VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jesse Ikola

ANDROID –SOVELLUS AUTON LÄMMITYSJÄRJESTELMÄN ETÄKÄYTTÖÖN

Tekniikan yksikkö
2015

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Jesse Ikola
Opinnäytetyön nimi	Android –sovellus auton lämmitysjärjestelmän etäkäyttöön
Vuosi	2015
Kieli	suomi
Sivumäärä	47
Ohjaaja	Timo Kankaanpää

Tässä opinnäytetyössä kehitettiin sovellus Android –mobiilikäyttöjärjestelmälle autojen polttoainekäyttöisten lisälämmittimien, kuten Webaston etäohjausta varten, käyttäen laitteiden väliseen kommunikointiin GSM –verkkoa. Tämä vaatii GSM –yhteyden lämmitinjärjestelmään, esimerkiksi matkapuhelimen kautta. Tarkoituksena oli suunnitella ja toteuttaa jo olemassa olevia sovelluksia yksinkertaisempi sekä älykkäämpi sovellus.

Opinnäytetyössä ei lähdetty suunnittelemaan uutta lämmitysjärjestelmää tai GSM –yhteyden implementoimista siihen vaan keskityttiin nimenomaan ohjaussovelluksen kehittämiseen. Sovelluksen tärkeimmiksi ominaisuuksiksi suunniteltiin lämmitysjärjestelmän automaattinen sytytys käyttäjän puolesta, sekä sääennusteeseen perustuva automaattisen lämmitysajan laskeminen. Näillä ominaisuuksilla voitaisiin helpottaa loppukäyttäjän sovelluksen käyttöä, sekä saavuttaa pieniä säästöjä polttoaineen kulutuksessa.

Opinnäytetyön tavoitteena oli suunnitella, kehittää ja testata sovellus julkaistavaan versioon asti, sekä suunnitella toimintojen lisäämistä sovelluksen jatkokehittämisessä.

Opinnäytetyön tuloksena saatiin kehitettyä julkaisukelpoinen Android –sovellus lisälämmittimien etäkäyttöä varten. Sovellus testattiin huolellisesti, ja se julkaistiin Google Play –sovelluskaupassa. Lisäksi suunniteltiin sovelluksen jatkokehitystä seuraavaa julkaisuversiota varten.

Avainsanat Android, auton lisälämmitin, GSM, Webasto

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Jesse Ikola
Title	Android –application for Remote Usage of Cars Heating System
Year	2015
Language	Finnish
Pages	47
Name of Supervisor	Timo Kankaanpää

The goal of the thesis was to develop application to Android –mobile operating system for remote usage of cars fuel based heating system. The solution requires GSM –connection for the heating system for example from mobile phone. Goal was to design and develop more simplified and feature rich application than the ones already existing.

In this thesis no new heating system nor telecommunication implementing was designed. Instead, full focus was on application for remote usage. As main features of the application were designed automatic start of the heating system and automatic calculation of needed heating time based on weather forecast. With these features end user's role with application could be eased and it could result with small savings in fuel usage.

The goal of this thesis was to design, implement and test application up to a release candidate, and to design features to be implemented in future development.

As a result of this thesis an Android –application was successfully developed for car's additional heating system. Application was properly tested and released in Google Play –application market. In addition, features for the next release version were designed.

Keywords Android, cars heating system, GMS, Webasto

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

TERMIT JA LYHENTEET

GSM	Global System for Mobile Communication Kansainvälinen matkapuhelinjärjestelmä
HTML	Hypertext Markup Language Hypertekstin merkintäkieli
DVM	Dalvik Virtual Machine Dalvik –virtuaalikone
ART	Android Runtime Androidin ajonaikainen ympäristö
JVM	Java Virtual Machine Java –virtuaalikone
AOT	Ahead of Time Ennenaikainen kääntäminen
IDE	Integrated Development Environment Integroitu ohjelmointiympäristö
SDK	Software Development Kit Ohjelmistokehityksen työkalupaketti
AVD	Android Virtual Device Virtuaalinen Android -laite
JRE	Java Runtime Environment Javan ajonaikainen ympäristö
JIT	Just-in-time Ajallaan tehtävä kääntäminen
JSON	JavaScript Object Notation Ohjelmointikielestä riippumaton notaatio

API	Application Programming Interface Ohjelmointirajapinta
HTTP	Hypertext Transfer Protocol Hypertekstin siirtoprotokolla
SMS	Short Message Service Matkapuhelinten tekstiviestijärjestelmä
SIM	Subscriber Identity Module Tilaajan tunnistamismoduuli
ACRA	Application Crash Reports for Android Kirjasto Android –sovelluksen kaatumisilmoitusten keräämiseen
GPS	Global Positioning System Kansainvälinen sijaintijärjestelmä

1 JOHDANTO

Polttoainekäyttöinen lisälämmitin on autoihin saatava lisävaruste, jossa käytetään auton omaa polttoainetta moottorin ja sisäilman lämmittämiseen. Tällaisen lämmittimen etuja perinteiseen lohkolämmittimeen verrattuna ovat käytön edullisuus sekä riippumattomuus sähköverkosta.

Tässä opinnäytetyössä kehitettiin Android –puhelimille uusi sovellus jolla auton polttoainekäyttöistä lisälämmitintä voitaisiin ohjata automaattisesti. Tarkoituksena oli suunnitella lämmitinvalmistajien omia ja muita markkinoilla olevia sovelluksia yksinkertaisempi ratkaisu. Samalla luoda arkkitehtuuri, joka mahdollistaa jatkokehityksen myötä tarvittavia lisäominaisuuksia. Sovellusohjauksella voidaan myös toteuttaa ominaisuuksia, jotka ovat lämmitinvalmistajalta hankittuina kuluttajalle kalliita lisäominaisuuksia. /15/

Sovellus suunniteltiin käytettäväksi lisälämmittimillä, jotka on yhdistetty GSM –verkkoon lämmittimen lisälaitteella. Tällainen voi olla esimerkiksi lämmitinvalmistajan oma tuote, tai GSM –matkapuhelimesta rakennettu vastaanotin. Täten älypuhelimet ovat ideaaleja laitteita lisälämmittimen etäkäyttöön niiden GSM –valmiuden, sekä ohjelmoitavan rajapinnan ansiosta.

2 POLTTOAINEKÄYTTÖISET LISÄLÄMMITTIMET

2.1 Yleistä

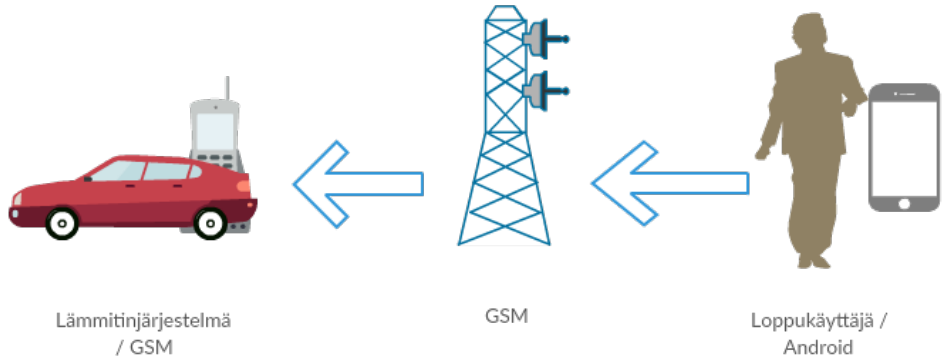
Polttoainekäyttöinen lisälämmitin on tavallisimmin auton moottoritilaan asennettava lisälaite, joka käyttää auton omaa polttoainetta moottorin, sekä mahdollisesti matkustamon ilman lämmittämiseen. Lisälämmittimiä on sekä ilma- että vesilämmitteisiä. Ilmalämmittimet toimivat erillään moottorista, ja puhaltavat lämmitetyn ilman ohjaamoon. Vesilämmittimet lämmittävät ohjaamon ilman lisäksi myös auton moottoria. Niissä lämmitinjärjestelmä yhdistetään jäähdytinnestekiertoon, ja syntynyt lämmin ilma johdetaan auton ilmakanavilla ohjaamoon. /18/

Lisälämmitintä käytetään tavallisimmin ohjaamoon asennettavalla ohjauspaneelilla. Tästä paneelista lämmitin voidaan sytyttää, sammuttaa sekä mallista riippuen säätää ajastetusti.

2.2 Lisälämmittimien GSM –valmius

Kaksi yleisintä polttoainekäyttöisten lisälämmittimen valmistajaa ovat Webasto ja Eberspächer. Molemmat valmistajat myyvät lämmittimiinsä lisävarusteina mm. GSM –vastaanotinta. Tällaisen moduulin ansioista lämmittimen ohjaaminen on mahdollista teoreettisesti miltä tahansa GSM –verkon kuuluvuusalueelta (**Kuvio 1.**). /16/

Valmistajan omien ratkaisujen lisäksi GSM –ominaisuuden lämmittimeensä voi lisätä kolmannen osapuolen valmistamalla lisälaitteella, tai sellaisen voi rakentaa itse. Yleensä tällaiset ratkaisut hyödyntävät jotain vanhempaa ja luotettavaa matkapuhelinta, josta otetaan herätesignaali lisälämmittimen käynnistämiseksi. Samalla matkapuhelimelle syötetään latausvirtaa auton akulta. Lämmitysjärjestelmään liitettyyn matkapuhelimeen soittaminen käynnistää lämmityslaitteen, esimerkiksi puhelimen tärinämoottorista saatavan signaalin perusteella. Tällaista GSM –ominaisuudella varustettua lämmitintä voidaan ohjata myös automaattisesti älypuhelimen ohjelmalla joka suorittaa herätteen antamisen käyttäjältä itsenäisesti.



Kuvio 1. GSM -yhteydellä varustetun lämmitinjärjestelmän toimintaperiaate.

