksi käyttäjä voi siirtyä eri kuukausien välillä painamalla kalenterinäkymän vasemmassa ja oikeassa yläkulmassa sijaitsevia nuolinäppäimiä. /9/



Kuva 14. Sovelluksen käyttöliittymä

6.1 Calendaractivity-muuttujat ja onCreate-metodi

Kuvassa 15 näkyy Calendaractivity-luokan alussa määritellyt tarvittavat muuttujat askelten näyttämiseen. CalendarView- ja textView-muuttujat ovat View-tyyppisiä muuttujia ja niitä käytetään apuna kalenterin sekä askeleiden esittämiseen laitteen näytöllä. Molemmat muuttujat on määritetty käyttöön activity_calendaractivity.xml–tiedostossa ja nämä muuttujat alustamalla niitä voidaan käyttää Java-koodissa. Steps-muuttujaa käytetään myös apuna valitun päivän askeleiden esittämiseen ja date-muuttujaa tarvitaan päivämäärän valitsemiseen. /14/

```
public class Calendaractivity extends AppCompatActivity {

    CalendarView calendarView;
    TextView textView;
    private int steps;
    public static String date;
```

Kuva 15. Calendaractivity-luokan muuttujien määrittely

Kuvassa 16 näkyy CalendarActivity-luokan onCreate-metodi, joka ”korvataan” (override) aluksi samaan tapaan kuin muissakin luokissa. OnCreate-metodia kutsutaan kun aktiviteetti käynnistetään. Lisäksi onCreate-metodia kutsutaan silloin kun käytetyn laitteen ruutua käännetään, mikä on metodin ominaisuus. Käytännössä metodia siis kutsutaan silloin, kun käyttäjä avaa sovelluksen, jolloin myös aktiviteetti käynnistyy.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_calendaractivity);
    calendarView = findViewById(R.id.calendarView);
    textView = findViewById(R.id.textView);
```

Kuva 16. Calendaractivity-luokan onCreate-metodi

SetContentView-komennolla activity_calendaractivity.xml-tiedosto linkitetään CalendarActivity Java-tiedostoon. Seuraavaksi findViewById-komennolla haetaan activity_calendaractivity.xml-tiedostosta ID:n perusteella kalenterinäkymä sekä tekstinäkymä. ID on käytännössä xml-tiedostossa näkymälle annettu nimi tai tunniste, jonka perusteella näkymä voidaan hakea Java-tiedostosta. Esimerkiksi kalenterinäkymän ID on alustettu xml-tiedostossa komennolla android:id=”@+id/calendarView”./14/

```
calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
    @Override
    public void onSelectedDayChange(CalendarView view, int year, int month, int
dayOfMonth)
    {
        date = dayOfMonth + "." + (month+1) + "." + year;
        getdata();
    }
});
```

Kuva 17. OnSelectedDayChange-metodin alustus

OnDateChangeListener-metodin käyttötarkoituksena on ”kuunnella” käyttäjän antamia käskyjä eli tässä tapauksessa sitä, kun käyttäjä valitsee päivämäärän kalenterinäkymästä. Kun käyttäjä valitsee haluamansa päivän, kutsutaan onSelectedDayChange-metodia, jonka argumentteina ovat valittu vuosi, kuukausi ja päivä. Luokan alussa määritettyä String-muotoista date-muuttujaa käytetään apuna välittämään valittu päivä DBHelper-luokan getdataforCalendar-metodille, joka palauttaa valitun päivän askeleet. Lopuksi kutsutaan vielä getdata-metodia, joka hakee valitun päivän askeleet textView-kenttään ruudulle näytettäväksi. GetData-metodin toiminta on selitetty tarkemmin seuraavassa luvussa.

Aktiviteettien elinkaarimallin mukaan onSelectedDayChange-metodi olisi voitu laittaa myös onStart-metodin sisään, mutta tämä olisi vaatinut myös muiden metodien, kuten onStop lisäämistä. Tämän sovelluksen toiminnan kannalta kuitenkin riitti pelkän onCreate-metodin käyttö ja se säästi hieman vaivaa, koska muita metodeja ei tarvinnut lisätä.

6.2 Intent

Intent (suom. aikomus) on objekti, jota voidaan käyttää apuna viestinnässä sovelluksen eri komponenttien välillä. Käytännössä intentien avulla sovelluksen toinen komponentti voi pyytää toiselta komponentilta jotakin toimintoa. Tässä tapauksessa intentiä käytetään viestinnässä sovelluksen aktiviteetti- ja palvelukomponentin välillä. Aktiviteetti pyytää palvelua käynnistämään taustapalvelun, joka vastaanottaa askelia ja tallentaa niitä tietokantaan. Käytännössä tämä pyyntö tapahtuu, kun käyttäjä avaa sovelluksen ensimmäisen kerran, koska intent sijaitsee aktiviteettiluokan onCreate-metodin sisällä. /10,11/

Intentejä on kahta eri tyyppiä. Explicit intentejä käytetään sovelluksen sisäiseen viestintään eri komponenttien välillä. Implicit intentillä sovellus voi viestiä samalla laitteella sijaitsevien muiden sovellusten kanssa. /10/

