OULUN AMMATTIKORKEAKOULU

Lauri Niku

JUOKSUSOVELLUS ANDROID-ALUSTALLE

JUOKSUSOVELLUS ANDROID-ALUSTALLE

Lauri Niku
Opinnäytetyö
Syksy 2019
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä: Lauri Niku
Opinnäytetyön nimi suomeksi: Juoksusovellus Android-alustalle
Opinnäytetyön nimi englanniksi: Running Application for Android Platform
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: Syksy 2019
Sivumäärä: 34

Opinnäytetyön tarkoituksena oli ohjelmoida juoksusovellus Android-älypuhelimille. Työssä suunniteltiin ja toteutettiin sovellus, jolla voidaan mitata juoksijan harjoitus ja tallentaa se tietokantaan myöhempää tarkastelua varten.

Kehittämistyökaluina käytettiin Android Studiota ja Google Maps SDK:ta. Sovellus kirjoitettiin Java-ohjelmointikielellä ja XML-merkintäkielellä. Tiedon tallennukseen käytettiin SQLite-tietokantaa. Sovellus sisältää myös Servicen harjoituksen seuraamiseen taustalla. Raportissa käydään läpi sovelluksen suunnittelun ja toteutuksen kaikki vaiheet.

Työn tuloksena syntyi toimiva sovellus, jossa kaikki päätoiminnallisuudet toimivat suunnitellusti. Sovelluksella voidaan mitata ja näyttää harjoituksen nopeus, matka, aika ja kuljettu reitti. Sovellusta olisi mahdollista kehittää tulevaisuudessa vielä pilvessä olevalla tietokannalla, jolloin käyttäjä voisi kirjautua sovellukseen ja säilyttää vanhat harjoitukset vaihtaessaan puhelinta.

Asiasanat: Android, Java, juoksu, mobiilisovellukset, tietokannat

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Engineering

Author: Lauri Niku
Title of thesis: Running Application for Android Platform
Supervisor: Eino Niemi
Term and year when the thesis was submitted: Fall 2019
Pages: 34

The goal of this thesis was to produce a running application for Android smartphones. An application was designed and implemented to measure the runner's training and save the collected data to a database for later use.

The development tools used were Android Studio and the Google Maps SDK. The application was written in Java programming language and XML markup language. The SQLite database was used to store data. The app also includes a service for tracking the training in the background. The report covers all stages of application design and implementation.

As a result of the work, a functional application was created, where all the main functionalities worked as planned. The application can measure and display runners exercise speed, distance, time and route traveled. In the future, it would be possible to develop the application with a cloud database, allowing the user to log in to the application and keep the old exercises when switching phones.

Keywords: Android, Java, running, mobile application, database

ALKULAUSE

Haluan kiittää opinnäytetyöni ohjaajana ja tilaajana toiminutta lehtori Eino Niemeä ja viestinnän opettajaa Tuula Hopeavuorta tuesta ja avusta opinnäytetyössäni.

Oulussa 8.10.2019

Lauri Niku

SISÄLLYS

SANASTO

Android OS	Yleisin mobiilikäyttöjärjestelmä.
GPS	(Global Posotioning System) Maailmanlaajuinen satellit-tipaikannusjärjestelmä.
Java	Android Studion tukema ohjelmointikieli.
JSON	(JavaScript Object Notation) Kevyt tiedonsiirtomuoto.
Kotlin	Androidin uusi virallinen kehityskieli.
Luokka	Määrittelee olioiden yhteiset piirteet ja rakenteen, joten kaikki saman luokan oliot sisältävät samat toiminnalli-suudet.
Olio	Luokan yksittäinen ilmentymä.
Periytyminen	Tapahtuu, kun uusi luokka perii olemassa olevan luokan ominaisuudet.
SDK	(Software Development Kit) Ohjelmistokehityspaketti.
UI	(User Interface) Sovelluksen käyttöliittymä.
Widget	Android-sovelluksen sovelluksessa näkyvä komponentti
Wi-Fi	Langaton lähiverkko.
XML	(Extensible Markup Language) Merkintäkieli Android-sovelluksien käyttöliittymien määrittelyyn.

1 JOHDANTO

Työn aiheen keksin ja valitsin, koska harrastan juoksemista. Halusin opinnäytetyön, joka liittyy jollakin tapaa siihen ja jonka aikana voin oppia mahdollisimman monipuolisesti uusia asioita Android-sovellusten kehittämisestä.

Opinnäytetyöni aiheena on Android-mobiilisovellus, jolla voi seurata juoksijan urheilusuorituksia, tallentaa ne ja tarkastella niitä myöhemmin. Sovellus laskee juoksijan urheilusuorituksen keston ja käyttää puhelimen GPS-paikannusta harjoitusten matkan ja reitin seuraamiseen, jonka avulla sovellus laskee juoksijan nopeuden. Sovellukseen sisältyy myös SQLite-tietokanta, johon juoksija voi paikallisesti tallentaa omat suorituksensa.

Juoksusovelluksia löytyy jo ennestään paljon, mutta halusin kehittää oman vastaavanlaisen, jota kehittäessä tulisin oppimaan Android-sovelluksien kehityksessä hyödyllisiä ja mielenkiintoisia asioita, kuten Google Maps SDK, GPS-paikannus, SQLite-tietokanta, Servicen-luokan toiminnallisuuksia ja käyttöliittymäosien eli Fragmentien käyttöä.

2 ANDROID-SOVELLUSKEHITYS

Android-käyttöjärjestelmän alun perin kehitti Android Inc., jonka Google osti vuonna 2005. Android on Linux-pohjainen käyttöjärjestelmä, joka on tällä hetkellä yleisin mobiililaitteiden käyttöjärjestelmä. (1.) Androidin lähdekoodi on avointa, joten sille sovellusten kehittäminen on ilmaista.

2.1 Ohjelmointikielet

Android-sovelluksia Android-studiolla voidaan tehdä Java-, Kotlin- ja C++-kielillä, joista yleisimmät ovat tällä hetkellä Java ja Kotlin (2).

2.1.1 Java

Java on Sun Microsystemsin kehittämä teknologiaperhe ja ohjelmistoalusta, johon kuuluu laitteistoriippumaton oliopohjainen ohjelmointikieli. Java julkaistiin vuonna 1995. (3.) Nykyään Java on Oraclen omistama ja on tällä hetkellä yleisin Android-kehitys kieli, jolla on tehty suurin osa Google Play -kaupan sovelluksista.

2.1.2 Kotlin

Kotlin on JetBrainsin vuonna 2011 kehittämä ohjelmointikieli. Siitä tuli ensisijainen kieli Android-sovellusten kehityksessä 7. toukokuuta 2019. Nykyään Kotlin on noussut suosituimmaksi Android-kehityksen ohjelmointikieleksi ja sitä käyttää yli puolet kehittäjistä. Suurin syy Kotlinin suosion nousemiseen Javan ohi on, että kehittäjän tarvitsee kirjoittaa vähemmän koodia. (4.)

2.1.3 XML

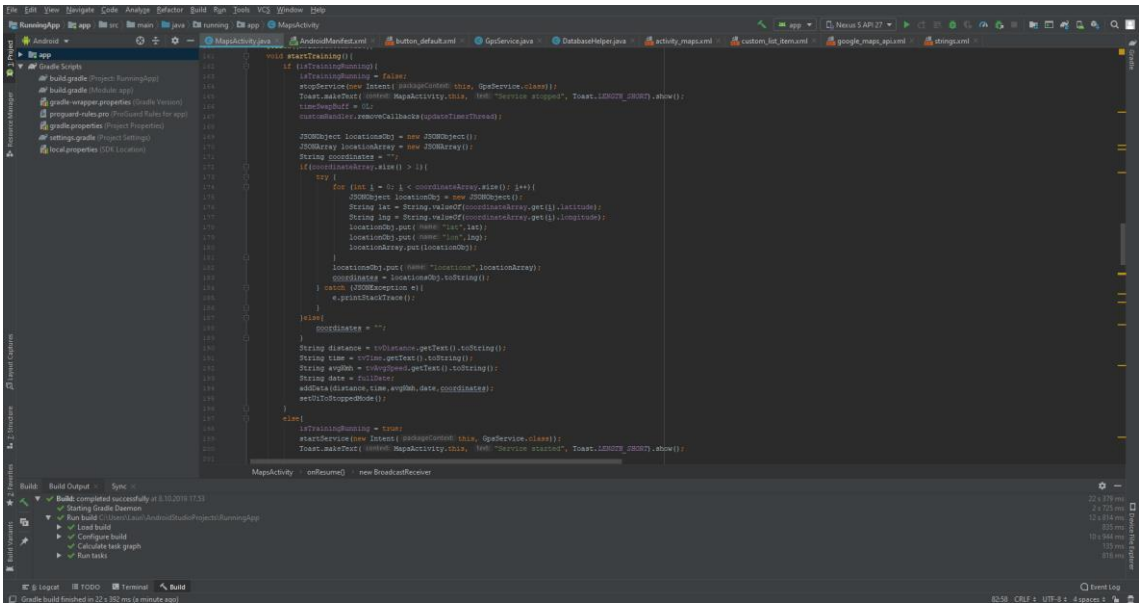
XML (Extensible Markup Language) on World Wide Web Consortiumin kehittämä merkintäkieli (5). XML-kieltä käytetään Androidissa käyttöliittymien asetteluiden ja näyttöelementtien määrittelyyn (6).