2.3 Lisälämmittimen tarkasta ajastamisesta saatava hyöty

Tällaisten lämmitinjärjestelmien isoin ongelma on niiden virrankulutus. Vaikka itse lämmitin ei käytä juurikaan virtaa, tarvitaan lämmitetyn ilman siirtämiseksi ohjaamoon auton omia puhaltimia. Tyypillisesti polttoainekäyttöiset lisälämmittimet kuluttavat n. 40-80W tehon, mikä tarkoittaa n. 3-7A virrankulutusta. Isoksi ongelmaksi muodostuu virrankulutus kaupunkiajossa, jolloin auton akku ei ehdi lataantumaan tarpeeksi lämmityksen jälkeisen lyhyen ajon aikana. Näin ollen tarkasti ajoitetun lämmityksen käynnistymisen tuoma 10 minuutinkin säästö voi olla tärkeä. Lisälämmittimien polttoaineen kulutus on melko pieni; vain noin kaksi-kolme dl lämmityskertaa kohti. Pitkällä aikavälillä kuitenkin tarkalla lämmitysajalla saadaan aikaan säästöjä myös polttoaineen kulutuksessa. /16; 23/

3 KÄYTETYT TEKNIIKAT

3.1 Android –mobiilikäyttöjärjestelmä

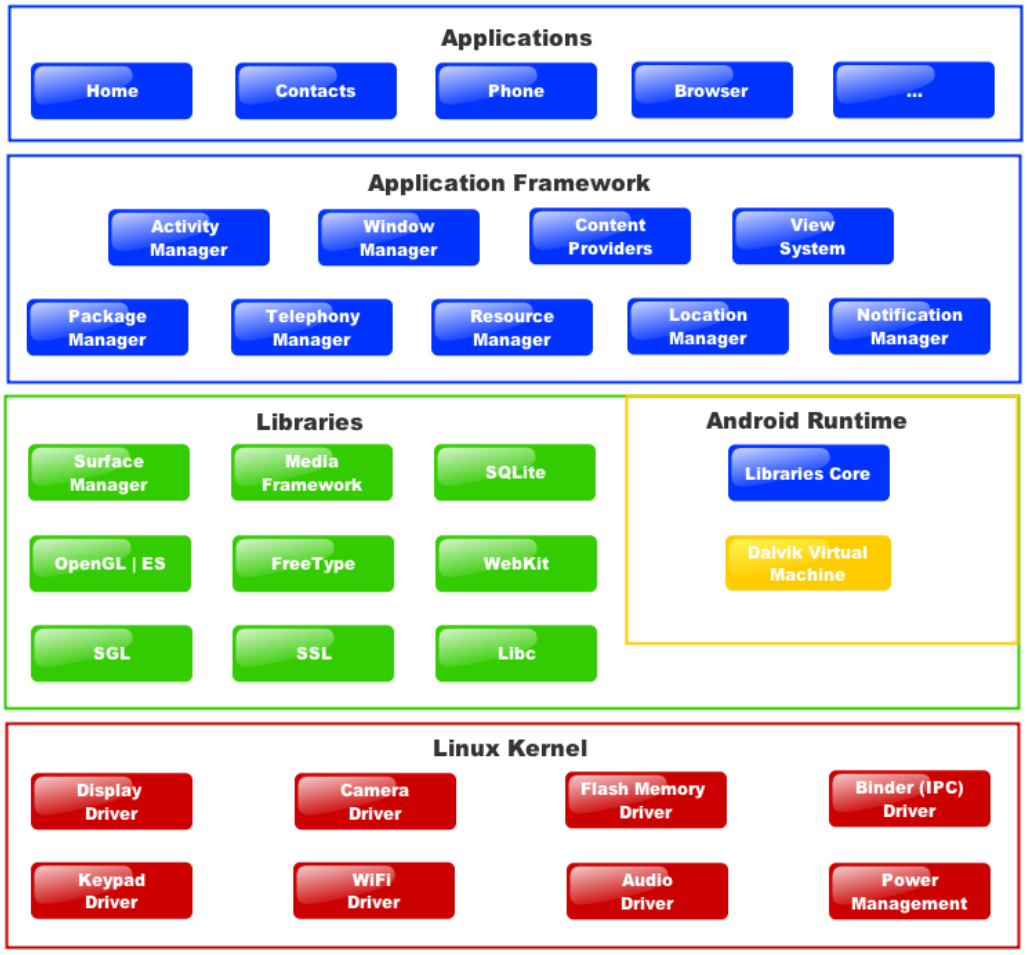
Android on avoimen koodin käyttöjärjestelmä, jota kehittää ja ylläpitää Google. Android on kustomoitava ja helppokäyttöinen käyttöjärjestelmä, joka on käytössä yli miljardilla laitteella ympäri maailmaa. Android -ympäristöä on kehitetty puhelimien ja tablet –tietokoneiden lisäksi myös kelloille, TV –laitteille ja autoille. /3./

Työn tekohetkellä uusin Android –versio on Lollipop 5.1. Lollipop julkaistiin marraskuussa 2014, ja sen tärkeimpiä ominaisuuksia olivat mm. uudistukset käyttöliittymään, parannukset ilmoitustenhallintaan, akkukeston parannukset sekä smart lock –ominaisuus. /7/

Tässä opinnäytetyössä kehitetty Android –sovellus kohdennettiin Android järjestelmille API –tasosta 15 eteenpäin. Tämä tarkoittaa sitä, että sovellus on tuettu Android versio 4.0.3 Ice Cream Sandwichissa, ja sitä uudemmissa käyttöjärjestelmissä. Tämä joukko kattaa 95,7 % Android –laitteista. /17/

3.1.1 Androidin arkkitehtuuri

Androidin arkkitehtuuri koostuu useista hierarkisista tasoita, jotka sisältävät ohjelmia eri toimintojen suorittamiseen (**Kuvio 2.**). Alimmalla tasolla toimii Linux Kernel, joka sisältää laitteiston ajurit. Kernelin tehtävänä on välittää ylemmiltä tasoilta tulevat käskyt laitteiston raudan suoritettavaksi. /5/



Kuvio 2. Android –käyttöjärjestelmän arkkitehtuuri. /4/

Kernelin päällä toimivat Androidin kirjastot. Kirjastojen tehtävänä on huolehtia datan käsittelystä. Esimerkiksi kirjasto SQLite on tietokantamoottori, joka huolehtii datan säilömisestä laitteeseen, ja WebKit on selainmoottori HTML –sisällön näyttämiseen. Samalla tasolla kirjastojen kanssa toimii myös Androidin ajoympäristö (Android Runtime). /5/

Android Runtimeen kuuluu Dalvik Virtual Machine (DVM), tai Android Runtime (ART), sekä ydinkirjastot (Core Libraries). DVM on prosessivirtuaalikone ohjelmien ajamiseen. Osa Androidin kirjastoista, sekä kaikki Google Play –kaupasta ladatut sovelluksen suoritetaan sen sisällä. Se poikkeaa Sunin Java Virtual Machinesta (JVM) esimerkiksi siten, että se ei käytä ohjelmien ajamiseen .class tiedostoja vaan niistä luotuja .dex tiedostoja joiden tavurakennetta voidaan muo-

kata asennuksen jälkeen. Tämä parantaa usein hieman tehottomien matkapuhelimien suorituskykyä. Android –versiosta 5.0 Lollipop alkaen DVM on korvattu ART:llä, joka on sitä kehittyneempi prosessivirtuaalikone. /24; 4/

ART sisältää mm. Ahead of Time (AOT) kääntämisen ohjelmille, joka kääntää ne asennusvaiheessa konekielelle ajonopeuden parantamiseksi, sekä merkittäviä parannuksia ohjelmien suorituksen aikaiseen muistinhallintaan. Lisäksi Androidin ajoympäristö sisältää ydinkirjastot, joihin kuuluu ohjelmoinnissa käytettävät Androidin luokat, osa Javan luokista, sekä esimerkiksi Apache –luokkia. /19/

Kahdella ylimmällä tasolla Androidin arkkitehtuurissa toimivat ohjelmistokehys- sekä ohjelmakerrokset. Ohjelmistokehys tarjoaa sovelluskehittäjille palveluita, joita sovellukset voivat käyttää. Tällaisia palveluita ovat mm. Activity Manager, joka hallinnoi ohjelmien elinkaarta ja Telephony Manager, joka hallinnoi laitteen puheluita. Ohjelmakerroksessa toimivat kaikki Androidin ohjelmat. Näitä ovat mahdollisesti laitteeseen valmiiksi asennetut ohjelmat, kuten selain ja viestisovellus sekä kaikki laitteeseen sovelluskaupasta ladattavat ohjelmat. /4/

3.2 Kehitysympäristö

Sovellus kehitettiin käyttäen Googlen julkaisemaa Android Studiota. Se sisältää IntelliJ IDEAn päälle rakennetun IDEn, Androidin SDK –työkalut, käyttöliittymän suunnittelutyökalun, AVD Managerin sekä Gradle –kokoamistyökalun. Gradlen ansioista omasta ohjelmastaan pystyy luomaan useita APK –paketteja eri ominaisuuksilla. Android Studiosta oli käytössä versio 1.3.x. /6/

Kehitysympäristöä käytettiin Applen Macintosh –tietokoneella, jossa toimi OS X –käyttöjärjestelmän versio 10.10 Yosemite.

3.3 Java

Yleisin menetelmä kehittää Android –sovelluksia on käyttää Javaa, kuten tässäkin opinnäytetyössä. Java on ohjelmointikieli, ja -ympäristö, jonka Sun Microsystems julkaisi alun perin vuonna 1995. Java on ohjelmointiparadigmaltaan oliopohjainen, mm. Objective-C, ja C++ kielistä vaikutteita ottanut ohjelmointikieli. Sen

suurimpia vahvuuksia on mahdollisuus suorittaa sillä kirjoitettuja ohjelmia lähes millä alustalla tahansa. Javaa käytetään Javan ajonaikaisen ympäristön avulla (JRE), joka sisältää esimerkiksi JVM:n, sekä Javan ydinluokkia. JVM on virtuaalikone, jonka sisällä Javalla kirjoitettujen ohjelmien ajaminen tapahtuu. Tällä tavoin ohjelmien suorittamisesta saadaan turvallisempaa kuin esimerkiksi C:llä, tai C++:lla, jossa ohjelmilla on suora pääsy laitteiston fyysisiin resursseihin. Toisaalta virtuaalikoneen käyttö ohjelmien suorittamiseen heikentää hieman laitteiston suoritustehoa. Tätä suorituskyvyn puutetta ollaan saatu kurottua kiinni käyttämällä Just-in-time (JIT) kääntämistä, eli ohjelmien kääntämistä vasta tarvittaessa ajon aikana tavukoodiksi eikä etukäteen kuten perinteisesti. Myös Androidin edellinen prosessivirtuaalikone Dalvik käytti JIT kääntämistä hyväkseen. Java on ollut avointa lähdekoodia vuodesta 2006 lähtien. /25; 13/

3.4 Kolmannen osapuolen kirjastot ja rajapinnat

3.4.1 GSON

GSON on Googlen Java –kirjasto olioiden kääntämiseksi JSON formaattiin, ja vastaavasti JSON –tekstistä takaisin olioiksi. GSON pystyy kääntämään JSON:ksi myös valmiiksi luodut oliot pelkän datan perusteella ilman pääsyä ohjelman lähdekoodiin.