```
Intent mStepsIntent = new Intent(getApplicationContext(), ServiceClass.class);
ContextCompat.startForegroundService(this, mStepsIntent);
```

Kuva 18. Intent määrittely

Kuvassa 18 on määritetty intent käyttöön Calendaracitivity-luokassa. Kuvan 18 ensimmäisellä rivillä intentin parametreiksi annetaan konteksti getApplicationContext ja luokaksi ServiceClass.class. Konteksti on abstrakti luokka, joka tarjoaa pääsyn applikaation resursseihin sekä luokkiin ja mahdollistaa muun muassa intentien vastaanottamisen. ServiceClass on luokka jossa taustapalvelu, joka halutaan käynnistää, on määritetty. /10/

Kuvan 18 toisella rivillä käynnistetään StartForegroundService-komennolla ServiceClass-luokassa sijaitseva taustapalvelu, joka vastaanottaa sensorilta askeleita ja tallentaa ne SQLite-tietokantaan. Komennon parametreiksi tulee konteksti this, joka viittaa aktiviteettiin tai luokkaan, jossa komento sijaitsee ja toinen parametri on intent, joka luotiin ylemmällä rivillä.

6.3 GetData-metodi

GetData-metodin toimintaperiaate on hyvin yksinkertainen ja sen käyttötarkoituksena on näyttää käyttäjälle valitun päivän askeleet ruudulla textView-kentässä. GetData-metodia kutsutaan sen jälkeen, kun käyttäjä on painanut näytöllä haluamaansa päivämäärää kalenterinäkymästä.