2.2 Android-älypuhelin

Älypuhelimet erottuvat matkapuhelimista niiden huomattavasti tehokkaampien laitteisto-ominaisuuksien ja laajojen mobiilikäyttöjärjestelmien avulla. Älypuhelimet tyypillisesti sisältävät esimerkiksi GPS-paikantimen ja internet- ja Bluetooth-yhteyden. (7.) Suurimmat Android-käyttöjärjestelmää käyttävät älypuhelinvalmistajat ovat Samsung ja Huawei. Sovelluksen ajamiseen ja testaamiseen käytin Huawei P10 Lite -älypuhelintani.

2.3 Android Studio-ohjelmointiympäristö

Android Studio on Googlen ja JetBrainsin kehittämä ohjelmointiympäristö Android-sovellusten kehittämistä varten (kuva 1). Android Studio on ilmainen ja se on saatavilla Windows-, macOS- ja Linux-pohjaisille käyttöjärjestelmille. Android Studio julkaistiin joulukuussa 2014, jolloin se korvasi Eclipsen Android-sovellusten kehityksessä. Android Studio tarjoaa ohjelmistokehittäjälle paljon hyödyllisiä ominaisuuksia, kuten Lint, AVD ja graafisen käyttöliittymien tekotyökalun. Lint on työkalu, joka analysoi lähdekoodin ja merkitsee ohjelmointivirheitä ja epäilyttäviä ohjelman rakenteita. AVD luo virtuaalisen Android-laitteen, ja sillä voi ajaa ja testata tehdyn sovelluksen monessa eri Android-laitteessa. (8.)

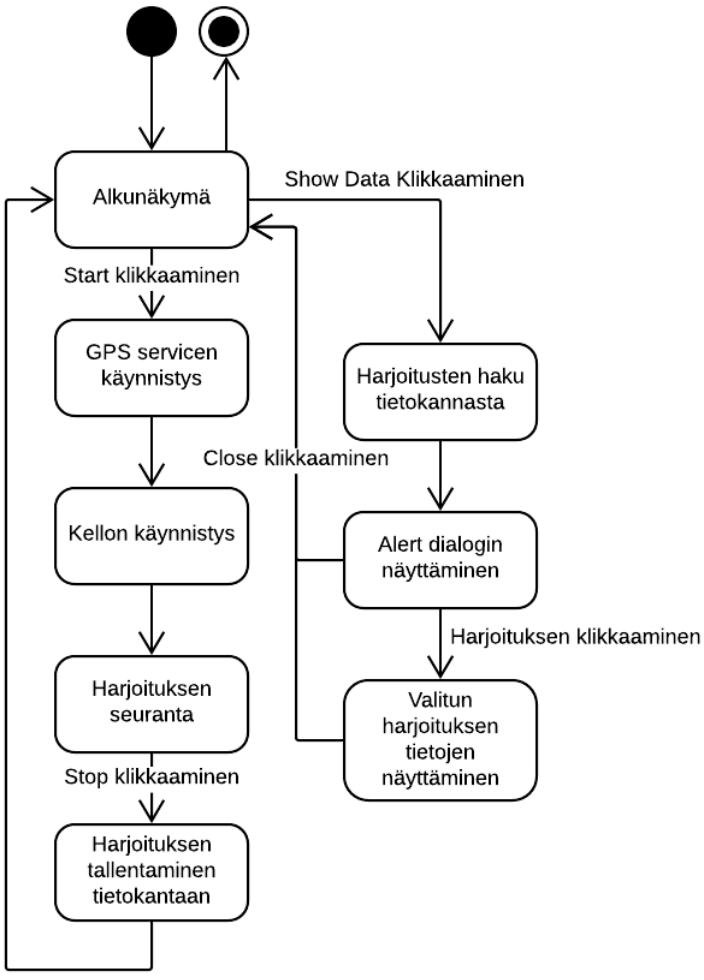


KUVA 1. Perusnäkymä Android Studiolla ohjelmoinnista

3 SUUNNITTELU

3.1 Työn rajaus

Sovelluksen rajaus aloitettiin miettimällä, mitä eri toiminnallisuuksia sovellus vaatii toimiakseen ja mitä lisätoiminnallisuuksia on realistista toteuttaa opinnäytetyössä vaaditussa tuntimäärässä. Tutustuin myös muihin urheilusovelluksiin ja niiden toiminnallisiin ratkaisuihin suunnitellessani sovellukseni toimintaa (kuva 2). Tämän jälkeen alkoi tiedon haku, millä teknologioilla halutut toiminnallisuudet voidaan tehdä.



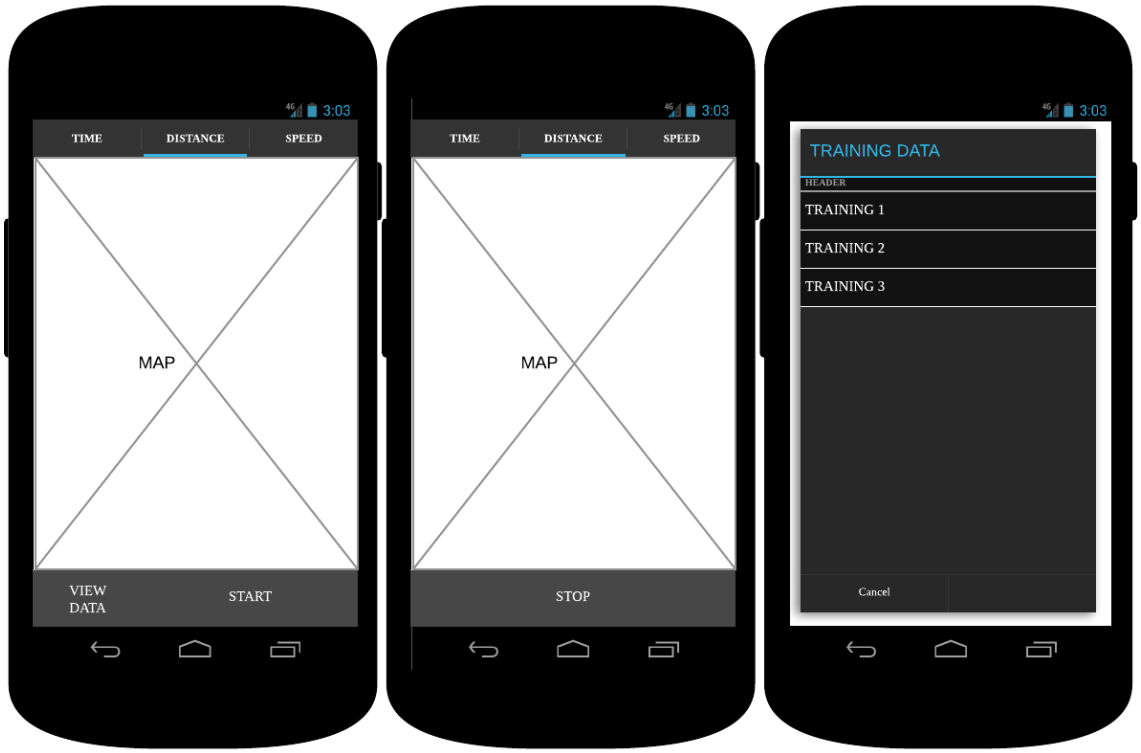
KUVA 2. Tilakaavio sovelluksen toiminnasta

3.2 Käytettyjen teknologioiden valinta

Kun oli tiedossa, mitä sovelluksen tulisi sisältää, valittiin teknologiat, joilla halutut toiminnallisuudet tehdään. Kartan tekemiseen, joka näyttää urheilijan sijainnin ja harjoituksen, valittiin Google Maps SDK, koska se on yleisin valmis ohjelmistokehityspaketti karttasovellusten tekemiseen Androidilla ja sen käyttämiseen on saatavilla helposti ohjeita (9). Ajan laskentaan käytin Android OS SystemClockia, koska olen sitä käyttänyt ajanlaskentaan aiemminkin tekemissäni Android-sovelluksissa. SystemClockin avulla voidaan toteuttaa ja seurata monia erilaisia ajan laskentaa vaativia toiminnallisuuksia (10). Sijainnin seurantaan käytin GPS-paikannusta. Harjoittelutietojen tallentamista varten valitsin SQLite-tietokannan, jolla voidaan toteuttaa sovellukselle paikallinen tietokanta puhelimen muistiin (11). SQLite-tietokannan valitsin, koska tiedon tallennuksen tarvitsi olla vain paikallinen ja sitä käyttämällä sai mahdollisuuden oppia lisää tietokannoista.

3.3 Käyttöliittymän suunnitelma

Sovelluksen käyttöliittymän suunnittelun aloitin kirjoittamalla muistioon, mitä toiminnallisuuksia ja näkymiä käyttöliittymässä tulisi olla. Sovelluksen käyttöliittymän piti sisältää seuraavat käyttöliittymäkomponentit. Fragment-luokkaa tarvittiin kartan näyttämistä varten. Fragment on käyttöliittymän osa, jolla voi esimerkiksi toteuttaa monen näytön käyttöliittymän (12). AlertDialog-luokassa näytetään edellisten harjoituksien tiedot. AlertDialog on Dialog-luokan alaluokka, jolla voidaan aktiviteetin näkymän päälle näyttää uudessa ikkunassa haluttua tietoa (13). TextView'iä käytettiin harjoituksen tietojen näyttämistä varten, jotka ovat käyttöliittymäelementtejä, joilla voidaan näyttää tekstiä käyttäjälle (14). Painikkeet (Button) lisättiin harjoituksen aloittamista, lopettamista ja AlertDialogin aukaisemista varten. Painikkeet ovat käyttöliittymäelementtejä, joilla käyttäjä voi aloittaa toimintoja klikkaamalla niitä (15). Kun käyttöliittymän vaatimukset oli suunniteltu, piirsin käyttöliittymästä LucidChart-ohjelmalla luonnokset sovelluksessa tarvittavista käyttöliittymän näkymistä (kuva 3).