GSON:in toJson(), ja fromJson() metodien käyttäminen omassa koodissa on helppoa. Geneeristen olioiden käsittelykyvyn vuoksi GSON:a käyttävä koodi on helposti uudelleenkäytettävissä (**Kuvio 3.**). /8/

```
    //        this method turns JSON String to an arrayList of objects
    public static List ConvertToList(String JSONString, String SHARED_PREFS) {
        Gson gson = new Gson();

        Type type;

//        we create a list based on SHARED_PREFS string
//        this is defined in Type type
        if (SHARED_PREFS.equals("profiles")) {
            type = new TypeToken<ArrayList<Profiles>>() {
            }.getType();
        } else {
            type = new TypeToken<ArrayList<TimedStarts>>() {
            }.getType();
        }

        List<Object> sharedPrefList = gson.fromJson(JSONString, type);
        return sharedPrefList;
    }
```

Kuvio 3. GSON –kirjaston käyttäminen olioiden muodostamiseen.

JavaScript Object Notation (JSON) on kevytrakenteinen tiedonvaihtoformaatti. Sen etuina on helppolukuisuus ihmisille, ja käsittelynopeus koneille. JSON koostuu sarjasta avain-arvo –pareja, tai listasta haluttuja arvoja. JSON on ohjelmointikielestä riippumaton formaatti. /10/

3.4.2 Joda-Time

Joda-Time on kirjasto Javan aika- ja päiväluokkien korvaamiseksi. Se on kehitetty korvaamaan Javan omat luokat luvaten niitä rikkaamman kokonaisuuden. Työssä käytettiin Joda-Timen uusinta versiota 2.8.2 muuttamaan aikoja eri aikavyöhykkeiden välillä. Opinnäytetyöksi tehdyssä ohjelmassa käytetty sääennuste palauttaa tiedot käyttäen UTC –aikavyöhykettä, ja Joda-Timen avulla käyttäjän ajat saatiin helposti muunnettua samaan aikavyöhykkeeseen (**Kuvio 4.**). /12/

```
//            Weather forecast returns times in UTC
 //           Format our start time to UTC
//            Formatter for our datetime
            DateTimeFormatter dtF = DateTimeFormat.forPattern("HH:mm");
 //           Convert time in string to DateTime
            DateTime dt = dtF.parseDateTime(alarmTime);
 //           Convert to DateTime in UTC
            DateTime dtUTC = dt.withZone(DateTimeZone.forID("UTC"));
```

Kuvio 4. Joda-Timen käyttäminen aikavyöhykkeiden muuntamiseen.

3.4.3 OpenWeatherMap

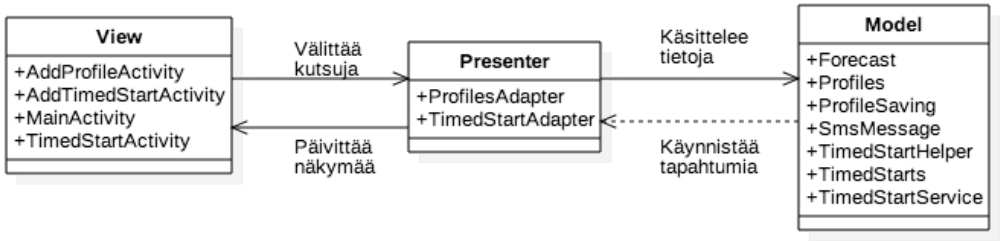
OpenWeatherMap tarjoaa ilmaisen rajapinnan sääennusteiden noutamiseen. OpenWeatherMap tarjoaa yli 200 000 kaupungin säätiedot, sekä säätietojen kyselemisen koordinaattien perusteella. Data kerätään yli 40 000 sääasemalta ympäri maailmaa, ja tietojen noutaminen tapahtuu helppokäyttöisen rajapinnan avulla. /26/

Tässä opinnäytetyössä OpenWeatherMapia käytettiin käyttäjän sijainnin sääennusteen saamiseen. Tämän tiedon perusteella laskettiin ihanteellinen lämmitysaika käyttäjän auton lämmityslaitetta varten.

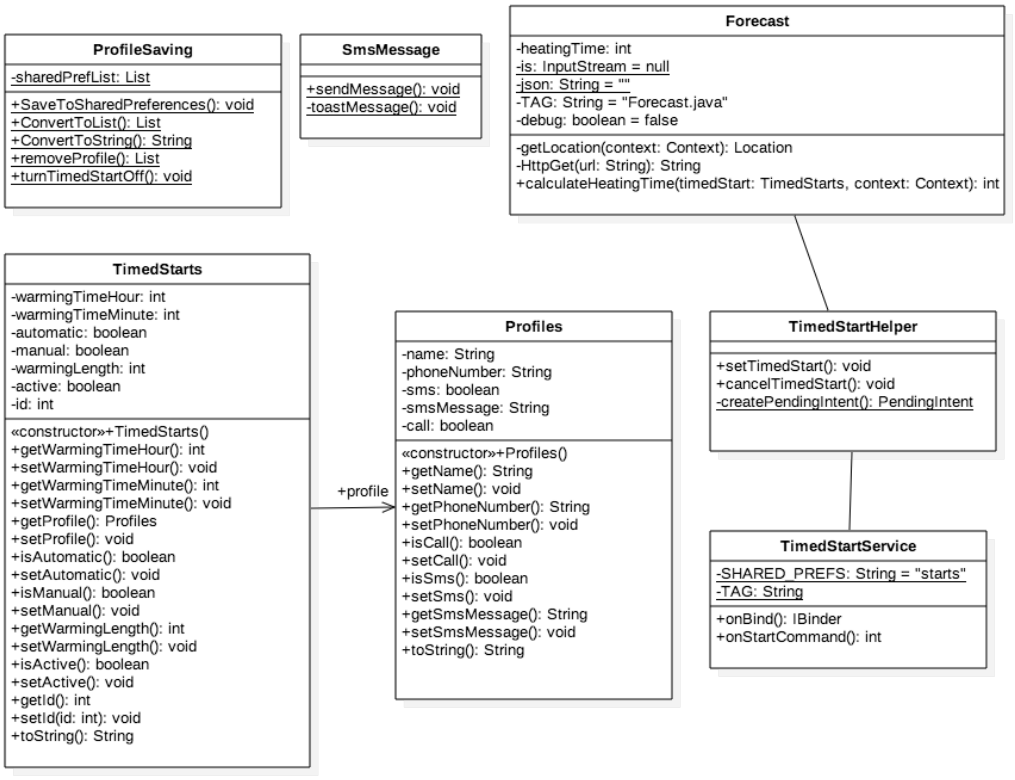
4 MÄÄRITTELY JA SUUNNITTELU

4.1 Sovelluksen rakenne

Sovellus toteutettiin käyttäen Model-View-Presenter (MVP) rakennetta (**Kuvio 5.**). Tämä poikkeaa perinteisemmästä MVC (Model-View-Controller) rakenteesta siten, että käyttöliittymää ohjaavat komponentit jakautuvat näkymää ohjaaviin (view), sekä näkymää varten dataa mallintaviin (presenter) paketteihin. Tämän lisäksi taustalla toimii model -paketin luokat, joiden tehtävänä on esimerkiksi datan käsitteleminen ja rajapintakutsut (**Kuvio 6.**).

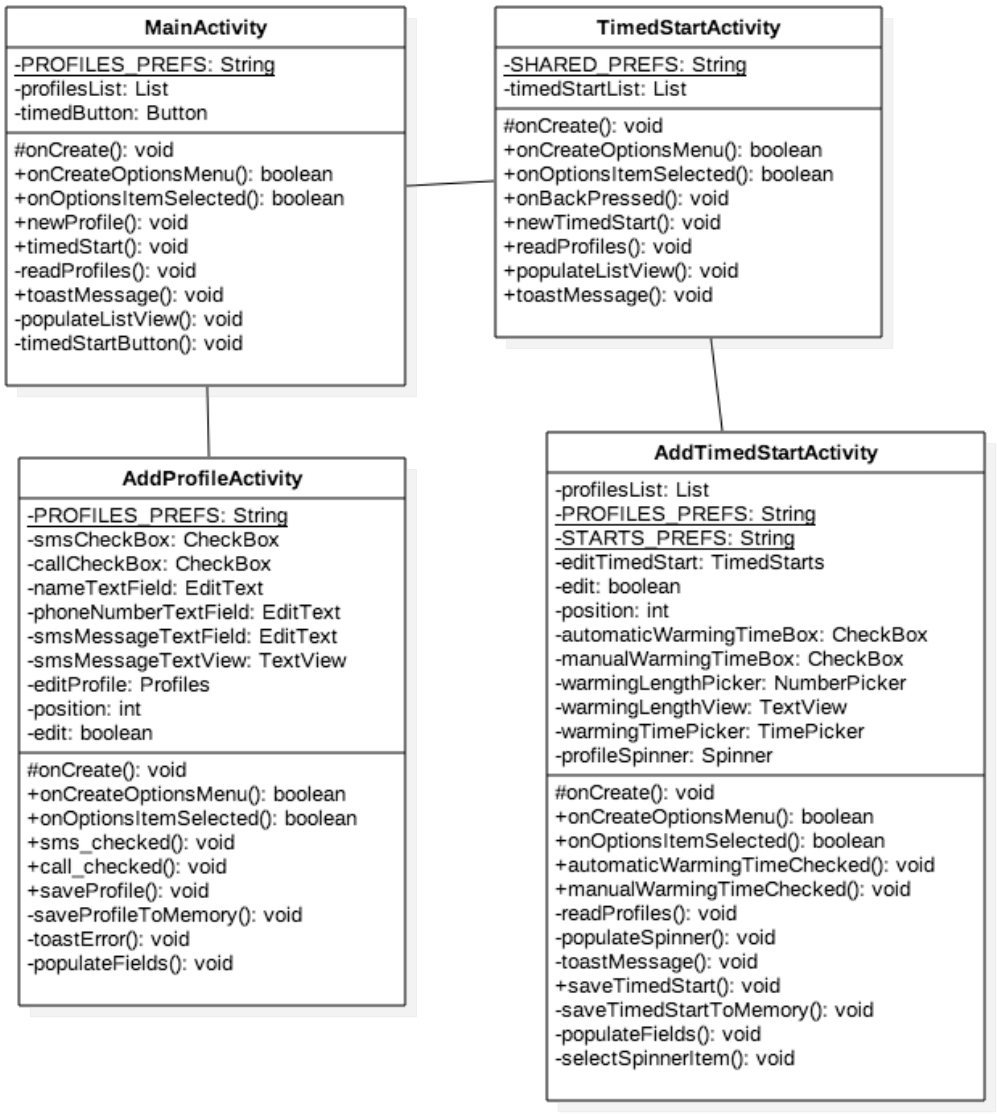


Kuvio 5. Sovelluksen pakkauskaavio.