```
public void getdata()
{
    DBhelper mDBHelper = new DBhelper(this);
    steps = mDBHelper.getdataforCalendar();
    textView.setText(String.valueOf(steps));
}
```

Kuva 19. GetData-metodin alustus

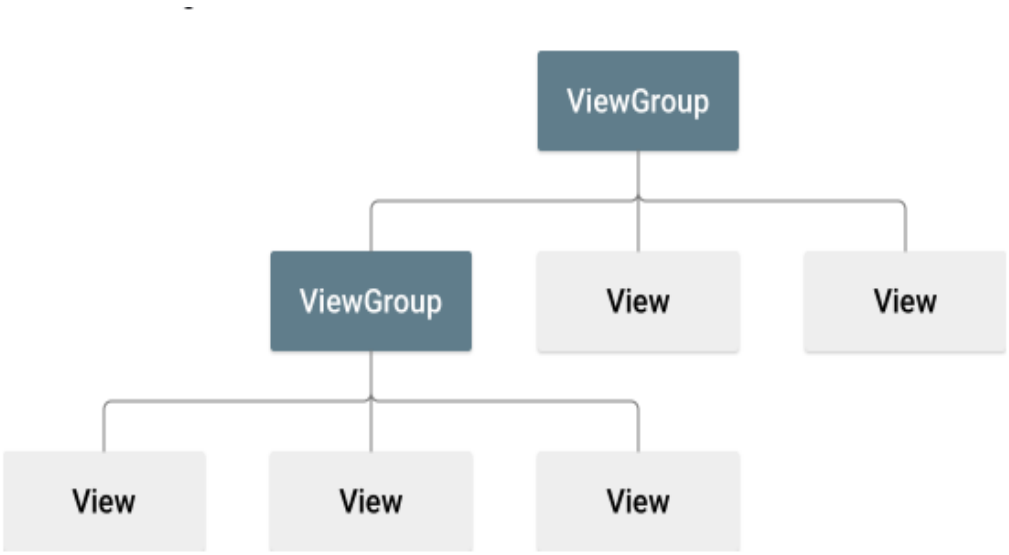
GetData-metodin sisällä luodaan uusi DBhelper-luokan objekti. DBhelper-luokkaa tarvitaan, koska se sisältää tarvittavat metodit tiedon noutamiseen SQLite-tietokannasta, johon askelmäärät ja päivämäärät on tallennettu. Seuraavaksi integer tyyppiselle steps-muuttujalle määritetään arvoksi DBHelper-luokan getdataforCalendar-metodin palauttama arvo. GetdataforCalendar-metodin toiminta on selitetty kappaleessa 5.3 ja kyseinen metodi palauttaa integer tyyppisen steps-muuttujan, joka vastaa valittuna päivänä otettuja askeleita.

Viimeisenä vaiheena on asettaa steps-muuttujan arvo, eli valitun päivän askelmäärä näkyville textView-kenttään. SetText-metodi ottaa parametriksi String tyyppisen muuttujan, joten integer tyyppinen steps-muuttuja on vielä muutettava String-muotoon käyttämällä String.valueOf-metodia, joka palauttaa syötetyn int-muuttujan tekstimuodossa.

7 SOVELLUKSEN ULKOASUN SUUNNITTELU

Sovelluksen layout (ulkoasu) suunnitellaan Java-tiedostoista erillisissä xml-tiedostoissa. Layoutilla määritellään sovelluksen käyttöliittymän sommittelu. Kaikki layoutin elementit rakennetaan käyttämällä View- ja ViewGroup-objekteja. View-objekteja käytetään piirtämään näytölle jotain, jonka käyttäjät voivat nähdä ja jonka kanssa he voivat olla vuorovaikutuksessa. ViewGroup-objektit puolestaan ovat näkymättömiä säiliöitä, jotka määrittävät niiden sisällä olevista View-objekteista ja toisista ViewGroup-objekteista tehdyn ulkoasun rakenteen. /12, 14/

Esimerkkejä View-objekteista ovat muun muassa tässä sovelluksessa käytetyt TextView ja CalendarView. TextView-objektia käytetään piirtämään näytölle tekstikenttä ja CalendarView-objekti piirtää näytölle kalenterin, jonka kanssa käyttäjä voi myös olla vuorovaikutuksessa klikkaamalla kalenterin päivämääristä. /14/



Kuva 20. Viewgroup- ja View-objektit /12/

Kuvan 20 kaavio havainnollistaa kuinka ViewGroup-objektin sisällä voi olla useita View-objekteja sekä toisia ViewGroup-objekteja, joiden sisällä puolestaan on lisää View-objekteja. /12/

7.1 Layouttyyppejä

Android Studio tarjoaa käytettäväksi useita erilaisia layouttyyppejä, joita vaihtelemalla voidaan muokata sovelluksen ulkoasua. ViewGroup-objektit ja layoutit tarkoittavat käytännössä samaa asiaa, mutta yleisesti käytetään nimitystä layout. Tässä sovelluksessa on käytetty ainoastaan Relative layoutia, mutta muita yleisesti käytettyjä layouteja ovat muun muassa Linear Layout, Absolute layout ja List View. /12,13/

7.1.1 Linear Layout

Linear layoutia käytetään, kun halutaan asettaa useita ”lapsi-objekteja” eli käytännössä View- ja ViewGroup-objekteja peräkkäin samaan suuntaan pysty-tai vaakasuoraan. Käytännössä tämän sovelluksen käyttötarkoituksiin Linear layout olisi kelvannut, mutta päädyin käyttämään Relative layoutia siksi, että haluan tulevaisuudessa lisätä käytössä olevaan ulkoasuun lisää View-objekteja. /13/

7.1.2 Absolute Layout ja List View

Absolute Layoutia käyttämällä View-objektien sijainti voidaan määritellä tarkalleen antamalla koordinaatit mihin objekti halutaan sijoittaa. List View-layoutilla voidaan puolestaan luoda vieritettävä lista, jossa halutut tiedot on listattu päällekäin. /13/

7.1.3 Relative Layout

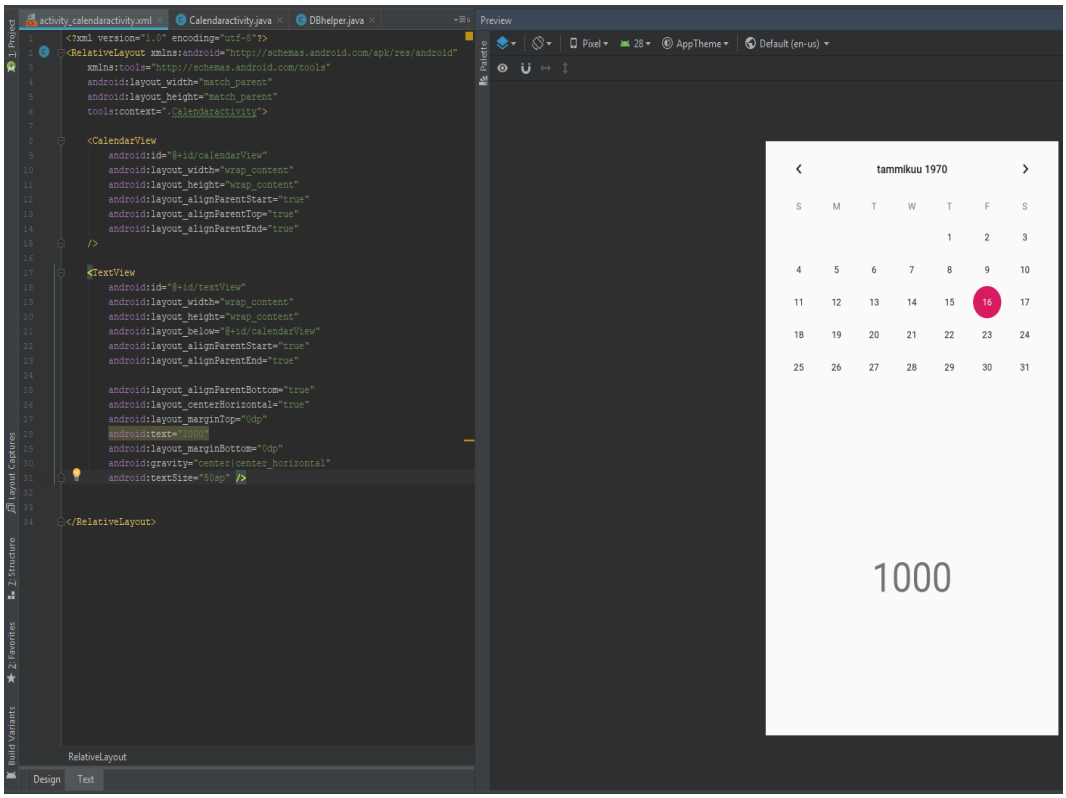
Tässä sovelluksessa käytetty Relative Layout on ViewGroup-objekti, joka antaa mahdollisuuden asetella View-objekteja suhteessa toisiin ViewGroup- ja View-objekteihin. Relative Layout valikoitui käytettäväksi tähän työhön, koska sitä on helpoin muokata, mikäli tulevaisuudessa sovellukseen halutaan lisätä uusia ominaisuuksia, jotka vaativat uusien View-objektien lisämistä vanhojen viereen tai alapuolelle. /13/

7.2 Android Studio layout editor