KUVA 3. Lucidchartilla piirretty luonnos käyttöliittymän näkymistä

3.4 Tietokannan suunnittelu

Aluksi suunnitellaan tietokannan rakenne miettimällä, mitä eri sarakkeita tietokanta vaatii (taulukko 1). Sovellus vaati tietokannan, jossa on omat sarakkeet seuraaville tiedoille: yksilöllinen ID tietyn harjoituksen datan näyttämiseen ja omat sarakkeet harjoituksen matkalle, kestolle, keskinopeudelle ja harjoituksen tapahtumapäivämäärälle. Tietokannassa oli myös oma sarake GPS-koordinaattien tallentamista varten, että harjoituksen reitin pystyy näyttämään.

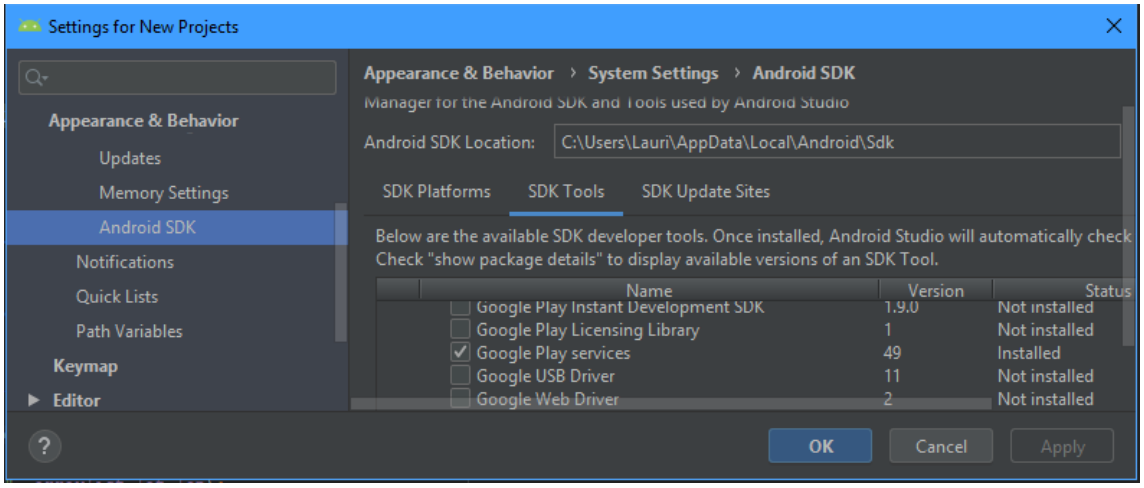
TAULUKKO 1. Tietokannan taulukon rakenne

ID	Distance	Time	Speed	Date	Coordinates

4 TOTEUTUS

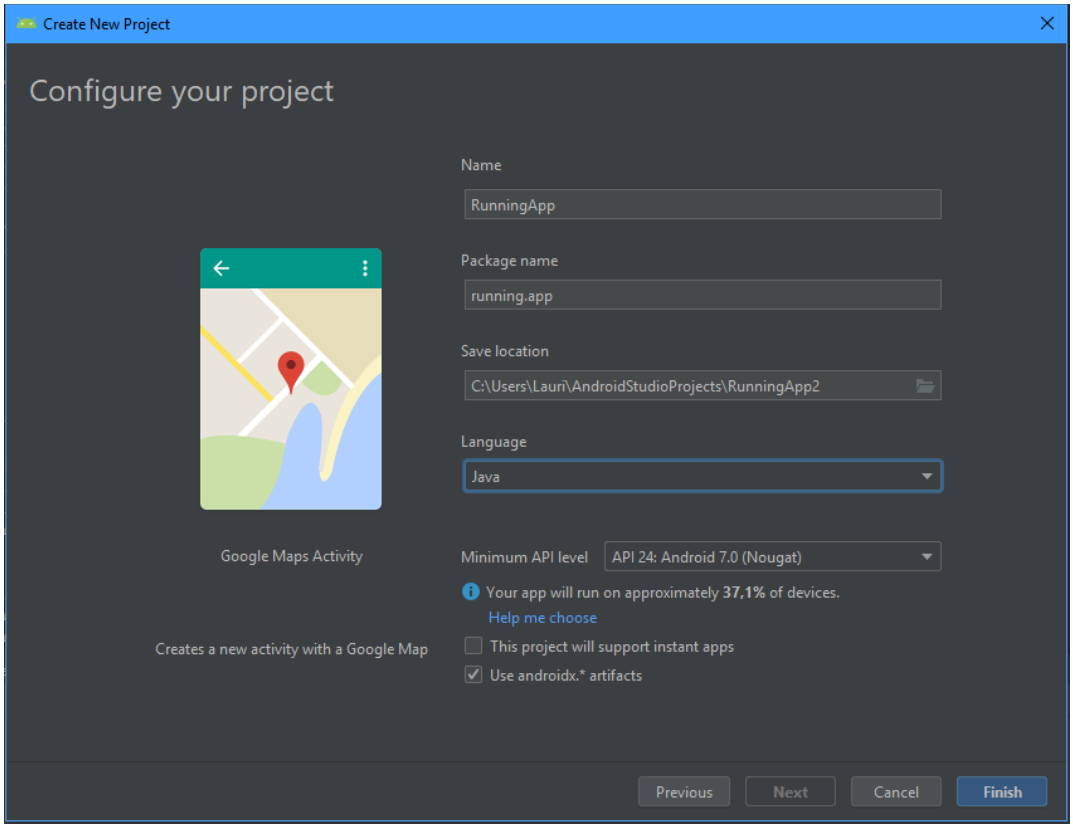
4.1 Google Maps SDK -ohjelmistokehityspaketti

Ensimmäisenä piti tarkistaa, että Google Play Services SDK on asennettuna Android Studioon (kuva 4). Tämän voi tarkistaa ja asentaa klikkaamalla Android Studion yläpalkista Tools ja valitsemalla sieltä SDK Manager. Google Play Services SDK mahdollistaa kehitettävälle sovellukselle pääsyn Googlen tarjoamiin ominaisuuksiin kuten karttoihin (16).



KUVA 4. SDK Managerin näkymä asennetuista SDK:ista

Kun Play Services SDK oli asennettu, projekti voitiin aloittaa luomalla uusi Google Maps -projekti Android Studiolla. Uusi projekti Android Studiossa luodaan klikkaamalla vasemmasta yläkulmasta File. Avautuvasta valikosta valitaan New ja sen valikosta New Project. Uuden projektin luominen avautuu. Projektiksi valitaan Google Maps Activity ja klikataan Next. Seuraavaksi määritellään projektin nimi, paketin nimi, projektin tallennuspaikka, projektin koodauskieli ja minimisovellusversio, missä sovellus toimii (kuva 5). Kun sovelluksen asetukset on määritelty, klikataan Finish, jolloin syntyy projekti, jonka käyttöliittymä sisältää kartta Fragmentin.



KUVA 5. Sovelluksen asetuksien valinta

Sellaisenaan kartta ei vielä toimi, joten sovellukselle piti antaa käyttöluvat sijaintitietoihin ja verkkoyhteyteen ja hakea Google Maps API key. API key on yksilöllinen avaintunniste, jota käytetään projektin tekemien pyyntöjen todentamiseen käyttöä ja laskutusta varten (17). Käyttölupa sijaintitietoihin ja verkkoyhteyteen annetaan AndroidManifest.xml-tiedostossa (kuva 6). Nopein tapa API keyn hakemiseen on käyttää projektin google_maps_api.xml-tiedostossa olevaa linkkiä ja seurata siellä annettuja ohjeita. Tämän jälkeen API key lisätään sovelluksen google_maps_api.xml-tiedostoon ja Google Maps SDK on valmiina käyttöön.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

KUVA 6. Kartan käyttöön tarvittavat käyttöluvat koodissa

Kun kartta oli saatu toimimaan sovelluksessa, harjoiteltiin oman sijainnin näyttämistä ja linjojen piirtämistä kartalle. Uusi linja piirretään määrittelemällä sen asetukset ensiksi PolylineOptions-luokan avulla (18) ja piirtämällä se Polylineä käyttäen (19). Näiden luokkien avulla voi piirtää kokonaisen ArrayListin koordinaattien