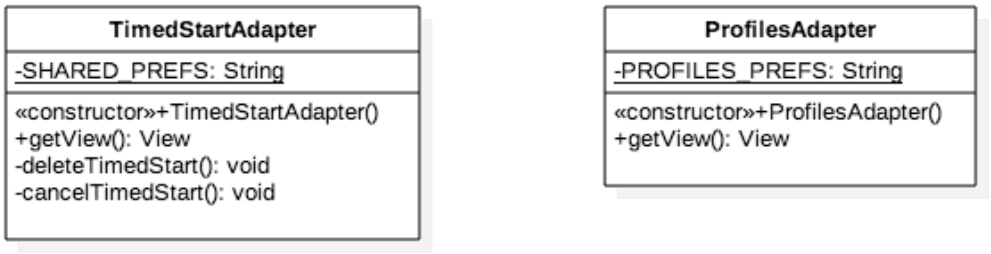
Kuvio 6. Model –paketin luokkakaavio.

Androidissa käyttöliittymä koostuu erilaisista komponenteista, jotka kommunikoivat epäsuorasti keskenään. View –ryhmään kuuluvat tässä sovelluksessa Activity –luokat (**Kuvio 7.**). Activityt ovat lähes aina käyttäjälle esitettäviä interaktiivisia näkymiä, joihin käyttöliittymä voidaan rakentaa.



Kuvio 7. View –paketin luokkakaavio.

Käyttöliittymän rakentaminen tehtiin xml –tiedostoilla, joissa määritellään kussakin activityssä näkyvät käyttöliittymän komponentit, sekä niiden suhteet keskenään. Tämän lisäksi käyttöliittymän rakentamiseen käytettiin presenter –ryhmään kuuluvia adaptereja (**Kuvio 8.**). Adapterit perustuvat yhden alkion kokoiseen xml –tiedostoon, jonka perusteella monistetaan käyttäjälle lista halutusta datajoukosta. Tässä ohjelmassa adaptereja käytettiin käyttäjän luomien profiilien, sekä ajastettujen käynnistysten esittämiseen. /2./



Kuvio 8. Presenter –paketin luokkakaavio.

Sovellus koostuu kolmestatoista luokasta. Activityjen ja adaptereiden lisäksi se sisältää luokkia datan muodostamista ja tallentamista varten. Tämän lisäksi siinä on luokka sääennusteen hakemiselle ulkopuoliselta rajapinnalta, ja luokat palvelujen (services) muodostamista sekä suorittamista varten. Androidin palvelut ovat ohjelmakomponentteja myöhemmäksi sijoittuvien toimintojen suorittamiseen taustalla.

4.2 Vaatimusmääritelmä

Ennen ohjelman toteuttamista sille määriteltiin vaatimukset, jotka sen tulisi toteuttaa ja jotka se voisi toteuttaa, mikäli aikataulu mahdollistaisi. Vaatimukset jaettiin kolmeen prioriteettiin: 1: vaatimukset jotka tulee olla, 2: vaatimukset jotka pitäisi olla sekä 3: vaatimukset jotka olisi hyvä lisä.

Näistä vaatimuksista opinnäytetyövaiheessa ehdittiin toteuttaa kaikki prioriteetin 1 ja 2 saaneet ominaisuudet sekä osa prioriteetin 3 vaatimuksista. Loput toteutetaan ajan myötä jatkokehitysvaiheessa.

Taulukko 1. Ohjelman vaatimusmääritelmä.

Vaatimus	**Prioriteetti**
Ohjelmalla pystyy käynnistämään lisälämmittimen käyttäen puhelua	1
Ohjelmalla pystyy käynnistämään lisälämmittimen käyttäen tekstiviestiä	2

Ohjelmalla voi käynnistää lisälämmittimen välittömästi painikkeella	1
Käyttäjältä varmistetaan halutaanko lisälämmitin varmasti sytyttää välittömästi	3
Ohjelmaan pystyy asettamaan herätyskellon tavoin ajankohdan sille koska auton on oltava lämmin	1
Ohjelma osaa laskea ajankohdan perusteella koska lämmitys pitää käynnistää	1
Ohjelma osaa laskea tarvittavan lämmitysajan käyttäen hyväksi ulkoilman lämpötilaa (sääennuste)	2
Ohjelmaan pystyy luomaan profiileja eri lisälämmittimiä varten	2
Ohjelmaan pystyy asettamaan kerralla monta ajankohtaa monelle eri profiilille	3
Ohjelma ilmoittaa käyttäjälle seuraavan kerran ohjelmaa käytettäessä milloin sytytys on suoritettu ja kuinka kauan autoa lämmitettiin	3
Ohjelma näyttää käyttäjälle, että sytytys on suoritettu	2

4.3 Toiminnallinen määrittely

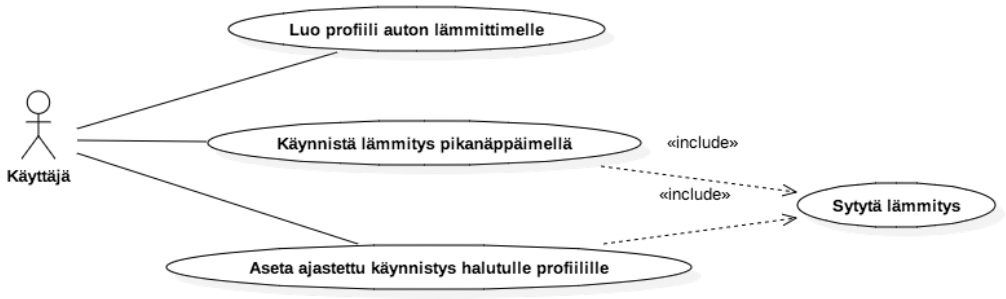
Käyttäjän tulee pystyä lisäämään, muokkaamaan ja poistamaan profiileja, joihin syötetään auton lisälämmittimen tiedot. Virheentarkistuksen jälkeen profiili kirjoitetaan laitteen muistiin. Sovelluksen aloitusnäkymä listaa käyttäjän luomat profiilit, ja näyttää jokaiselle profiilille pikakuvakkeen lämmityksen käynnistämistä varten.

Käyttäjä voi valita profiililleen joko puheluun tai tekstiviestiin perustuvan käynnistyksen, ja tietty profiili käyttää vain valittua metodia. Käyttäjä voi asettaa herätyskellotyylisesti ajastetun käynnistyksen lisälämmittimen sytytystä varten. Ajastetussa käynnistyksessä käyttäjä valitsee ajankohdan, jolloin haluaa autonsa olevan lämmin. Ajastetulle käynnistykselle valitaan haluttu profiili listalta.

Ajastetussa käynnistyksessä käyttäjä voi valita joko manuaalisesti asetettavan käynnistysajan tai sääennusteeseen perustuvan käynnistysajan, jonka ohjelma laskee automaattisesti. Virheentarkistuksen jälkeen ohjelma tallentaa ajastetun käynnistyksen laitteen muistiin sekä asettaa ajastetun toiminnon. Käyttäjän luomat ajastetut käynnistykset näytetään listalla, ja käyttäjä voi sammuttaa tai asettaa ne takaisin käyttöön.

Ajastetun käynnistyksen syttyessä laite suorittaa joko puhelinsoiton tai tekstiviestin lähettämisen luodun profiilin tietojen perusteella. Tämän jälkeen käyttäjälle näytetään ilmoitus suoritetusta toiminnosta. Ilmoituksesta käyttäjä pääsee takaisin sovellukseen, jossa ajastettu käynnistys näkyy nyt sammutettuna.

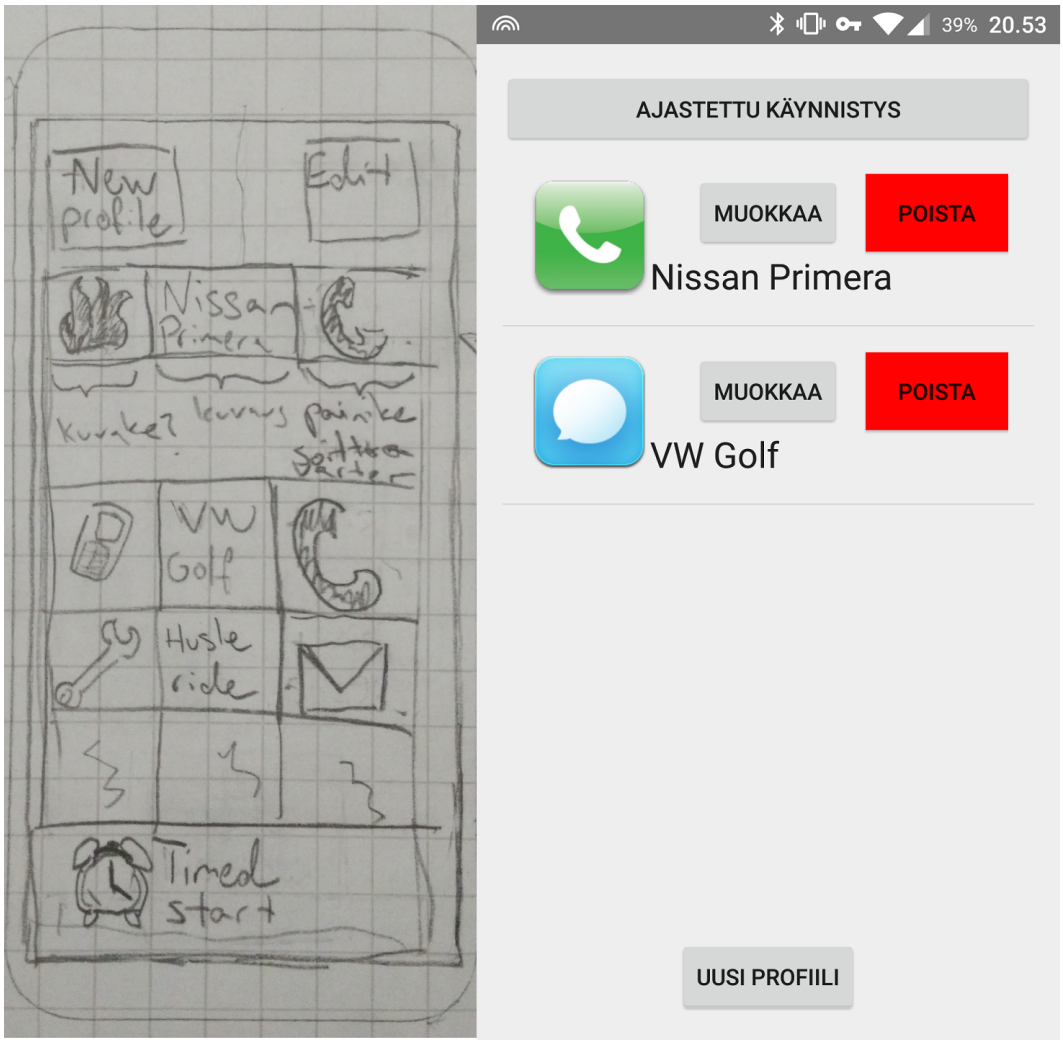
Käyttötapauskaaviolla pyritään kuvaamaan sovelluksen tärkeimmät toiminnot ja toimijat yksinkertaisesti. Tässä tapauksessa käyttötapauskaavioon on kuvattuna toimijaksi ohjelman käyttäjä, ja käyttötapauksiksi ohjelman tärkeimmät toiminnot (**Kuvio 9.**). Nämä ovat uuden profiilin luominen, lämmityksen käynnistäminen pikanäppäimellä, sekä ajastetun käynnistyksen luominen. Näistä kaksi viimeistä suorittavat lopulta saman toiminnon: sytyttävät lämmityksen.