Android Studio-kehitysympäristö tarjoaa erityisen layout editor-työkalun sovellusten ulkoasujen suunnittelun helpottamiseksi. Layout editor aukeaa Android Studiossa oletuksena, kun käyttäjä valitsee auki olevan projektin hakemistosta xml-tiedoston jota haluaa muokata.

7.2.1 Layout editor Text-näkymä

Layout editorissa on 2 erilaista näkymää, joissa sovelluksen ulkoasua voi suunnitella. Näkymien välillä voi siirtyä painamalla layout editorin vasemman alakulman text- ja design-painikkeita. Kuvassa 21 näkyy layout editor avattuna text-näkymässä. Ruudun vasemmassa reunassa on suunniteltu ulkoasu XML-tekstimuodossa ja ruudun oikealla puolella näkyy esikatselu suunnitellusta ulkoasusta niin kuin se näkyisi käytetyn laitteen ruudulla. Text-näkymä on erittäin kätevä työkalu ulkoasun suunnitteluun, sillä xml-koodia muokatessa ruudun oikeassa reunassa näkyvä esikatselu päivittyy reaaliaikaisesti.

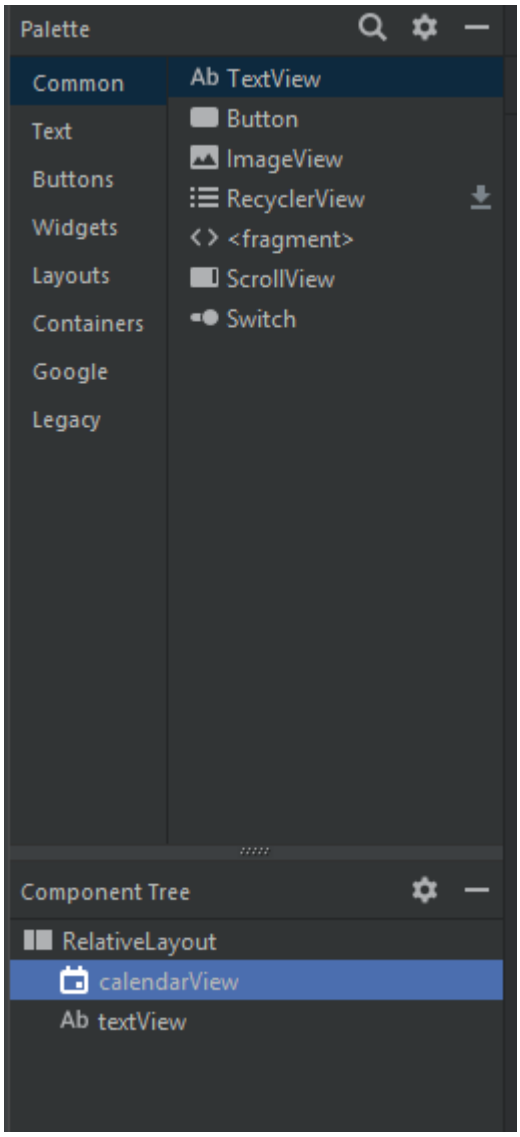


Kuva 21. Layout editor

7.2.2 Layout editor Design-näkymä

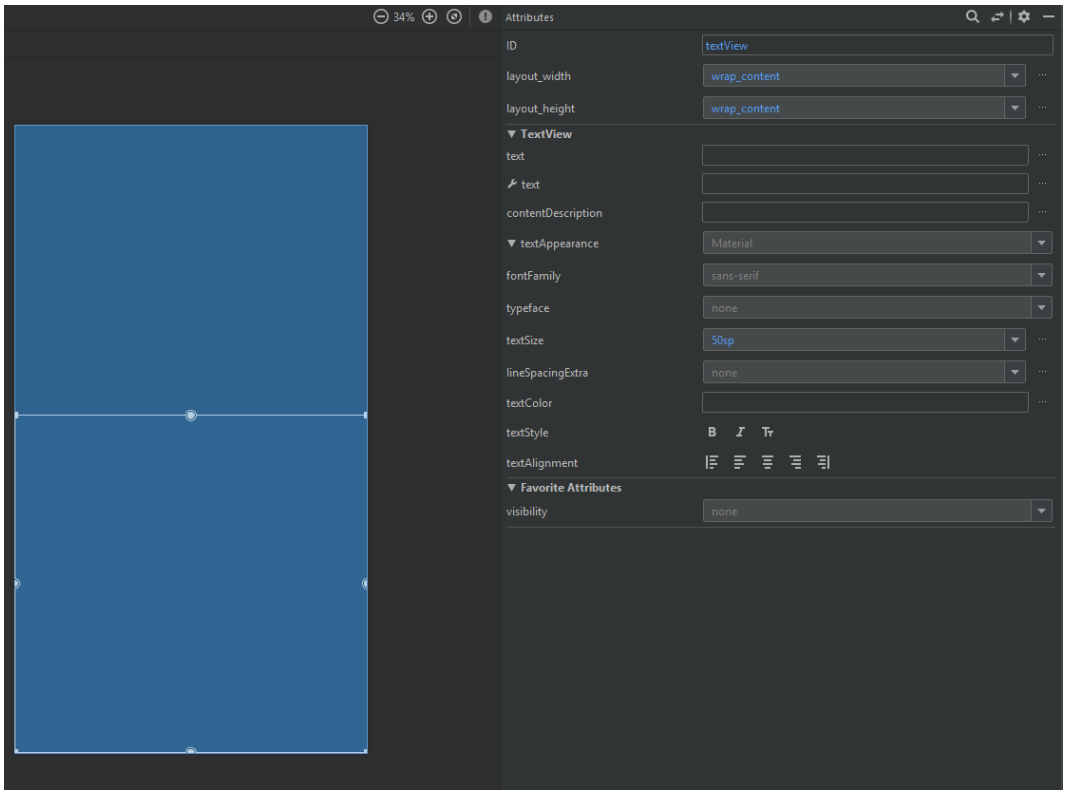
Layout editorin design-näkymässä käyttäjä pystyy hakemaan haluamiaan View-objekteja hakusanoilla tai valitsemaan paletti-valikosta valmiiksi listattuja objekteja. Kuvassa 22 näkyy design-näkymän vasemmassa reunassa sijaitseva paletti-valikko josta voi valita esimerkiksi Textview- tai Button-objektin ja raahata sen valikosta suoraan kuvassa 23 näkyvään suunnitelmaan (blueprint) ja pudottaa objektin siihen. Tätä metodia käyttämällä suunnittelijan ei edes tarvitse kirjoittaa xml-koodia, vaan

esimerkiksi TextView-objektia vastaava koodi generoituu layout editorin text-näkymään automaattisesti. Tämän jälkeen xml-koodia voi toki muokata halutunlaiseksi jos objektin oletusarvot eivät ole sopivia.



Kuva 22. Layout editor paletti-valikko

Kuvassa 23 näkyy myös oikeassa reunassa valikko, josta joitakin objektin arvoja voidaan muokata. Kuvan 23 esimerkissä on TextView-objekti ja muun muassa objektin sisällä olevan tekstin fonttia ja väriä voi muokata suoraan tästä valikosta, jolloin xml-koodi generoituu myös automaattisesti.



Kuva 23. Layout editor design-näkymä

8 ACTIVITY_CALENDARACTIVITY

Calendaractivity on toistaiseksi sovelluksen ainoa aktiviteettiluokka, joka vastaa sovelluksen käyttöliittymän toiminnollisuuksista. Activity_calendaractivity.xml-tiedostossa puolestaan määritellään aktiviteetin ulkoasu. Calendaractivity.java-tiedosto ja activity_calendaractivity.xml-tiedosto on linkitetty toisiinsa Calendaractivity luokan koodissa setContentView(R.layout.acitivity_calendaractivity)-komennolla. Uusien näkymien luonti sovellukseen vaatii aina uuden aktiviteettiluokan sekä layout-tiedoston, joka linkitetään luokkaan. /9/

Xml-koodia tehtäessä saman ulkonäöllisen lopputuloksen voi saavuttaa useilla eri tavoilla. Tässä luvussa esitettyyn lopputulokseen on päästy testaamalla useita eri variaatioita ja päätymällä testaamisen kautta mahdollisimman yksinkertaiseen, vähiten rivejä sisältävään koodiin. XML-koodissa kaikki määreet ovat oletuksena false eli epätosia, joten jos jokin asia halutaan jättää pois ulkoasusta, se voidaan yksinkertaisesti jättää kokonaan pois koodista. Jos määreitä puolestaan halutaan lisätä, ne lisätään koodiin ja arvoksi annetaan true eli tosi.

8.1 Relativelayout ja CalendarView

Kuvassa 24 näkyy Activity_calendaractivity.xml-tiedoston koodin alkuosa. Tiedoston alussa määritellään käytettävän layoutin tyypiksi RelativeLayout, jonka toiminta on selitetty tarkemmin kappaleessa 7.1.3. RelativeLayout-tagi suljetaan koodin lopussa, koska muut objektit tulevat RelativeLayout Viewgroup-objektin sisään. RelativeLayout Viewgroup-objektin leveydeksi ja korkeudeksi määritetään match_parent, mikä tarkoittaa, että objektin halutaan olevan yhtä leveä ja yhtä korkea kuin käytössä olevan laitteen näyttö.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Calendaractivity">

    <CalendarView
        android:id="@+id/calendarView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />
```