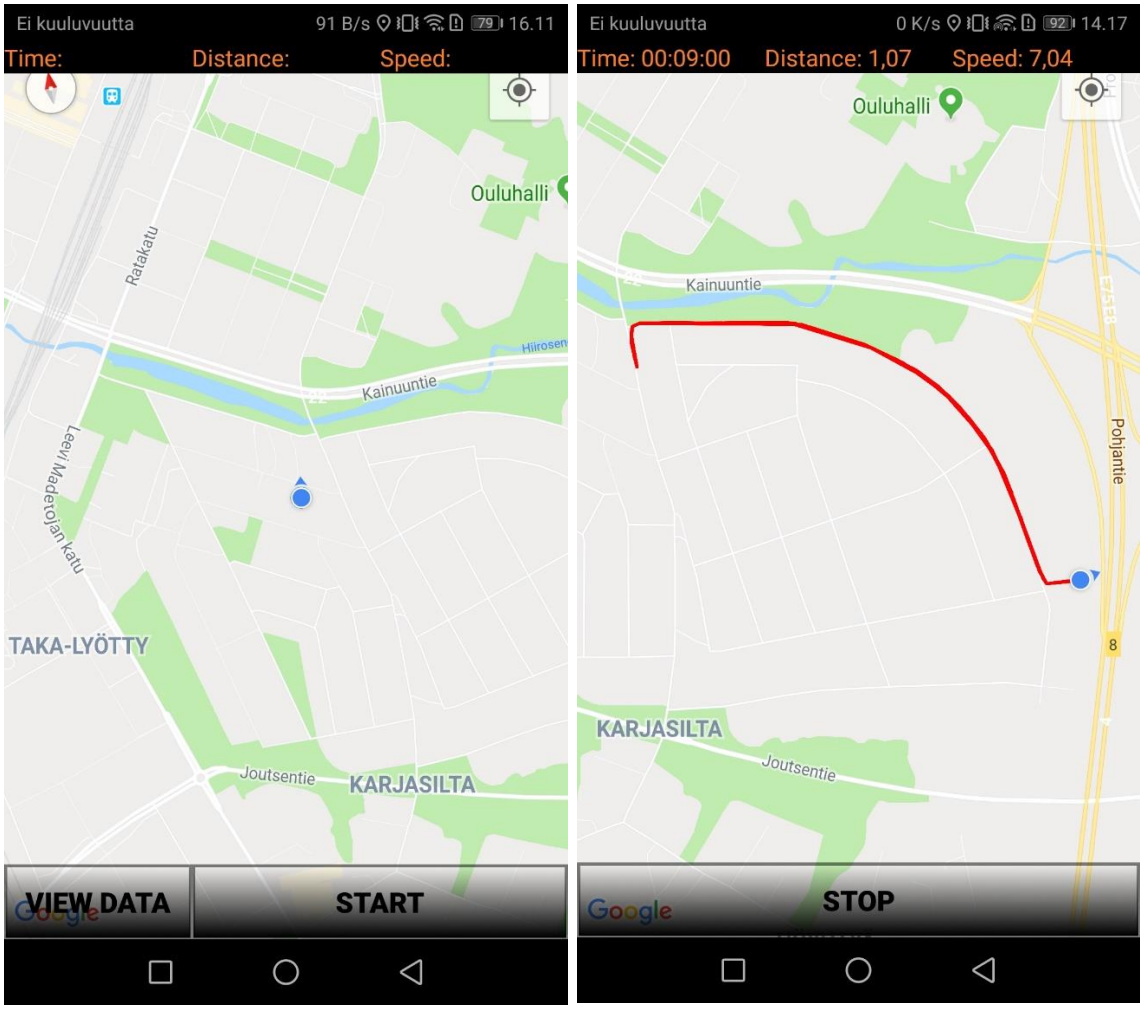
välisen matkan (kuva 7). ArrayList on Javan valmis luokka, jonka avulla voidaan säilöä haluttu määrä arvoja (20). Tässä tapauksessa käytin ArrayListiä, koska sinne voi tallentaa pareittain leveys- ja pituuspiirit.

```
PolylineOptions polylineOptions = new PolylineOptions().addAll(coordinateArray).color(Color.RED).width(6);
Polyline polyline = mMap.addPolyline(polylineOptions); //Piirtää kuljetun matkan kartalle
```

KUVA 7. Koodi ArrayListin piirtämiseen kartalle

4.2 Käyttöliittymä

Kun sovellukseen oli tehty toimiva kartta, alkoi muun käyttöliittymän rakentaminen sen ympärille. Sovelluksen käyttöliittymään tuli kolme eri mahdollista näkymää (kuva 8), jotka ovat alkunäkymä, harjoituksen seuranta ja edellisten harjoitusten tarkastelu.



KUVA 8. Sovelluksen alkunäkymä ja harjoituksen seurantanäkymä

Ensimmäisenä alkunäkymään tehtiin omat painikkeet harjoituksen aloittamiselle ja edellisten harjoitusten datan näyttämiselle. Kun harjoitus aloitetaan, muuttuu harjoituksen aloituspainike lopetuspainikkeeksi ja datannäyttöpainike piilotetaan harjoituksen ajaksi, koska silloin sitä ei tarvita. Näytön yläreunaan luotiin omat TextViewit harjoituksen keston, kuljetun matkan ja keskinopeuden seurantaan. Edellisten harjoitusten näyttämistä varten luotiin AlertDialog, jossa näytetään tietokantaan tallennetut tiedot.

Ulkoasut painikkeille tehtiin luomalla erilliset XML-tiedostot, jotka määrittelevät, miltä painikkeet näyttävät eri tiloissa (kuva 9). AlertDialogi luotiin tekemällä oma XML-tiedosto AlertDialogille, joka sisältää ListViewin ja painikkeen dialogin sulkemista varten. ListView on vieritettävä käyttöliittymäelementti, joka näyttää pystysuunnassa päällekkäin aseteltuna kaikki siihen lisätyt elementit (21). Myös AlertDialogin ListViewiä varten piti luoda oma xml-tiedosto, joka määrittelee, miltä siihen lisättävät elementit näyttävät, kun jokainen tallennettu harjoitus haetaan tietokannasta.

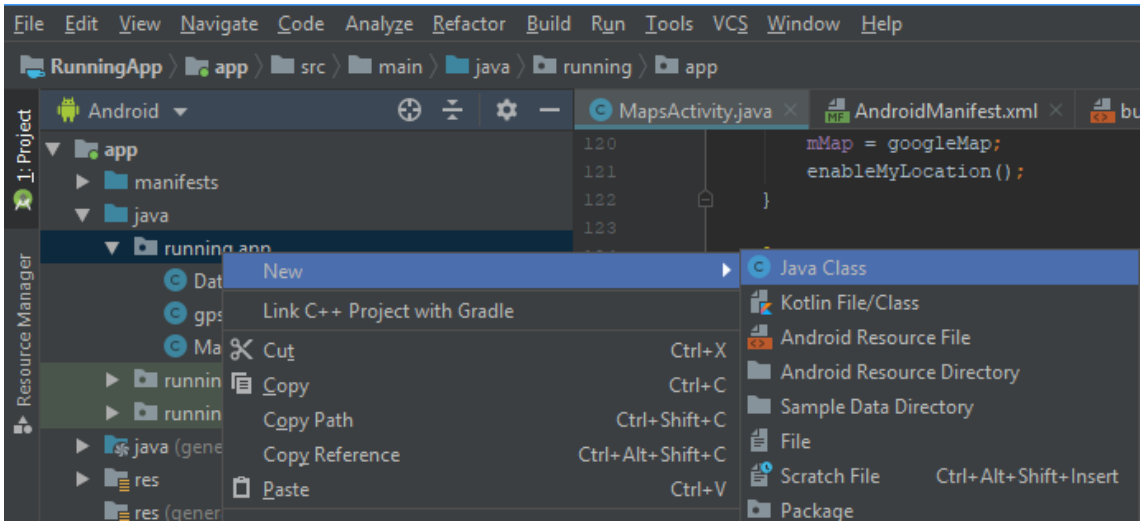
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <gradient
        android:angle="90"
        android:startColor="@color/colorBlack"
        android:endColor="@color/colorTransparent"/>
    <padding
        android:top="5pt"
        android:bottom="5pt"/>
    <stroke
        android:width="2dp"
        android:color="@color/colorTransparentBlack"/>
</shape>
```

KUVA 9. Esimerkki XML-koodista painikkeelle sen perustilassa

4.3 GPS-Service

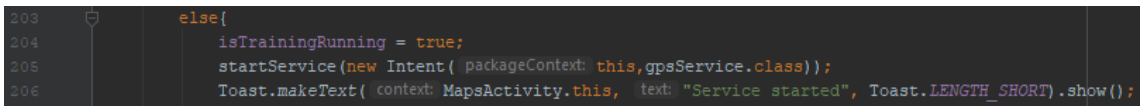
Sovellus tarvitsi GPS-paikannusta myös silloin kun puhelimen näyttö on suljettuna, joten sovellukseen piti tehdä Service (22), joka seuraa käyttäjän sijaintia jatkuvasti ja ottaa sijainnit talteen reitin näyttämistä varten. Service-luokka tehdään luomalla uusi Java-luokka. Android Studiossa uusi Java-luokka luodaan

projektin Java-tiedostokansiota hiiren oikeaa painiketta klikkaamalla. Avautuvasta valikosta valitaan New ja sen valikosta Java Class (kuva 10). Uuden luokan luominen avautuu. Luokan nimeksi annetaan gpsService ja muut asetukset jätetään oletuksiksi.



KUVA 10. Uuden Java luokan luominen

Jos sovellukselle ei olisi jo annettu käyttöoikeuksia sijaintitietoihin, ne annettaisiin tässä vaiheessa. Seuraavaksi ohjelmoitiin GPS Servicen käynnistäminen. Servicen käynnistystä varten aloituspainikkeen luokassa tarkastettiin Booleanin avulla, onko tällä hetkellä harjoitus jo kulkemassa. Jos ei Boolean asetetaan true ja Service käynnistetään (kuva 11). Boolean on muuttujatyyppi, jonka arvona voi olla joko tosi tai epätosi (23).



KUVA 11. Koodi Servicen käynnistykseen

Service-luokan koodaaminen alkoi luomalla sille aliluokka GpsService, joka perii Service-luokan ominaisuudet (kuva 12). Tämän jälkeen alustettiin tarvittavat objektit sijainnin seurantaan. LocationManager-objekti tarjoaa pääsyn käytetyn järjestelmän paikannuspalveluihin (24), joten sen avulla sovellus voi seurata säännöllisesti harjoittelijan GPS-sijaintia. LocationListeneriä käytetään LocationManagerin ilmoitusten vastaanottamiseen (25), esimerkiksi kun käyttäjän sijainti vaihtuu (kuva 12).