Kuvio 9. Sovelluksen käyttötapauskaavio.

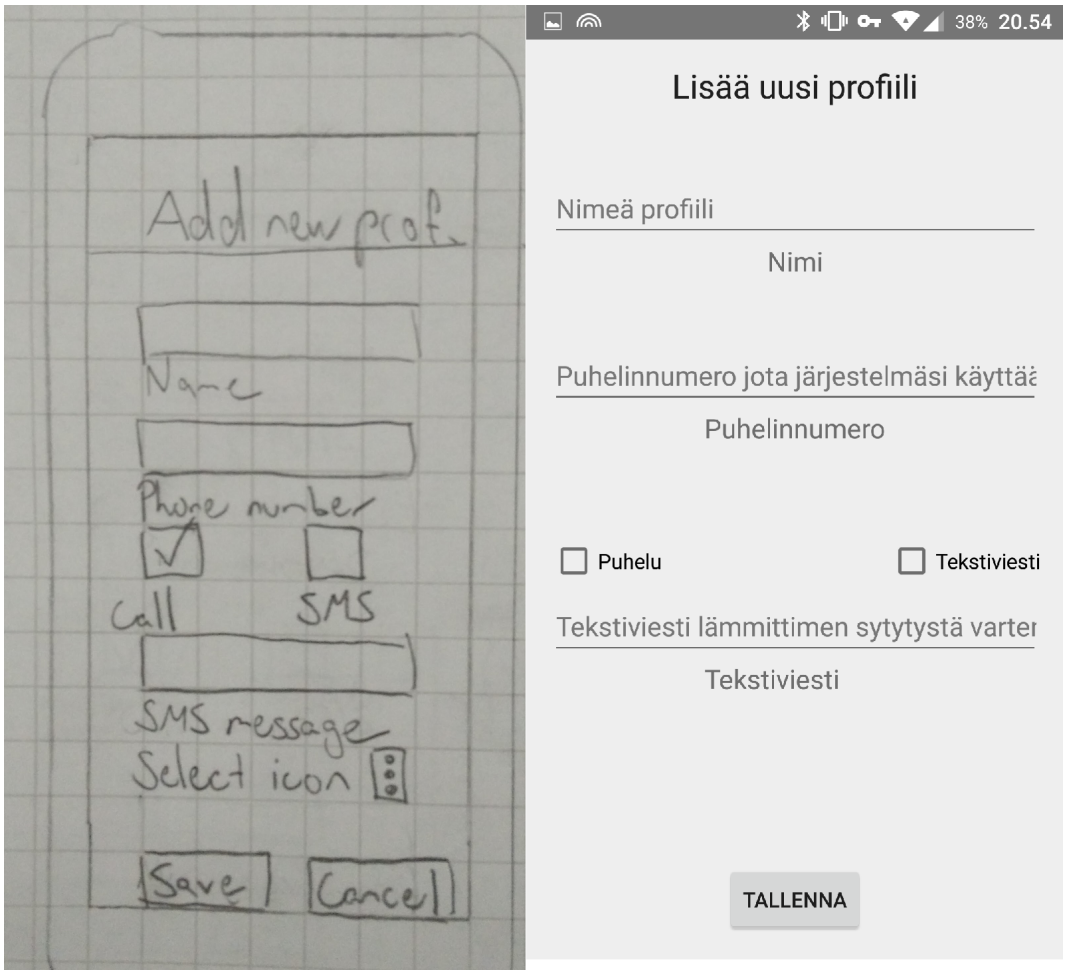
4.4 Käyttöliittymän suunnittelu

Yksi tärkeimmistä ohjelmalle suunnitteluvaiheessa asetetuista tavoitteista oli yksinkertainen ja helppokäyttöinen käyttöliittymä. Sovelluksen käyttöliittymää rakennettaessa pysyttiin uskollisena alkuperäisille mockupeille, ainoastaan joitakin kohtia yksinkertaistaen.



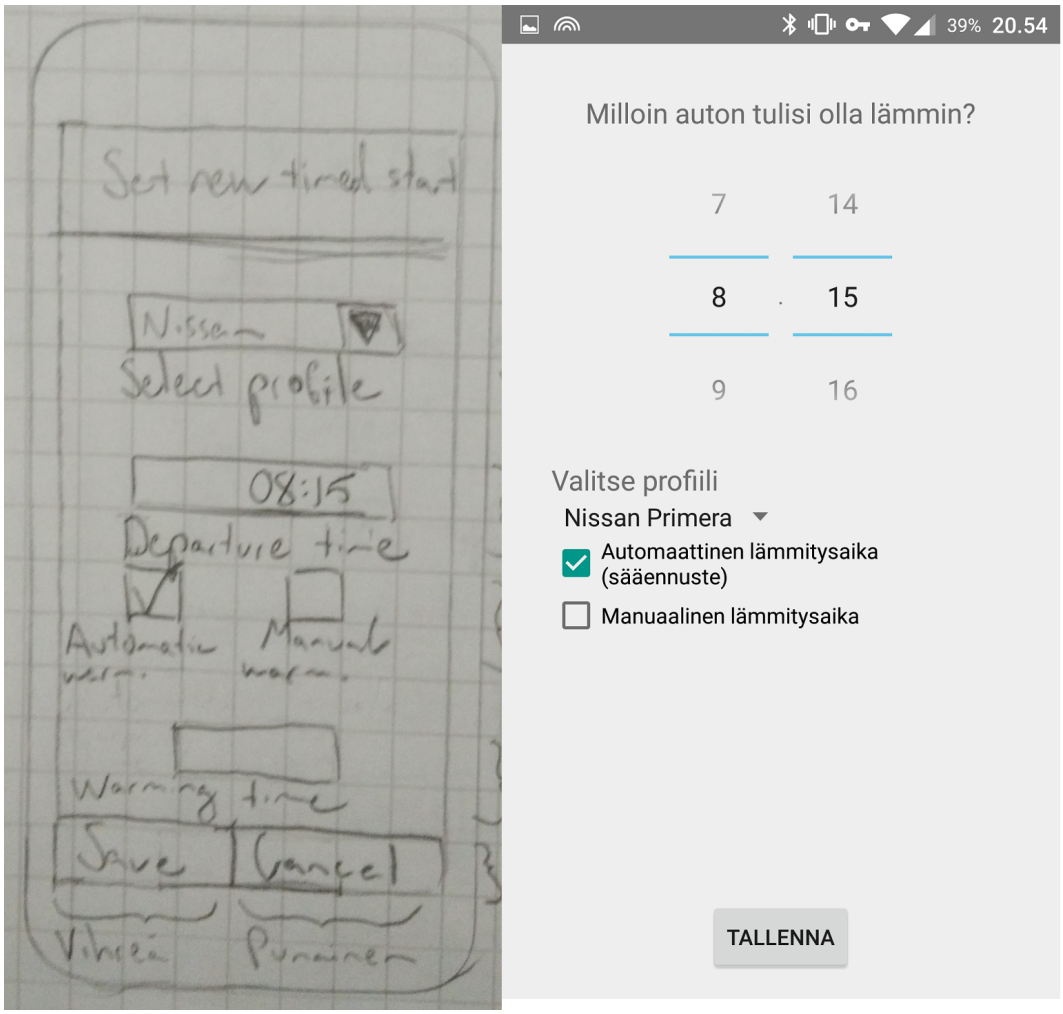
Kuvio 10. Aloitusnäkymä / profiilien listaus.

Kuvioissa 10-13 on esitetty rinnakkain alkuperäinen mockup, sekä julkaisuversion käyttöliittymä samasta näkymästä. Aloitusnäkymässä ohjelma listaa käyttäjälle tämän luomat profiilit, sekä näyttää painikkeet uuden profiilin luomiselle ja ajastettujen käynnistysten tarkkailulle (**Kuvio 10.**).



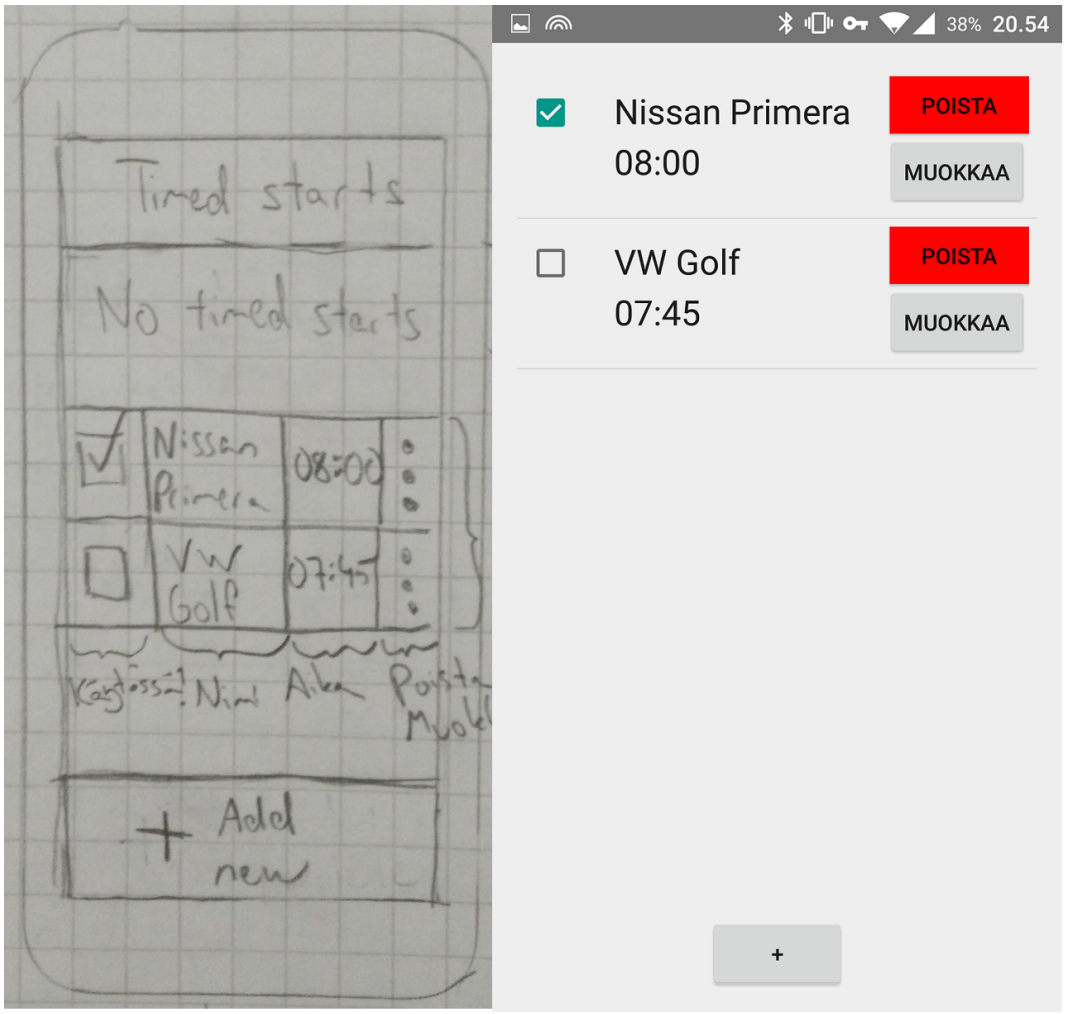
Kuvio 11. Uuden profiilin luominen.