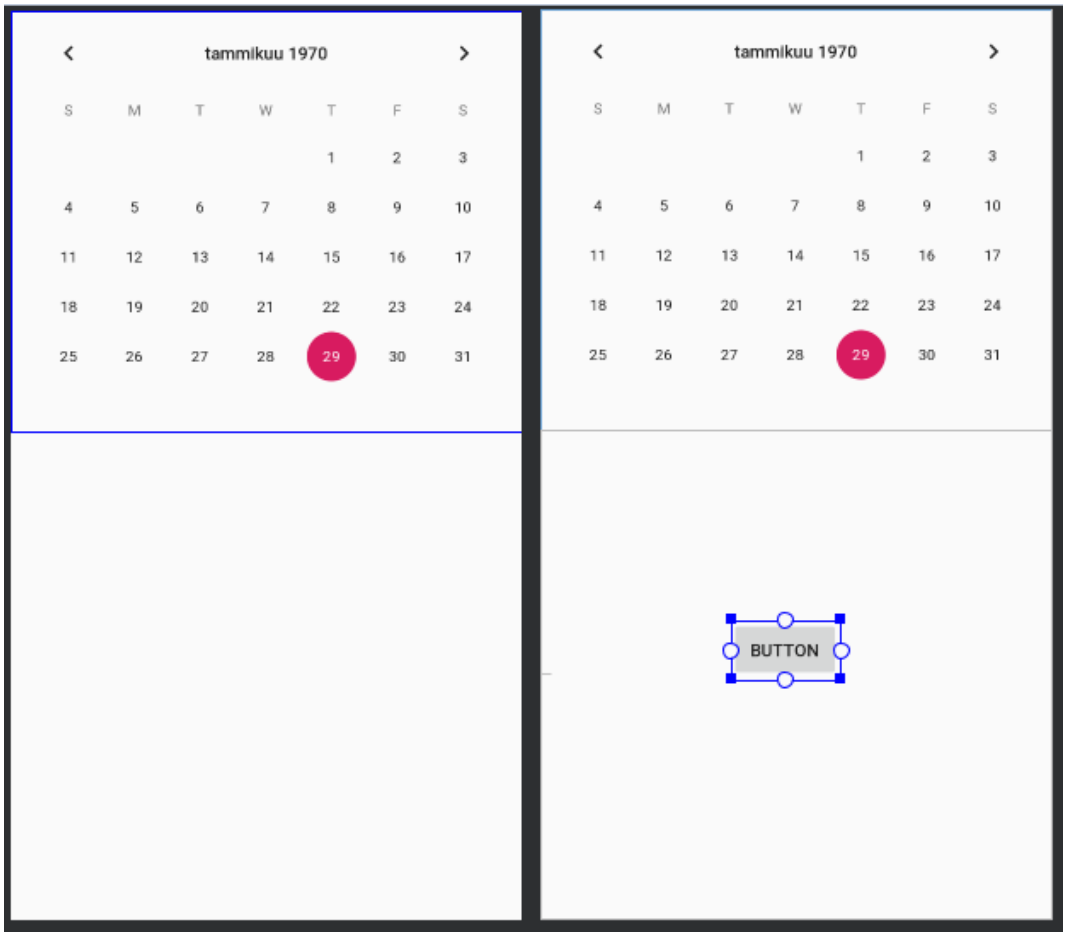
Kuva 24. Activity_calendaractivity.xml-tiedoston alkuosa