```
public class GpsService extends Service {
    private LocationListener listener;
    private LocationManager locationManager;
```

KUVA 12. Service-luokan periminen

Tämän jälkeen luotiin onCreate-metodi, joka suoritetaan, kun Service käynnistetään (26). OnCreate-metodin sisällä määriteltiin LocationListener ja LocationManager. LocationManagerissa asetetaan, millä tavalla sijainti haetaan. Tässä käytin pelkästään paikannusta GPS-yhteyden avulla, koska testatessani myös GPS- ja nettiyhteyspaikannusta huomasin, että sovellus ottaa häiriöitä sijaintiin talojen WiFi-yhteyksien takia. Paikannus tiheydeksi asetettiin, että käyttäjän pitää liikkua vähintään 10 metriä paikannusten välillä ja aikaa pitää kulua vähintään 5 sekuntia (kuva 13). Aina kun uusi sijainti saatiin, se lisättiin ArrayListiin.

```
locationManager = (LocationManager) getApplicationContext().getSystemService(Context.LOCATION_SERVICE);
locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,  minTime: 5000,  minDistance: 10, listener);
```

KUVA 13. LocationManagerin määrittely

Datan lähetys ja vastaanottaminen Serviceltä tapahtui Broadcast-luokan avulla (27). Ensiksi Serviceen ohjelmoitiin Broadcastin lähetysominaisuus. Kun uusi sijainti on lisätty ArrayListiin, määritellään uusi Intent-tietorakenne nimeltään arrayListIntent (28). Seuraavaksi määriteltiin uusi Bundle-olio nimeltään extra, johon ArrayList lisätään putSerializable komennolla. Tämän jälkeen arrayListIntenttiin lisätään extra putExtra-komennolla. Näin Intent on valmis lähetettäväksi ja se lähetetään kutsumalla sendBroadcast-luokkaa (kuva 14).

```
@Override
public void onLocationChanged(Location location) {
    arrayListLatLon.add(new LatLng(location.getLatitude(), location.getLongitude()));

    Intent arrayListIntent = new Intent( action: "coordinatesUpdate");
    Bundle extra = new Bundle();
    extra.putSerializable("coordinateArray", arrayListLatLon);
    arrayListIntent.putExtra( name: "extra", extra);
    sendBroadcast(arrayListIntent);
}
```

KUVA 14. ArrayListin täyttäminen ja lähettäminen

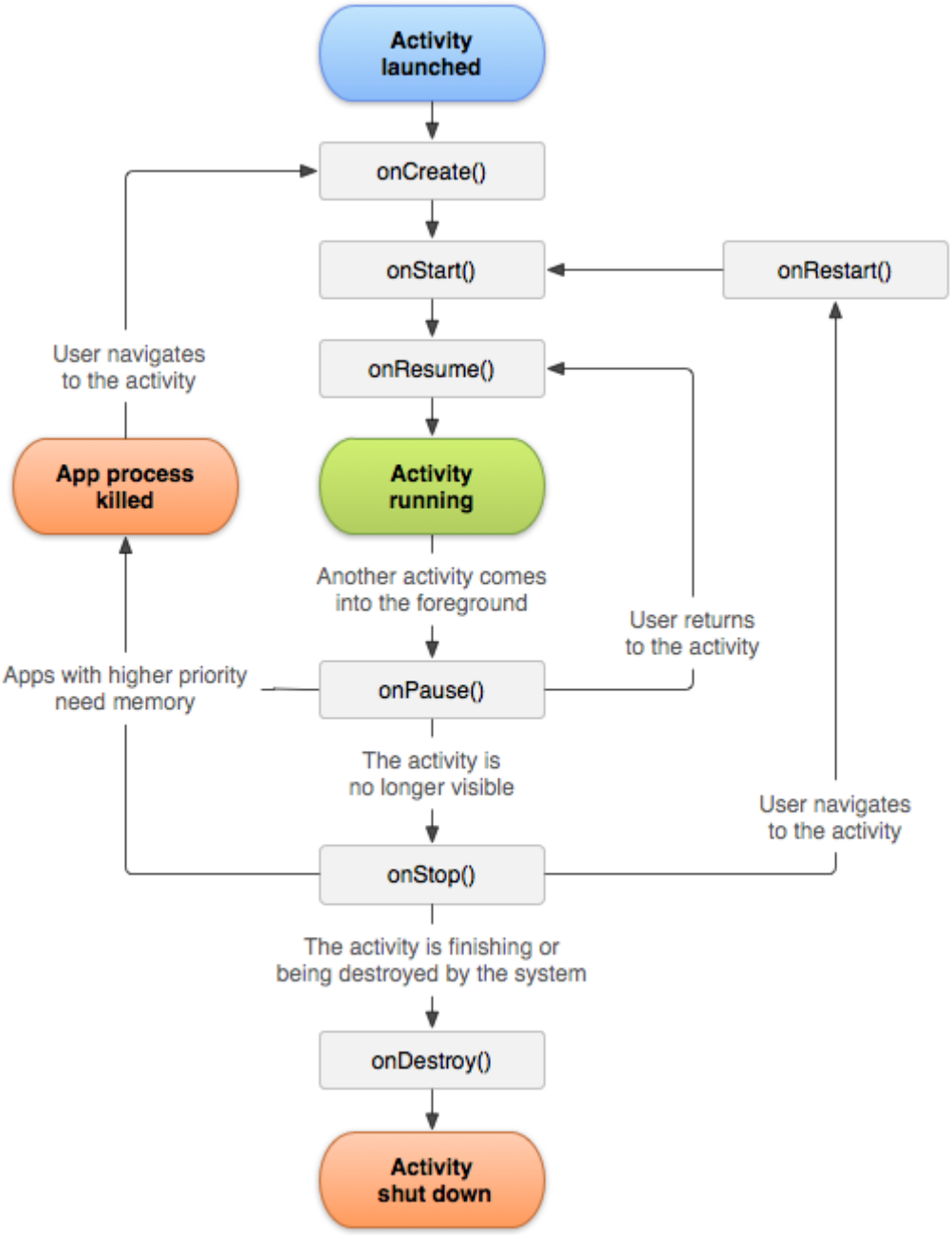
4.4 Harjoituksien seuranta

Kun käyttäjä klikkaa Start-painiketta, sovellus tarkistaa Booleanin avulla, onko harjoitus meneillään. Jos ei ole käynnistää sovellus GPS-Servicen. Seuraavaksi sovellus asettaa ajastimen aloitusajankohdan. Tämä tehdään SystemClock.uptimeMillis-komennon avulla, joka antaa millisekunteina ajan, kuinka kauan järjestelmä on ollut käynnissä (kuva 15). Lopuksi sovellus käynnistää vielä ajastinsäikeen. Säikeet ovat ohjelman suorituksessa itsenäisesti suoritettavia kevyempiä osia (30).

```
else{
    isTrainingRunning = true;
    startService(new Intent( packageContext: this, GpsService.class));
    Toast.makeText( context: MapsActivity.this,  text: "Service started", Toast.LENGTH_SHORT).show();

    startTime = SystemClock.uptimeMillis();
    customHandler.postDelayed(updateTimerThread,  delayMillis: 500);
    setUiToRunningMode();
}
```

KUVA 15. Harjoituksen laskennan käynnistäminen

Sijaintidatan vastaanottaminen Serviceltä tapahtuu BroadcastReceiverillä, jonka avulla voidaan vastaanottaa Broadcastin lähettämää dataa (31). Ensiksi alustettiin BroadcastReceiver ja onResume metodissa luodaan siitä uusi BroadcastReceiver, jotta se käynnistyy aina uudelleen vastaanottamaan Serviceltä dataa, kun sovellus on valmis vastaanottamaan tietoa, esimerkiksi kun käyttäjä palaa aktiviteettiin sen jälkeen, kun sovellus on ollut toiminnassa taustalla (32) (kuva 16).



KUVA 16. Kuva sovelluksen elinkaaren havainnollistamiseen (32)

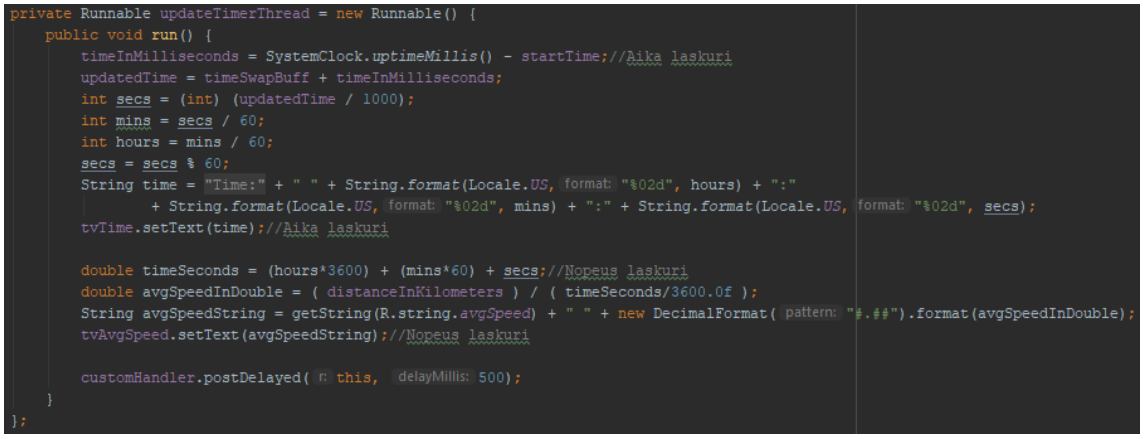
Aina kun uutta sijaintidataa vastaanotetaan, sovellus suorittaa onReceive-metodin, jossa vastaanotettu data asetetaan ensimmäiseksi coordinateArray-nimiseen ArrayListiin. Seuraavaksi sovellus tarkistaa, sisältääkö coordinateArray mitään. Jos se ei ole tyhjä, seuraavat toiminnot suoritetaan. Polyline-luokan avulla piirretään linja koko ArrayListin sijaintien välille. SphericalUtil-luokan avulla lasketaan ArrayListin välisten sijaintien matka ja asetetaan se matkan näyttämistä

varten olevaan TextViewiin (kuva 17). SphericalUtil on Google Maps SDKn sisältämä luokka, jolla voidaan laskea pituuksia, alueita ja suuntia GPS-sijaintien avulla (33). Lopuksi siirretään kartta viimeisen sijainnin kohdalle.