Uuden profiilin luomisessa ohjelma näyttää kentät, joihin käyttäjän tulee kirjoittaa vaadittavat tiedot. Virheentarkistuksen jälkeen ohjelma luo uuden profiilin käyttäjän antamista tiedoista (**Kuvio 11.**). Uuden ajastetun käynnistyksen asettaminen toimii samalla tavalla. Siinä käyttäjä valitsee ajan mihin mennessä toivoo autonsa olevan lämmin, valitsee haluamansa profiilin, sekä joko asettaa lämmitysajan itse tai valitsee automaattisesti laskettavan lämmitysajan (**Kuvio 12.**).



Kuvio 12. Uuden ajastetun käynnistyksen luominen.

Ajastetun käynnistyksen asettamisen jälkeen ohjelma listaa käyttäjän luomat ajastukset, ja näyttää valintalaatikon aktiivisten ajastusten edessä. Tästä näkymästä käyttäjä pystyy myös muokkaamaan, poistamaan ja aktivoimaan ajastuksiaan (**Kuvio 13.**).



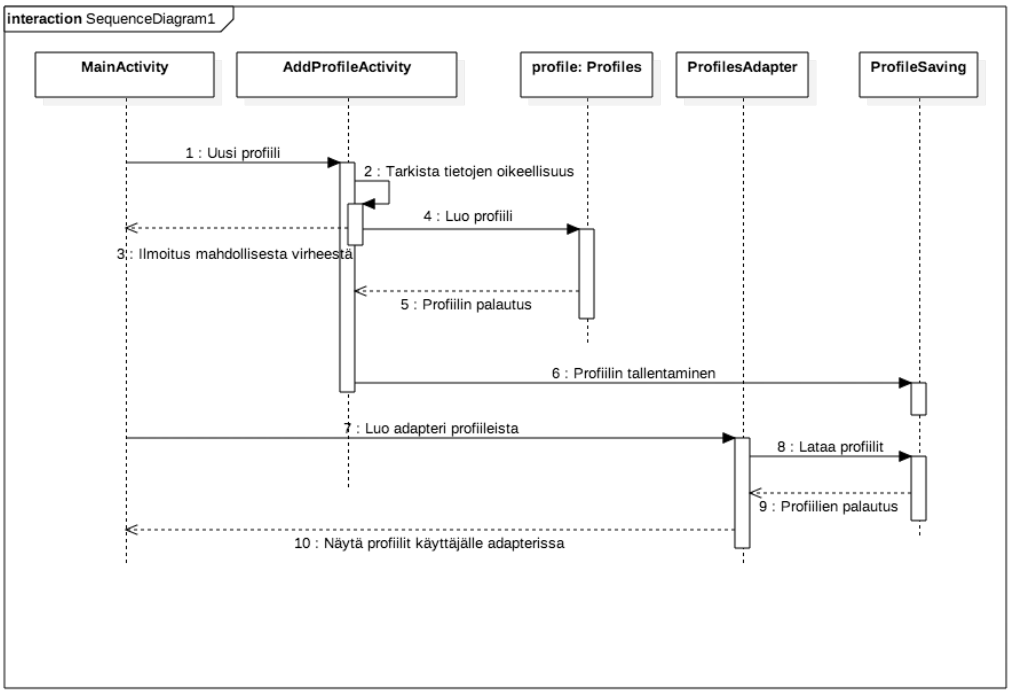
Kuvio 13. Käyttäjän luomat ajastetut käynnistykset.

4.5 Sekvenssikaaviot

Sekvenssikaavioiden tarkoitus on kuvastaa ohjelman toiminnallisuuksia, ja näyttää luokkien väliset suhteet tietyssä tapahtumaketjussa. Sekvenssikaavio havainnollistaa esimerkiksi mitä ohjelmassa tapahtuu käyttäjän suorittaman toiminnon jälkeen.

Käyttäjän luodessa uutta profiilia, vaihtaa ohjelma aloitusnäkymä MainActivityn AddNewProfileActivityyn. Täällä käyttäjä täyttää tarvittavat tiedot, ja tallentaa profiilin. Virheentarkistuksen jälkeen käyttäjän antamista tiedoista luodaan uusi

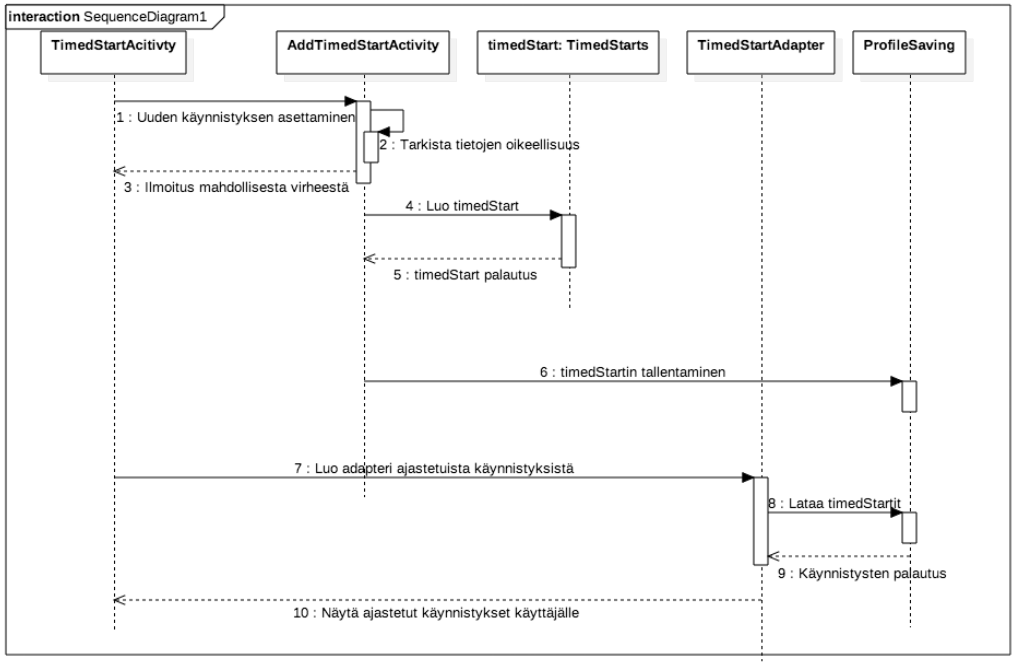
olio Profiles –luokan muodostimella. Uusi profiili välitetään ProfileSaving –luokalle, jossa se tallennetaan laitteen muistiin. Käyttäjä palautetaan MainActivityyn, ja siellä luodaan adapteri profiilien listaamista varten. Adapterille määritellään omassa luokassa siihen liitettävä data, sekä toiminnallisuudet jokaisen listan alkion painikkeita varten. Näin kaikkien profiilien painikkeet kohdistuvat automaattisesti oikealle oliolle (**Kuvio 14.**).



Kuvio 14. Sekvenssikaavio profiilin luomisesta ja profiilien esittämisestä.

Käyttäjän ollessa ajastetut käynnistykset listaavassa näkymässä on hänellä mahdollisuus luoda uusi ajastettu käynnistys. Tämä ohjaa käyttäjän TimedStartActivitystä AddTimedStartActivityyn. Täällä käyttäjältä kysytään haluttu ajankohta auton lämmityksen valmistumiselle, oikea profiili, sekä valinta automaattisen lämmitysajan tai käyttäjän itse asettaman lämmitysajan väliltä. Mikäli käyttäjä haluaa asettaa lämmitysajan itse, ilmestyy näkyviin valintarulla, josta tarvittava lämmitysaika voidaan valita. Ajastettua käynnistystä tallennettaessa ohjelma suorittaa ensin virheentarkistuksen, jonka jälkeen ajastetusta käynnistyksestä luodaan olio joka tallennetaan laitteen muistiin. Käyttäjä palautetaan TimedStartActivityyn

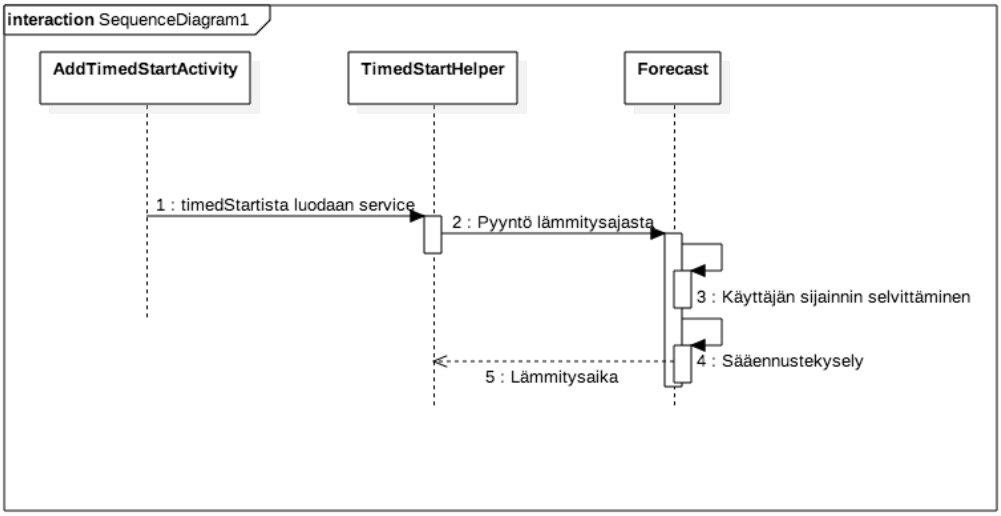
missä ohjelma luo adapterin laitteeseen tallennetuista ajastetuista käynnistyksistä, ja esittää ne käyttäjälle (**Kuvio 15.**).