ViewGroup-objektit ovat näkymättömiä säiliöitä, joiden sisään voidaan laittaa toisia ViewGroup-objekteja tai View-objekteja. Kyseinen objekti halutaan koko näytön kokoiseksi, koska sen sisään laitetaan tässä tapauksessa CalendarView- ja TextView-objektit, jotka halutaan saada koko näytön kokoisiksi. Näiden View-objektien koon määrittely onnistuu helposti, kun ViewGroup-objekti jonka sisään ne tulevat, on määritelty ensin koko ruudun kokoiseksi.

Calendaractivity-objektille annetaan aluksi nimi, eli määritellään id komennolla android:id="@+id/calendarView". Tätä id:tä voidaan käyttää apuna kyseisen View-objektin löytämiseen Java-koodissa. Objektin leveydeksi määritellään match_parent mikä tarkoittaa, että objektin leveys vastaa sen ViewGroup-objektin leveyttä, jonka sisällä tämä objekti sijaitsee. Tässä tapauksessa CalendarView-objekti on RelativeLayout-objektin sisällä, joten koko määrittyy sen leveyden mukaan. Calendarview-objektin korkeudeksi määritellään wrap_content, jolloin objektin koko määräytyy vain sen sisällön mukaan.

Kuvassa 25 on havainnollistettuna wrap_content-määreen toiminta. Kuvan vasemmassa reunassa on kuvakaappaus layout editorista, jossa näkyy sovelluksessa käytetty CalendarView-objekti ja sinisellä viivalla objektin koon rajaus wrap_content-määreen mukaan. Objektin koko on siis juuri sellainen, että sen sisään mahtuu kaikki CalendarView-objektin sisältö. Tässä tapauksessa objekti oli myös kokonaisuudessaan sopivan kokoinen, noin puolet näytöstä, ilman että esimerkiksi objektin päivämäärien kokoa tarvitsi lähteä muuttamaan.

Lisäksi kuvassa 26 oikealla puolella on lisätty Button-objekti ja sitä ympäröivä sininen viiva näyttää objektin koon, kun objektin korkeudeksi sekä leveydeksi on määritetty ”wrap_content”. Kyseinen Button-objekti ei ole todellisuudessa käytössä tässä sovelluksessa, vaan se on lisätty vain havainnollistamisen avuksi.



Kuva 25. Viewgroup- ja View-objektit esitettynä kuvassa sinisillä viivoilla

8.2 TextView

Kuvassa 26 näkyvässä Activity_calendaractivity.xml-koodin loppuosassa annetaan TextView-objektin määreet ja suljetaan RelativeLayout-tagi. TextView-objektille määritetään aluksi id textView, jota voidaan käyttää Java-koodissa. Objektin leveydeksi määritetään match_parent, joka tekee objektista yhtä leveän kuin käytetyn

laitteen näyttö, koska parent-objekti RelativeLayout oli aikaisemmin määritetty samalla määreellä näytön levyiseksi.