```
@Override
protected void onResume() {
    super.onResume();
    if(broadcastReceiver == null){
        broadcastReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                Bundle extra = intent.getBundleExtra( name: "extra");
                coordinateArray = (ArrayList<LatLng>) extra.getSerializable( key: "coordinateArray");

                if(coordinateArray != null){
                    PolylineOptions polylineOptions = new PolylineOptions().addAll(coordinateArray).color(Color.RED).width(6);
                    Polyline polyline = mMap.addPolyline(polylineOptions); //Piirtää kuljetun matkan kartalle
                    double totalDistanceInMeters = SphericalUtil.computeLength(coordinateArray);
                    distanceInKilometers = totalDistanceInMeters / 1000;
                    String distanceString = new DecimalFormat( pattern: "#.##").format(distanceInKilometers);
                    distanceString = getString(R.string.distance)+ " " + distanceString;
                    tvDistance.setText(distanceString);
                    if(coordinateArray.size() > 2){
                        LatLng currentLocation;
                        currentLocation = coordinateArray.get(coordinateArray.size() - 1);
                        mMap.moveCamera(CameraUpdateFactory.newLatLng(currentLocation));
                    }
                }
            }
        };
    }
    registerReceiver(broadcastReceiver, new IntentFilter( action: "coordinatesUpdate"));
}
```

KUVA 17. Koodi vastaanotetun datan päivittämiseen

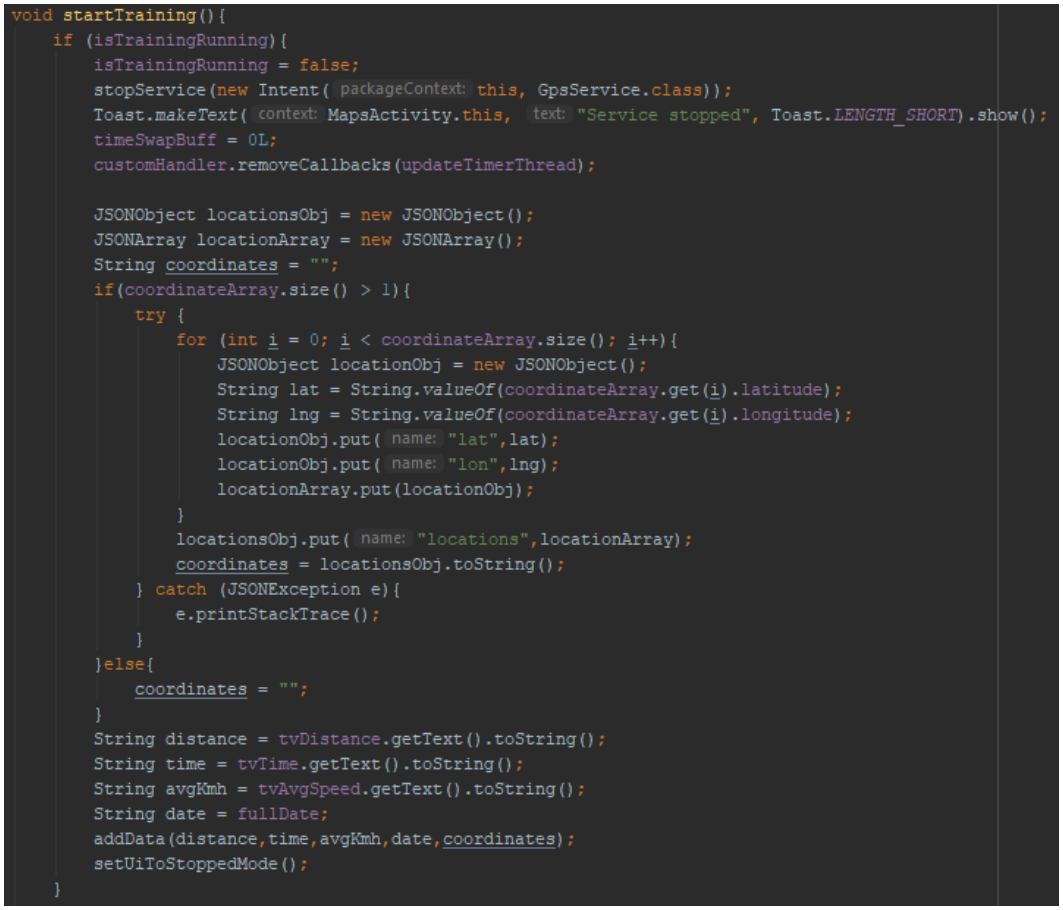
Harjoituksen keston ja nopeuden laskemiseen tehtiin uusi säie, nimeltään updateTimerThread. Ajanlaskenta tapahtuu ottamalla säännöllisin ajoin aika SystemClock.uptimeMillis-komennolla, josta vähennetään ajastimen aloitusaika startTime. Näin saadaan harjoituksen kesto millisekunteina, joka muutetaan sekunneiksi, minuuteiksi ja tunneiksi. Kun aika on muutettu haluttuun muotoon, asetetaan se ajan näyttämistä varten olevaan TextViewiin (kuva 18). Tämän jälkeen on helppo laskea nopeus ajan ja matkan avulla näytettäväksi. Lopuksi säie asetetaan pitämään puolen sekunnin tauon, koska näitä arvoja ei tarvitse päivittää jatkuvasti.



```
private Runnable updateTimerThread = new Runnable() {
    public void run() {
        timeInMilliseconds = SystemClock.uptimeMillis() - startTime;//Aika laskuri
        updatedTime = timeSwapBuff + timeInMilliseconds;
        int secs = (int) (updatedTime / 1000);
        int mins = secs / 60;
        int hours = mins / 60;
        secs = secs % 60;
        String time = "Time:" + " " + String.format(Locale.US, format: "%02d", hours) + ":"
                + String.format(Locale.US, format: "%02d", mins) + ":" + String.format(Locale.US, format: "%02d", secs);
        tvTime.setText(time);//Aika laskuri

        double timeSeconds = (hours*3600) + (mins*60) + secs;//Nopeus laskuri
        double avgSpeedInDouble = ( distanceInKilometers ) / ( timeSeconds/3600.0f );
        String avgSpeedString = getString(R.string.avgSpeed) + " " + new DecimalFormat( pattern: "#.##").format(avgSpeedInDouble);
        tvAvgSpeed.setText(avgSpeedString);//Nopeus laskuri

        customHandler.postDelayed( r: this, delayMillis: 500);
    }
};
```

KUVA 18. Koodi ajan ja nopeuden laskemiseen

Kun käyttäjä klikkaa harjoituksen lopettamispainiketta, sovellus asettaa ensimmäiseksi Booleanin arvoksi false. Seuraavaksi sovellus pysäyttää paikannus-Servicen ja ajastinsäikeen. Tämän jälkeen coordinateArraystä tehdään JSON-objekti (kuva 19). Koska JSON on kevyt tiedonsiirtomuoto (34), se on helposti muutettavissa String-muuttujaksi tietokantaan tallennusta varten ja takaisin ArrayListiksi, kun se haetaan tietokannasta reitin näyttämistä varten. Lopuksi sovellus asettaa kaikki harjoituksen tiedot muuttujiin ja kutsuu datantallennusluokkaa.