Kuvio 15. Sekvenssikaavio ajastetun käynnistyksen luomisesta ja niiden esittämisestä.

Ajastettua käynnistystä luodessa siitä tehdään myös service, eli myöhemmin suoritettava Androidin komponentti, joka ei tarjoa käyttöliittymää. Servicen avulla lämmityksen käynnistys saadaan suoritettua myöhemmin ilman käyttäjältä vaadittavia toimia. Serviceä luodessa ohjelma tarkistaa haluaako käyttäjä automaattisesti laskettavan käynnistysajan, ja niin ollessa pyytää Forecast –luokkaa laskemaan tarvittavan käynnistysajan (**Kuvio 16.**). Forecast –luokassa muodostetaan API –kutsu OpenWeatherMapin palvelimelle, jonne välitetään käyttäjän sijainnin koordinaatit. Näiden perusteella OpenWeatherMap palauttaa muutaman päivän sääennusteen, josta ohjelma etsii oikeaa tuntia koskevan ennusteen. Algoritmi laskee sääennusteen perusteella tarvittavan lämmitysajan, ja palauttaa sen TimedStartHelper –luokalle. Tämä muodostaa oikeasta ajasta olion, joka Intent –olion kanssa kiinnitetään luotavalle servicelle. Intent on Androidissa abstrakti kuvaus suoritettavasta operaatiosta. Intentissä määritellään ajastuksen syttyessä suoritettava

luokka, sekä siihen kiinnitetään timedStart –olio joka sisältää herätyksen tiedot. /20; 9/



Kuvio 16. Sekvenssikaavio servicen luomisesta.

5 TOTEUTUS

Opinnäytetyöksi tehdyn sovelluksen kehittäminen eteni noudattaen vesiputous –kehitysmallia. Ominaisuuksia lisätessä kehitettiin aluksi dummy –käyttöliittymä, joka näyttää oikealta, mutta ei vielä käytännössä tee mitään. Tämän päälle rakennettiin ominaisuuksia, jotka sen jälkeen testattiin usealla eri AVD:llä, sekä fyysisillä Android –laitteilla.

5.1 MainActivity ja ProfilesAdapter

Ohjelmointiosuuden suorittaminen aloitettiin ohjelman ensimmäisestä käyttäjälle aukeavasta näkymästä, eli MainActivitystä. Android Studiolla uutta activityä luodessa IDE generoi sille valmiiksi joitakin metodeja. Näitä ovat esimerkiksi onCreate(), joka suoritetaan kun activity ladataan, sekä onOptionsItemSelected(), jossa määritellään sovelluksen mahdollisesti käyttämän yläpalkin toiminnot.

Activityn kehittäminen aloitettiin luomalla sille xml –tiedosto, jossa määritellään activityn näyttämät painikkeet, ja niiden sijainnit. Tässä vaiheessa kehitystä MainActivityä ei pystytty kehittämään pidemmälle, ja siirryttiin uuden profiilin luomisesta vastaavaan AddProfileActivity –luokkaan. Myöhemmin kuitenkin MainActivitylle luotiin ProfilesAdapter, joka vastaa käyttäjän luomien profiilien esittämisestä MainActivityssä.

Adapteria varten luotiin oma luokka nimeltä ProfilesAdapter, sekä xml –tiedosto, jossa käyttöliittymä suunniteltiin. ProfilesAdapter –luokassa xml –tiedostossa määriteltyihin painikkeisiin kiinnitettiin toiminnallisuus, joka käsittelee MainActivitystä välitettyä tietojoukkoa, eli tässä tapauksessa käyttäjän luomia profiileja. Näin yhdellä onClickListener –määrittelyllä saadaan ohjattua jokaisen profiilin painikkeita (**Kuvio 17.**).

```
//      onClick for call or sms icon
        callImageView.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {

//              this performs phone call
                if (profile.isCall()) {
                    Intent phoneIntent = new Intent(Intent.ACTION_CALL);
                    phoneIntent.setData(Uri.parse("tel:" + profile.getPhoneNumber() + ""));
                    getContext().startActivity(phoneIntent);

//                  send SMS here
                } if (profile.isSms()) {
//                  toast user about attempt to send SMS
                    MainActivity mainActivity = new MainActivity();
                    Context context = getContext();
                    mainActivity.toastMessage(context.getString(R.string.sending_sms), context);
//                  Send SMS to profiles number with profiles message
                    SmsMessage.sendMessage(getContext(), profile.getPhoneNumber(), profile.getSmsMessage());
                }
            }
        });
```

Kuvio 17. Profiilin pikakäynnistyspainikkeen onClickListener.

MainActivityssä ProfilesAdapterista luotiin olio, ja sille välitettiin lista käyttäjän profiileista. Tämän jälkeen adapteri kiinnitettiin MainActivityn käyttöliittymän listanäkymään profilesListView (**Kuvio 18.**).

```
    private void populateListView() {
//      here we bind our custom adapter to listView
//      in this method we will populate listView with profiles of profilesList

        ListAdapter profilesAdapter = new ProfilesAdapter(this, profilesList);
        ListView profilesListView = (ListView) findViewById(R.id.profilesListView);
        profilesListView.setAdapter(profilesAdapter);
    }
```

Kuvio 18. Profiilien adapterin määritteleminen ja datajoukon kiinnittäminen siihen.

5.2 AddProfileActivity ja ProfileSaving

AddProfileActivity on näkymä, jonne käyttäjä ohjataan tämän luodessa uutta profiilia lämmityslaitteelleen. Activitylle luotiin .xml –tiedosto, jossa määriteltiin sen käyttöliittymä, ja tämän jälkeen kirjoitettiin virheentarkistus käyttäjän syöttämien tietojen tarkistamista varten.

Näistä tiedoista luodaan Profiles –luokassa määritelty olio, ja se tallennetaan laitteen muistiin käyttäen ProfileSaving –luokkaa. Luotu olio, tieto datan laadusta, sekä tallentamisessa tarvittavia muuttujia välitetään ProfilesSaving –luokalle (**Kuvio 19.**).

ProfileSaving -luokassa olio muutetaan JSON –dataksi käyttäen Googlen GSON –kirjastoa. Luotu olio lisätään puhelimen SharedPreferences tietueeseen siellä jo olevan datan lisäksi. Vanha profiili poistetaan, mikäli nyt tallennettu olio oli muokattu versio jo aiemmin tallennetusta profiilista.

```
//        here we save new profile to shared preferences
    private void saveProfileToMemory(Profiles profile) {
        ProfileSaving.SaveToSharedPreferences(this, PROFILES_PREFS, profile, position, edit);

//        finally return to mainActivity
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
```

Kuvio 19. Luodun olion välittäminen ProfileSaving –luokalle.

5.3 TimedStartHelper ja Forecast

Käyttäjän luomat ajastetut käynnistykset tallennetaan laitteen muistiin uudelleenkäyttämistä varten samalla tavalla kuin profiilit ProfileSaving –luokan avulla. Tämän lisäksi oli määriteltävä käyttäjän luoman ajastuksen perusteella service, jolla haluttuja toimintoja voidaan suorittaa pitkienkin aikojen päästä. Tämä määrittely tapahtuu TimedStartHelper –luokassa, joka suoritetaan aina kun uusi ajastettu käynnistys on luotu, tai kun vanha ajastettu käynnistys aktivoidaan uudelleen (**Kuvio 20.**).

```
//                  timedStart goes on here!
                else {
                    timedStart.setActive(true);

//                    informs users about success with toast message
                    tSActivity.toastMessage("Timed start enabled!", context);
//                    set service for our timedStart
                    TimedStartHelper tSHelper = new TimedStartHelper();
                    tSHelper.setTimedStart(context, timedStart);
                }
```

Kuvio 20. timedStart –olion aktivoiminen.

TimedStartHelper –luokalle välitetään käsiteltävä timedStart –olio, sekä sovelluksen tilatiedot context –muuttujassa. Käyttäjän luodessa uutta ajastettua käynnistystä hän valitsee manuaalisesti asetettavan käynnistysajan, ja automaattisesti laskettavan käynnistysajan väliltä. Mikäli timedStart –olio sisältää automaattisen

lämmitysajan, suorittaa ohjelma pyynnön tarkan lämmitysajan laskemisesta Forecast –luokalta.

Context ja timedStart välitetään edelleen Forecast –luokalle, missä ohjelma suorittaa API –kutsun OpenWeatherMapin palvelimelle. Tässä HTTP –kutsussa välitetään käyttäjän laitteen koordinaatit, ja niiden perusteella OpenWeatherMap palauttaa sijainnin sääennusteen tuleville päiville. Ohjelma parsii tuosta datasta tarvittavan ajankohdan ennusteen, ja laskee sen perusteella lämmittimen tarvitseman ajan.

Tuo tieto palautetaan TimedStartHelper –luokalle, ja sen perusteella muodostetaan kalenteri –olio servicen suoritusaikaa varten. Luokassa määritellään intent, jolle kerrotaan servicen suorittama luokka. Tämän lisäksi siihen kiinnitetään ohjelman tila contextin avulla, ja sille lisätään dataksi kyseessä oleva timedStart –olio (**Kuvio 21.**). Tämän intentin, sekä kalenteri –olion avulla määritellään service, joka tallennetaan puhelimeen myöhempää suorittamista varten.

```
//    Crete pendingIntent to be used with alarm
    private static PendingIntent createPendingIntent(Context context, TimedStarts timedStart) {

//        We pass timedStart object as extra into our intent
        Intent intent = new Intent(context, TimedStartService.class);
        intent.putExtra("timedStart", timedStart);

        return PendingIntent.getService(context, timedStart.getId(), intent, PendingIntent.FLAG_UPDATE_CURRENT);
    }
```

Kuvio 21. Servicen käyttämän intentin luominen.