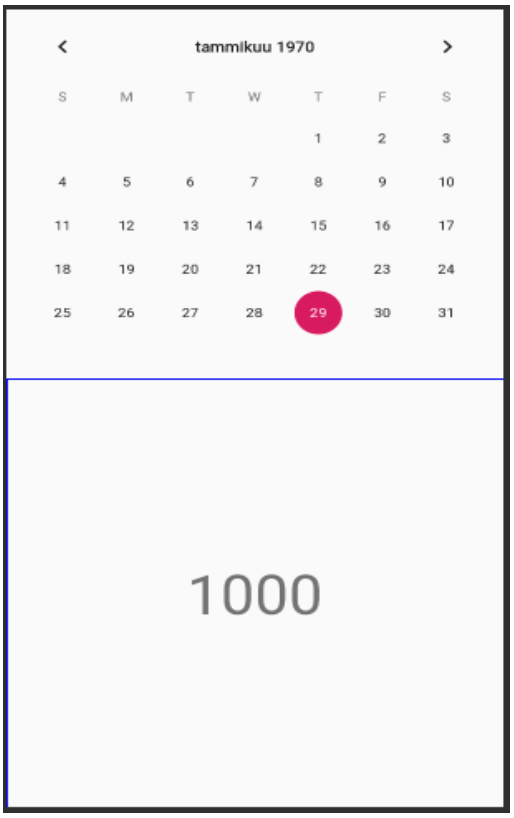
```
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/calendarView"
        android:layout_alignParentBottom="true"
        android:textSize="50sp"
        android:gravity="center"
        android:text="1000" />


</RelativeLayout>
```

Kuva 26. Activity_calendaractivity.xml-tiedoston loppuosa

View-objektit vaativat aina korkeuden sekä leveyden määrityksen. Näihin määrityksiin vaihtoehtoja on kolme: match_parent, wrap_content tai tarkka koko määritettynä pikseleinä. Tässä tapauksessa TextView-objektin korkeudeksi määritetään wrap_content eli korkeus määräytyy objektin sisällön mukaan. Todellisuudessa korkeutta muokataan vielä koodissa seuraavana tulevilla lisämääreillä.

Layout_below-määreen parametriksi asetetaan TextView-objektin id. Layout_below-määreellä saadaan kätevästi asetettua TextView-objekti heti calendarview-objektin alapuolelle. Näin saadaan CalendarView- ja TextView objektien rajat kohtaamaan. Kuvassa 27 on esitetty TextView-objektin raja sinisellä viivalla.



Kuva 27. TextView-objekti rajattuna sinisellä viivalla

Asettamalla Layout-alignParentBottom-määreen arvoksi true eli tosi, saadaan TextView-objektin alareuna vastaamaan parent-objektin eli RelativeLayout-objektin alareunaa, joka on tässä tapauksessa näytön alareuna.

TextSize-määreellä määritetään yksinkertaisesti objektin sisään tulevan tekstin koko. Olen määritellyt tämän tekstin kooksi 50sp, koska se vaikutti sopivalta koolta. Tekstin koon määrittelyyn voi käyttää useita eri yksiköitä, kuten px, dp tai sp. Tässä tapauksessa käytetty sp-yksikkö (Scale-independent Pixels) on suositeltu yksikkö tekstin koon määrittelyyn, sillä se skaalautuu näytön tarkkuuden lisäksi myös käytetyn laitteen kirjasinkoko-asetusten mukaan. Käytännössä, jos käyttäjä vaihtaa Android-laitteensa asetukista kirjasinkokoasetuksen valinnaksi, esimerkiksi suuret kirjasimet käyttöön, niin myös tämän askelmittarisovelluksen sisällä oleva teksti skaalautuu valinnan mukaan, koska se on määritetty sp-yksiköllä.

TextView-objektin määrittelyn lopuksi asetetaan vielä gravity-attribuutin arvoksi center. Gravity-attribuutti määrittelee nimensä mukaisesti View-objektin sisällön sijainnin objektin sisällä. Gravity-attribuutti määrittää sisällön sijainnin X- ja Y-akseleilla, joten center-arvo asettaa sisällön keskelle molempia akseleita. Kuvassa 27 näkyy otettujen askelien määrä (1000) asetettuna keskelle View-objektia. Kuvassa 26 näkyvä text-määre on tässä tapauksessa vain auttamassa havainnollistamaan askelmäärän sijaintia. Sovelluksen varsinaisesta koodista tämä määre on jätetty pois.

9 TESTAUSVAIHE JA VERTAILU

Askelmittarisovelluksen tarkkuuden testaamista varten asensin käytössäni olevalle OnePlus 6puhelimelle Android-sovelluskaupasta oman sovellukseni lisäksi kaksi muuta askelia mittaavaa sovellusta. Kaikki kolme sovellusta ovat olleet noin kuukauden ajan jatkuvassa käytössä puhelimellani samanaikaisesti. Vertailussa mukana olevalla Pedometer – Step counter-sovelluksella on yli 10 miljoonaa latauskertaa Android sovelluskaupassa ja StepsApp-sovelluksella yli miljoona latausta. Molemmat sovellukset ovat sovelluskaupan ladatuimpien askelmittarisovellusten joukossa.

Valitsin tarkkuusvertailua varten kuukauden ajalta yhden viikon, jossa askelmäärissä on aktiivisimman ja vähiten aktiivisen päivän välillä ollut mahdollisimman paljon vaihtelua.