```
void startTraining(){
    if (isTrainingRunning){
        isTrainingRunning = false;
        stopService(new Intent( packageContext: this, GpsService.class));
        Toast.makeText( context: MapsActivity.this,  text: "Service stopped", Toast.LENGTH_SHORT).show();
        timeSwapBuff = 0L;
        customHandler.removeCallbacks(updateTimerThread);

        JSONObject locationsObj = new JSONObject();
        JSONArray locationArray = new JSONArray();
        String coordinates = "";
        if(coordinateArray.size() > 1){
            try {
                for (int i = 0; i < coordinateArray.size(); i++){
                    JSONObject locationObj = new JSONObject();
                    String lat = String.valueOf(coordinateArray.get(i).latitude);
                    String lng = String.valueOf(coordinateArray.get(i).longitude);
                    locationObj.put( name: "lat",lat);
                    locationObj.put( name: "lon",lng);
                    locationArray.put(locationObj);
                }
                locationsObj.put( name: "locations",locationArray);
                coordinates = locationsObj.toString();
            } catch (JSONException e){
                e.printStackTrace();
            }
        }else{
            coordinates = "";
        }
        String distance = tvDistance.getText().toString();
        String time = tvTime.getText().toString();
        String avgKmh = tvAvgSpeed.getText().toString();
        String date = fullDate;
        addData(distance,time,avgKmh,date,coordinates);
        setUiToStoppedMode();
    }
```

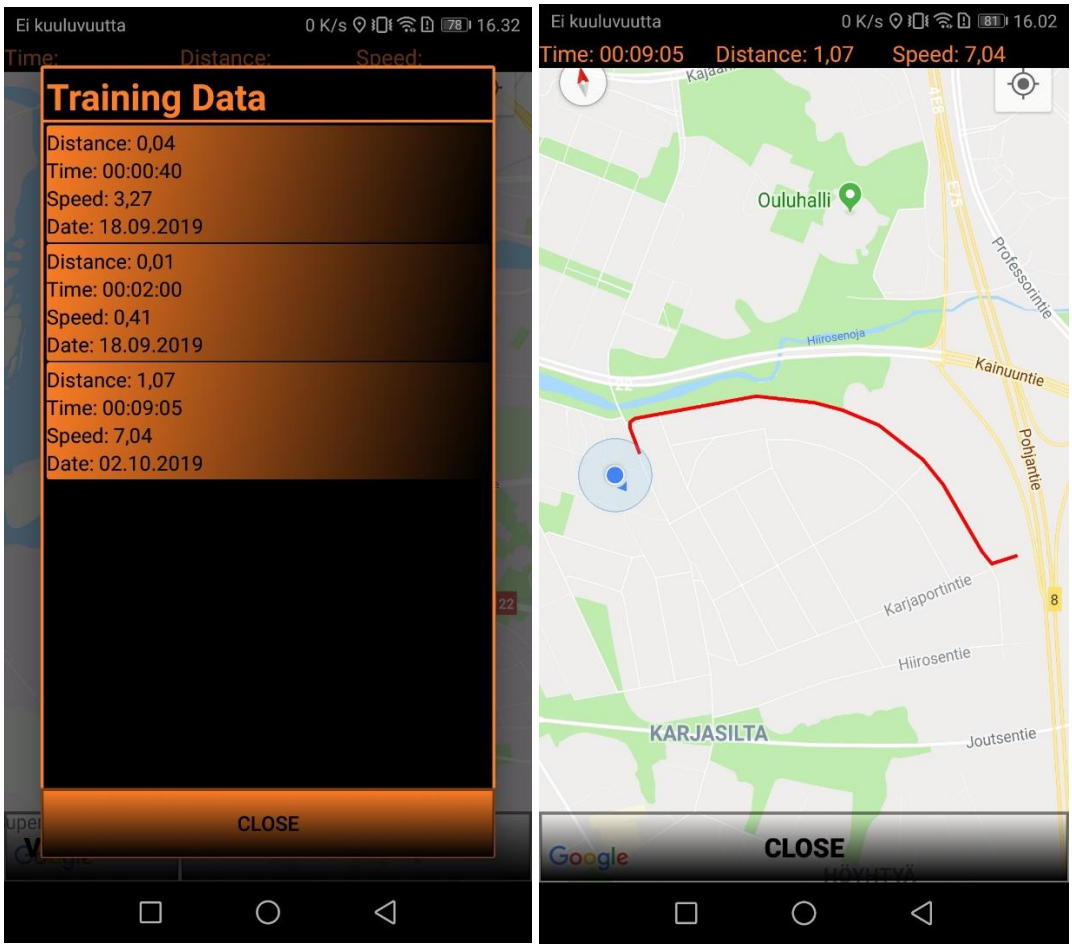
KUVA 19. Koodi JSON-objektin luomiseen

VIEW DATA -painiketta painamalla sovellus avaa AlertDialogin, johon sovellus hakee tietokannasta kaikki tallennetut harjoitukset. Jokainen harjoitus näkyy omana erillisenä klikattavana elementtinä, jossa näkyy harjoituksen matka, aika, keskinopeus ja päivämäärä. Jotta harjoitukset pystyttiin näyttämään kartalla, luotiin luokka, jolle annettiin parametreinä kaikki harjoituksen arvot. Kaikki muut arvot pystyttiin asettamaan paikalleen sellaisenaan paitsi koordinaatit, jotka olivat tällä hetkellä vielä JSON-objektissa, joka oli muutettu String-muuttujaksi. Ensiksi luotiin uusi ArrayList, joka nimettiin coordinatesToShow. Tämän jälkeen koordinaatit sisältävästä String-muuttujasta tehtiin JSON-objekti. Tämä JSON-objekti sisälsi taulukon, joka käytiin läpi for-loopin avulla. Kun taulukkoa käytiin läpi, jokainen leveys- ja pituuspiiri asetettiin ArrayListiin. Kun kaikki arvot oli lisätty, harjoituksen reitti voitiin näyttää Polylinen avulla (kuva 20).

```
public void showDataOnMap(String distance, String time, String avgKmh, String coordinates){
    tvDistance.setText(distance);
    tvTime.setText(time);
    tvAvgSpeed.setText(avgKmh);

    buttonStart.setVisibility(View.GONE);
    buttonViewData.setText(getString(R.string.buttonCloseData));
    isTrainingViewed = true;

    ArrayList<LatLng> coordinatesToShow = new ArrayList<>();
    try {
        JSONObject coordinatesObject = new JSONObject(coordinates);
        JSONArray coordinatesArray = coordinatesObject.getJSONArray( name: "locations");
        for (int i = 0; i < coordinatesArray.length(); i++) {
            JSONObject getCoordinateObject = coordinatesArray.getJSONObject(i);
            double lat = Double.parseDouble(getCoordinateObject.getString( name: "lat"));
            double lng = Double.parseDouble(getCoordinateObject.getString( name: "lon"));
            coordinatesToShow.add(new LatLng(lat, lng));
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    if(coordinatesToShow.size() >  1){
        PolylineOptions polyline_options = new PolylineOptions().addAll(coordinatesToShow).color(Color.RED).width(6);
        Polyline polyline = mMap.addPolyline(polyline_options);
    }
}
```

KUVA 20. Koodi JSON objektin muuttamiseen ArrayListiksi

Yksittäistä harjoituselementtiä klikkaamalla sovellus näyttää harjoituksen tiedot ja reitin kartalla. Harjoituksen tarkastelu suljetaan CLOSE-painiketta klikkaamalla (kuva 21). Harjoituksia on myös mahdollista poistaa tietokannasta klikkaamalla harjoitusta pitkään, jolloin sovellus kysyy, haluaako käyttäjä varmasti poistaa valitsemansa harjoituksen.



KUVA 21. Harjoitusten tarkastelua sovelluksessa

4.5 SQLite-tietokanta

Hyvä tapa SQLite-tietokannan luomiseen on luoda SQLiteOpenHelper-luokan aliluokka. SQLiteOpenHelper on apuluokka, joka sisältää hyödyllisiä ohjelmointirajapintoja helpottamaan tietokannan luomista ja sen hallintaa. (11.) Ensin luodaan julkinen luokka DatabaseHelper tietokannan ohjelmointia varten.

Tietokantaluokka luotiin luomalla projektiin uusi Java-luokka, joka nimettiin DatabaseHelperiksi. DatabaseHelper-aliluokalla periytetään SQLiteOpenHelper-luokka (kuva 22). DatabaseHelper-luokka vastaa tietokannan taulukon luomisesta ja sen datan päivittämisestä.

Aikaisemmin on määritelty, että tietokantaan tallennetaan ID, kuljettu matka, harjoituksen kesto, keskinopeus, harjoituksen mittauspäivämäärä ja kuljetun matkan

koordinaatit. Taulukkoa varten luodaan String-muuttujat taulukon ja sen sarakkeiden nimille (kuva 22).

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "trainingData.db";
    public static final String TABLE_NAME = "training_table";
    public static final String COL_1 = "ID";
    public static final String COL_2 = "DISTANCE";
    public static final String COL_3 = "TIME";
    public static final String COL_4 = "AVGKMH";
    public static final String COL_5 = "DATE";
    public static final String COL_6 = "COORDINATES";
```

KUVA 22. Koodi SQLiteOpenHelperin periyttämiseen ja taulun String-muuttujat

OnCreate-metodia kutsutaan, kun tietokanta luodaan ensimmäisen kerran. Uuden taulukon luominen tapahtuu siis tässä metodissa execSQL-metodia kutsumalla. Metodille annetaan parametriksi luotavan taulun ja sen sarakkeiden nimet ja niihin tallennettavat tietotyypit (kuva 23).