Taulukko 1. Askelmäärien vertailu eri sovellusten välillä

	Pedometer – Step counter	**StepsApp**	**Askelmittari-sovellus**
7.9.2020	4067	4096	4094
8.9.2020	686	686	676
9.9.2020	1573	1573	1573
10.9.2020	3082	3082	3082
11.9.2020	2589	2589	2589
12.9.2020	9212	9212	9212
13.9.2020	2461	2462	2462
Yhteensä	**23670**	**23700**	**23688**

Taulukossa 1 on listattuna kolmen eri askelmittarisovelluksen askeleet viikon ajalta. Kaikki sovellukset ovat olleet samanaikaisesti päällä samalla laitteella. Taulukon 1 sarakkeista vasemmalta katsottuna 2 ensimmäistä saraketta on Android sovelluskaupasta ladattuja sovelluksia ja viimeinen sarake ”Askelmittari-sovellus” on tässä työssä tehty sovellus.

Viikon kestäneestä vertailusta voidaan nähdä, että tässä työssä tehty askelmittarisovellus vastaa tarkkuudeltaan hyvin tarkasti Android-sovelluskaupan suosituimpien sovellusten tarkkuutta ja päiväkohtainen suurin ero eri sovellusten välillä on suurimmillaan noin 1,5 prosenttiyksikköä. Viikkotasolla askelten yhteenlasketussa määrässä ero on alle 0,1 prosenttiyksikköä.

Sovelluksen noin 2 kuukautta kestäneessä testikäytössä on satunnaisesti huomattu, että taustapalvelu, joka vastaanottaa sensorilta askeleita ja tallentaa niitä tietokantaan, on pysähtynyt. Ongelma on havaittu noin 2 kertaa kuukaudessa ja on korjaantunut sillä, että sovellus on avattu puhelimella, jolloin palvelu on jälleen käynnistynyt. En ole pystynyt varmuudella selvittämään mistä ongelma johtuu, joten sen korjaaminen on aiheuttanut haasteita.

10 YHTEENVETO

Opinnäytetyön lopputuloksena sain kehitettyä toimivan askelmittarisovelluksen. Sovelluksen tärkeimmät toiminnollisuudet, kuten askelten laskeminen ja tallentaminen tietokantaan, ovat kunnossa ja sovelluksen kehittämistä on helppo jatkaa kehittämisvaiheessa opituilla taidoilla eteenpäin. Yksi työn tärkeimmistä tavoitteista oli oman osaamisen kasvattaminen Android-sovelluskehityksessä sekä Java-ohjelmoinnissa ja mielestäni tämä tavoite saavutettiin hyvin.

Työn suurimmat haasteet liittyivät taustalla toimivan, askelia laskevan, palvelun käynnistämiseen ja sen päällä pitämiseen. Palvelu sammuu edelleen itsestään satunnaisesti noin 2-3 kertaa kuukauden aikana ja tämän bugin korjaaminen olisi tärkeä kehityskohde, mikäli sovellusta kehitetään vielä pidemmälle. Työtä aloittaessa Android Studio-kehitysympäristö oli minulle käytännössä tuntematon, joten alussa haasteita aiheutti myös uuden työkalun käytön opettelu.

Tulevaisuudessa jatkan mahdollisesti sovelluksen kehittämistä pidemmälle ja kehityksen kohteina ovat ainakin ulkoasun parantaminen, bugien korjaaminen sekä uusien ominaisuuksien lisääminen. Uusia ominaisuuksia voisivat olla esimerkiksi kalorilaskuri ja GPS-paikannus.

LÄHTEET

/1/ Mobiililaitteiden käyttöjärjestelmien markkinaosuudet. Viitattu 14.7.2020 https://gs.statcounter.com/os-market-share/mobile/worldwide

/2/ Suosituimmat ohjelmointikielet Githubin mukaan. Viitattu 14.7.2020 https://www.businessinsider.com/most-popular-programming-languages-github-2019-11?r=US&IR=T#3-java-8

/3/ Metodin korvaaminen. Viitattu 16.8.2020 https://www.cs.helsinki.fi/u/wikla/Ohjelmointi/Sisalto/4/PeriYty.html

/4/ Android 4.4 KitKat Step Sensors. Viitattu 16.8.2020 https://www.youtube.com/watch?v=yv9jskPvLUc&t=2s

/5/ Sensoriviive Android sovelluksissa. Viitattu 16.8.2020 https://developer.android.com/guide/topics/sensors/sensors_overview

/6/ Android DatabaseHelper-luokka. Viitattu 1.9.2020 https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper

/7/ Taustapalveluiden käytön rajoitukset. Viitattu 10.9.2020 https://developer.android.com/about/versions/oreo/background

/8/ Start_sticky parametri. Viitattu 10.9.2020 https://stackoverflow.com/questions/9093271/start-sticky-and-start-not-sticky

/9/ Android aktiviteetit. Viitattu 2.9.2020 https://developer.android.com/reference/android/app/Activity

/10/ Android intent. Viitattu 19.9.2020 https://developer.android.com/guide/components/intents-filters

/11/ Android komponentit. Viitattu 19.9.2020 https://developer.android.com/guide/components/fundamentals#Components

/12/ Android layout ja ulkoasun suunnittelu. Viitattu 1.10.2020 https://developer.android.com/guide/topics/ui/declaring-layout

/13/ Android layout-tyyppejä. Viitattu 1.10.2020 https://www.tutorialspoint.com/android/android_user_interface_layouts.html

/14/ View-objektit. https://developer.android.com/reference/android/view/View Viitattu 12.10.2020

/15/ Service. Viitattu 12.11.2020 https://developer.android.com/guide/components/services

/16/ Aktiviteetit. Viitattu 12.11.2020 https://developer.android.com/guide/components/activities/intro-activities