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + TABLE_NAME + "("+COL_1+" INTEGER PRIMARY KEY AUTOINCREMENT, "+COL_2+" TEXT, "+COL_3+" TEXT, "+COL_4+" TEXT, "+COL_5+" TEXT, "+COL_6+" TEXT)");
}
```

KUVA 23. Koodi uuden taulun luomiseen tietokantaan

Kun tietokantaa päivitetään jollakin tapaa, kutsutaan onUpgrade-metodia. Aluksi poistetaan vanhat tiedot taulukosta execSQL-metodin avulla. Komento vanhojen tietojen poistamiseen on DROP TABLE IF EXISTS, jonka jälkeen voidaan luoda uusi tyhjä taulukko onCreate-metodia uudelleen kutsumalla (kuva 24).

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(db);
}
```

KUVA 24. Koodi tietokannan päivittämiseen

Kun tietokantaan halutaan lisätä uuden harjoituksen tiedot, käytetään insertData-metodia. Kyseisen metodin sisällä ensimmäiseksi avataan tietokanta kutsumalla getWritableDatabase-metodia (kuva 25). Tämän jälkeen käytetään ContentVa-

lues-luokkaa, jonka avulla voidaan varastoida kokoelma arvoja tallentamista varten (35). Parametreiltä saadut arvot lisätään Put-luokkametodia käyttämällä kokoelmaan (kuva 25). Kun kaikki arvot on lisätty kokoelmaan, kutsutaan insert-funktiota, joka lisää uuden rivin tietokantaan.

```
public boolean insertData (String distance, String time, String avgKmh, String date, String coordinates) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COL_2,distance);
    contentValues.put(COL_3,time);
    contentValues.put(COL_4,avgKmh);
    contentValues.put(COL_5,date);
    contentValues.put(COL_6,coordinates);
    long result = db.insert(TABLE_NAME, nullColumnHack: null, contentValues);
    if (result == -1){
        return false;
    } else {
        return true;
    }
}
```

KUVA 25. Koodi datan lisäämiseen tietokantaan

Kun tietokantaan tallennettuja harjoituksia halutaan tarkastella, kutsutaan getAllData-metodia, jossa haetaan tietokannan jokainen rivi. Tämä metodi käyttää Cursor-rajapintaa, joka tarjoaa luku- ja kirjoitusoikeuden tietokantakyselyn palauttamaan joukkoon (36). Ensiksi metodissa aukaistaan tietokanta. Tämän jälkeen määritellään Cursor, joka tekee kyselyn ja valitsee kaikki rivit taulusta (kuva 26). Lopuksi metodi palauttaa valitun datan.

```
public Cursor getAllData(){
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res = db.rawQuery( sql: "select * from " + TABLE_NAME,  selectionArgs: null);
    return res;
}
```

KUVA 26. Koodi datan hakemiseen tietokannasta

Lopuksi ohjelmoitiin vielä datanpoisto-metodi deleteData, joka ottaa parametrin antaman ID:n tietokannan rivistä, joka halutaan poistaa. Ensiksi metodissa aukaistaan tietokanta. Tämän jälkeen käytetään Delete-funktiota, jolle annetaan taulun nimi ja rivin ID, joka halutaan poistaa tietokannasta (kuva 27).

```
public Integer deleteData (String id){
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete(TABLE_NAME,  whereClause: "ID = ?",new String[] {id});
}
```

KUVA 27. Koodi rivin poistamiseen tietokannasta

4.6 Testaus

Sovelluksen testaaminen tapahtui enimmäkseen kotonani. Aina kun sain uuden toiminnallisuuden tehtyä sovellukseen, aloin testaamaan sen toimivuutta välittömästi. Näin oli heti mahdollista tehdä siihen parantelut ja korjata virheitä ohjelmoinnissa, jos en ollut sen toimivuuteen tyytyväinen. GPS-paikannuksen testaamisen tein ulkona kävellen ja juosten ja vertailin sen tarkkuutta toisella puhelimella saman aikaisesti Huawei Health -sovelluksen kanssa.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa Android-alustalle toimiva juoksusovellus, jolla on mahdollista tallentaa ja tarkastella harjoituksia myöhemminkin. Työn lopputulokseen olen erittäin tyytyväinen, koska kaikki toiminnallisuudet toimivat niin kuin pitääkin työn aikana kehittämisessä olleista ja taloudellisista haasteista huolimatta.

Suurin haaste itse sovelluksen kehityksessä oli harjoituksen koordinaattien tallennus tietokantaan ja niiden saaminen sieltä takaisin sellaiseen muotoon, jossa niiden näyttäminen kartalla on mahdollista. Lopullisessa työssä tämäkin toimi erinomaisesti.

Kokonaisuudessaan opinnäytetyö oli sopivan haastava ja todella opettavainen Android-ohjelmoinnista ja laajempien sovelluksien suunnittelun vaikutuksesta ja tärkeydestä sovelluksen kehittämisessä. Työssä opitut asiat tulevat varmasti hyödyttämään minua tulevalla työurallani ja ratkaisemaan sovelluskehityksessä tulevia haasteita.

Sovellusta olisi mahdollista kehittää tulevaisuudessa vielä pilvessä olevalla tietokannalla, jolloin käyttäjä voisi kirjautua sovellukseen ja säilyttää vanhat harjoitukset vaihtaessaan puhelinta.

LÄHTEET

1. Android (operating system). 2019. Wikipedia. Saatavilla: https://en.wikipedia.org/wiki/Android_(operating_system). Hakupäivä 8.10.2019.

2. Android software development. 2019. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Android_software_development. Hakupäivä 8.10.2019.

3. Java. 2019. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Java. Hakupäivä 8.10.2019.

4. Kolehmainen, Aleksi 2019. Google kehottaa käyttämään kotlinia – Android-java sai naulan arkkuunsa. Tivi. Saatavissa: https://www.tivi.fi/uutiset/google-kehottaa-kayttamaan-kotlinia-android-java-sai-naulan-arkkuunsa/3818e838-3eae-4ca6-a879-e17cae6d89c4. Hakupäivä 8.10.2019.

5. XML. 2019. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/XML. Hakupäivä 8.10.2019.

6. Layouts. Android Developers. Saatavissa: https://developer.android.com/guide/topics/ui/declaring-layout. Hakupäivä 8.10.2019.

7. Smartphone. 2019. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Smartphone. Hakupäivä 8.10.2019.

8. Android Studio. 2019. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Android_Studio. Hakupäivä 8.10.2019.

9. Overview. Google Maps Platform. Saatavissa: https://developers.google.com/maps/documentation/android-sdk/intro. Hakupäivä 8.10.2019.

10. SystemClock. Android Developers. Saatavissa: https://developer.android.com/reference/android/os/SystemClock. Hakupäivä 8.10.2019.

11. Save data using SQLite. Android Developers. Saatavissa: https://developer.android.com/training/data-storage/sqlite. Hakupäivä 8.10.2019.

12. Fragments. Android Developers. Saatavissa: https://developer.android.com/guide/components/fragments. Hakupäivä 8.10.2019.

13. AlertDialog. Android Developers. Saatavissa: https://developer.android.com/reference/android/app/AlertDialog. Hakupäivä 8.10.2019.

14. TextView. Android Developers. Saatavissa: https://developer.android.com/reference/android/widget/TextView. Hakupäivä 8.10.2019.

15. Button. Android Developers. Saatavissa: https://developer.android.com/reference/android/widget/Button. Hakupäivä 8.10.2019.

16. Overview of Google Play Services. Google APIs for Android. Saatavissa: https://developers.google.com/android/guides/overview.html. Hakupäivä 8.10.2019.

17. Get an API Key. Google APIs for Android. Saatavissa: https://developers.google.com/maps/documentation/javascript/get-api-key. Hakupäivä 8.10.2019.

18. PolylineOptions. Google APIs for Android. Saatavissa: https://developers.google.com/android/reference/com/google/android/gms/maps/model/PolylineOptions. Hakupäivä 8.10.2019.

19. Polyline. Google APIs for Android. Saatavissa: https://developers.google.com/android/reference/com/google/android/gms/maps/model/Polyline. Hakupäivä 8.10.2019.

20. ArrayList. Android Developers. Saatavissa: https://developer.android.com/reference/java/util/ArrayList. Hakupäivä 8.10.2019.

21. ListView. Android Developers. Saatavissa: https://developer.android.com/reference/android/widget/ListView. Hakupäivä 8.10.2019.

22. Services overview. Android Developers. Saatavissa: https://developer.android.com/guide/components/services. Hakupäivä 8.10.2019.

23. Boolean. Android Developers. Saatavissa: https://developer.android.com/reference/java/lang/Boolean. Hakupäivä 8.10.2019.

24. LocationManager. Android Developers. Saatavissa: https://developer.android.com/reference/android/location/LocationManager. Hakupäivä 8.10.2019.

25. LocationListener. Android Developers. Saatavissa: https://developer.android.com/reference/android/location/LocationListener. Hakupäivä 8.10.2019.

26. Activity. Android Developers. Saatavissa: https://developer.android.com/reference/android/app/Activity. Hakupäivä 8.10.2019.

27. Broadcasts overview. Android Developers. Saatavissa: https://developer.android.com/guide/components/broadcasts. Hakupäivä 8.10.2019.

28. Intent. Android Developers. Saatavissa: https://developer.android.com/reference/android/content/Intent. Hakupäivä 8.10.2019.

29. Bundle. Android Developers. Saatavissa: https://developer.android.com/reference/android/os/Bundle. Hakupäivä 8.10.2019.

30. Thread. Android Developers. Saatavissa: https://developer.android.com/reference/java/lang/Thread. Hakupäivä 8.10.2019.

31. BroadcastReceiver. Android Developers. Saatavissa: https://developer.android.com/reference/android/content/BroadcastReceiver. Hakupäivä 8.10.2019.

32. Understand the Activity Lifecycle. Android Developers. Saatavissa: https://developer.android.com/guide/components/activities/activity-lifecycle. Hakupäivä 8.10.2019.

33. Maps SDK for Android Utility Library. Android Developers. Saatavissa: https://developers.google.com/maps/documentation/android-sdk/utility. Hakupäivä 8.10.2019.

34. JSON. Introducing JSON. Saatavissa: https://www.json.org/. Hakupäivä 8.10.2019.

35. ContentValues. Android Developers. Saatavissa: https://developer.android.com/reference/android/content/ContentValues. Hakupäivä 8.10.2019.

36. Cursor. Android Developers. Saatavissa: https://developer.android.com/reference/android/database/Cursor. Hakupäivä 8.10.2019.