5.4 Lokalisointi

Kehityksen alkuvaiheessa ohjelmasta tehtiin englanninkielistä, mutta ennen julkaisuversiota se käännettiin myös suomeksi. Androidilla kielituen lisääminen ohjelmaan on tehty hyvin helpoksi, mikäli kehittäjä on noudattanut Googlen ohjeita tekstimuuttujien esittelemisessä. Tekstimuuttujat tulisi esitellä erillisessä strings.xml –tiedostossa. Lähdekoodissa niihin saadaan viitattua niiden kokonaislukuarvoa käyttäen (**Kuvio 22.**).

```
//        We deactivate our timedStart we want to cancel and inform user
    private void cancelTimedStart(TimedStarts timedStart, Context context) {
        TimedStartActivity tSActivity = new TimedStartActivity();
//          Inform user about canceling
        tSActivity.toastMessage(context.getString(R.string.disabled_message), context);
        TimedStartHelper tsHelper = new TimedStartHelper();
//          Cancel timedStart
        tsHelper.cancelTimedStart(context, timedStart);
    }
```

Kuvio 22. Tekstimuuttujan noutaminen strings.xml –tiedostosta käynnistystä peruttaessa

Tämän jälkeen ohjelma saadaan käännettyä uudelle kielelle luomalla uusi strings.xml –tiedosto. Tiedostolle asetetaan kieli, ja kehitysympäristön kääntämisnäkymällä jokaiselle ohjelmassa esitellylle tekstimuuttujalle saadaan nopeasti käännettyä arvo toiselle kielelle (**Kuvio 23.**).

Key	Default Value	Untra...	Finnish (fi)
action_settings	Settings		Asetukset
add_new_timed_start	"+"		"+"
app_name	Smart Car Heater		Smart Car Heater
automatic_warming	Automatic warming time (forecast)		Automaattinen lämmitysaika (sääennuste)
call_check_box	"Call"		Puhelu
call_or_sms_error	Select call or SMS		Valitse puhelu tai tekstiviesti

Kuvio 23. Näkymä Android Studion kääntämiseditorista.

Ohjelmaa ajettaessa Android suorittaa tarkistuksen, jossa se vertaa käyttöjärjestelmän valittua kieltä ohjelman tarjoamiin käännöksiin. Mikäli ohjelma on käännetty samalle kielelle millä Androidia käytetään, se valitaan käyttöön. Muussa tapauksessa ohjelma käyttää oletuskieltä, eli tässä tapauksessa englantia. /21/

6 TESTAUS

6.1 Yleistä

Sovelluksen testaamista suoritettiin jatkuvasti toteutuksen ohella. Uudet ominaisuudet testattiin välittömästi, että niissä esiintyvät mahdolliset virheet eivät vaikuttaisi syvemmälle ohjelman toimintaan. Työn suunnitteluvaiheessa kirjoitettiin ylös tulevia testitapauksia, ja niitä lisättiin myöhemmin ohjelman rakenteen selkiytyessä.

Testaamisessa käytettiin Android Studiosta ajettavia AVD –ympäristöjä, sekä fyysisiä Android –laitteita. Ohjelmaa testattiin virtuaalisessa ympäristössä järjestelmillä Android 4.0.3 matalaresoluutioinen 4” laite, Android 5.0 korkearesoluutioinen 5” laite, sekä Android 5.0 korkearesoluutioinen 9” laite. Fyysisinä laitteina testeissä toimivat Android 5.1 5,5” laite, sekä Android 4.4 4” laite. Näin varmistuttiin ohjelman toiminnallisuuksien yhtenäisyydestä eri sukupolvien, ja hintaluokkien laitteiden välillä.

6.2 Testitapaukset

Testitapauksissa suunniteltiin tapaukset, ja niiden toivotut tulokset. Näin testaamiseen kuluva aika saatiin minimoitua, ja testit suoritettua sulavasti toteutuksen ohella.

Taulukko 2. Testitapaukset.

Testitapaus	Toivottu tulos	Tulos
Profiilin luominen	Käyttäjän luoma profiili tarkistetaan virheiden varalta, tallennetaan Androidin shared preferencesiin ja näytetään MainActivityn listalla	OK
Profiiliadapteri	Profiiliadapteri luo jokaiselle listalla näytettävälle profiilille painikkeet poistoa, muokkausta sekä pikakäynnistystä varten, ja painikkeet toimivat	OK

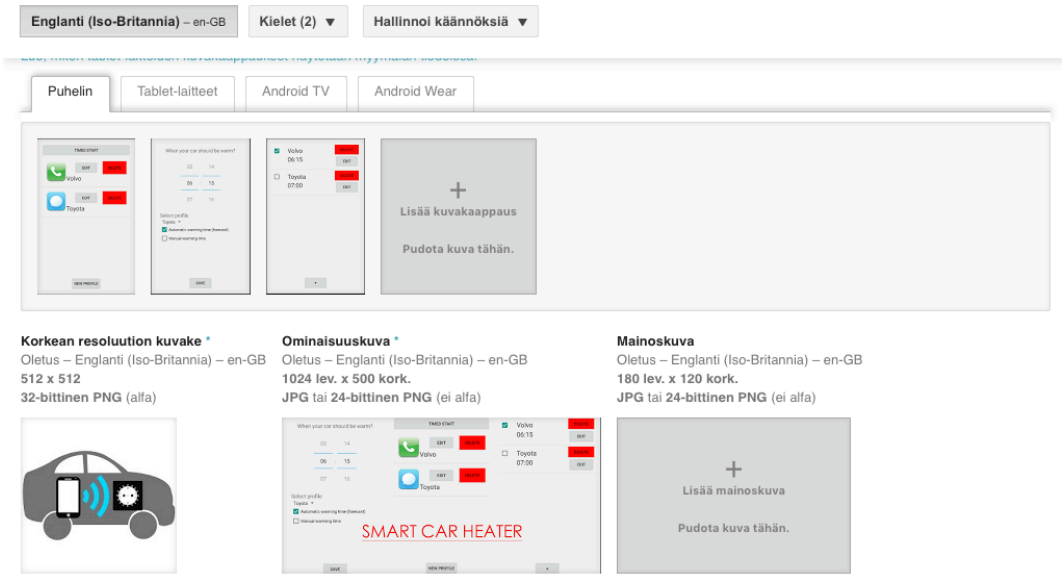
	kyseiselle profiilille	
Lämmityksen käynnistäminen	Sovellus suorittaa halutun toiminnon (puhelu tai SMS) oikeaan numeroon perustuen profiiliin tallennettuihin tietoihin	OK
Ajastetun käynnistyksen luonti	Käyttäjä voi asettaa ajastetun käynnistyksen valitsemalleen profiilille. Tarkistetaan virheiden varalta ja tallennetaan Androidin shared preferencesiin	OK
Ajastettujen käynnistysten adapteri	Käyttäjän luomat ajastetut käynnistykset näytetään listassa. Jokaista ajastettua käynnistystä varten luodaan painikkeet poistoa, ja muokkausta varten, sekä valintalaatikko ajastetun sytytyksen käynnistystä varten. Painikkeet käsittelevät vain haluttua profiilia.	OK
Ajastettu käynnistys	Käyttäjä asettaa ajastetun käynnistyksen päälle valintalaatikkoa käyttäen. Oikealla ajankohdalla puhelin suorittaa itsestään joko puhelun tai tekstiviestiherätyksen profiilin tietojen mukaisesti.	OK
Säätietoihin perustuva käynnistysaika	Käyttäjä voi valita lämmitysajan sijaan sääennusteeseen perustuvan automaattisen ajan. Puhelin suorittaa ennen käynnistystä API –kyselyn, laskee ideaalisen lämmitysajan ja suorittaa oikealla hetkellä käynnistyksen.	OK
Käyttäjän sijainnin tarkka selvittäminen	Käyttäjän sijainti selvitetään viimeisimpien sijaintitietojen perusteella. Mikäli näitä tietoja ei ole saatavissa kysellään GPS yhteydellä uusi sijainti.	KESKEN

Sovelluksen tärkeimmät ominaisuudet saatiin testattua onnistuneesti. Ainoastaan käyttäjän sijainnin selvittäminen kaikissa mahdollisissa tilanteissa ei tällä hetkellä toimi. Tällä hetkellä sovellus hakee laitteesta käyttäjän viimeisimmän tallennetun sijainnin. Tämän sovelluksen tarkoituksiin tämä sijainti on yleensä tarpeeksi tarkka. Ongelmaksi tämä muodostuu, jos käyttäjä on esimerkiksi juuri käynnistänyt laitteensa uudestaan. Tällöin sijaintitietoja ei välttämättä ole laitteen muistissa, ja ohjelma joutuu virhetilaan. Tällä toiminnolla on jatkokehityksessä yksi korkeimmista prioriteeteista.

7 KÄYTTÖÖNOTTO

Työn toteutusvaiheen jälkeen sovellus saatiin julkaisukelpoiseksi. Android ympäristössä sovellusten yleisin julkaisualusta on Googlen oma Play –sovelluskauppa. Sovelluskehittäjät voivat rekisteröityä Googlen virallisiksi kehittäjiksi, ja pienen maksun jälkeen omia sovelluksia voi julkaista Google Play –kaupassa. Myös tässä työssä kehitetty sovellus Smart Car Heater päätettiin julkaista Play –sovelluskaupassa.

Itse julkaisuprosessi on hyvin suoraviivainen. Sovellus käännettiin kehitysympäristössä apk –paketiksi, joka allekirjoitettiin käyttämällä omaa salausavainta. Googlen developer consolessa valittiin uuden sovelluksen julkaiseminen. Sovellus nimettiin, sille kirjoitettiin kuvaus, ja sille annettiin grafiikkasisältöä sovelluskaupassa näytettäväksi. Grafiikkasisältöön lukeutuu sovelluksen kuvake, kuvakaappauksia, kuvakaappaukset mahdollisista käännöksistä sekä ominaisuuskuvake, joka näytetään Google Playssä kyseisen sovelluksen mainosbannerina (**Kuvio 24.**).



Kuvio 24. Sovelluksen julkaiseminen Google Play –kaupassa

Google Play tarjoaa mahdollisuuden julkaista sovellus myös alfa- tai beeta –tasoisena. Työn sovellus julkaistiin valmiina 1.0 versiona. Sovellukselle suoritettiin kysely mahdollisia sisällön ikärajoitteita koskien. Tämän kyselyn perusteella Google Play asettaa sovellukselle suositellut ikärajoitukset. Julkaisun yhteydessä on myös mahdollista valita maat, joissa sovellus julkaistaan. Smart Car Heater päädyttiin julkaisemaan maailmanlaajuisesti. Tämän jälkeen Google Play käsitteli sovellusta muutaman tunnin ajan ennen kuin se julkaistiin virallisesti.

Google Playn valitseminen julkaisualustaksi oli helppo päätös. Androidille tarjolla olevista sovelluskaupoista se on kaikkein suosituin. Lisäksi tulevien päivitysten julkaiseminen sen kautta tulee olemaan vaivatonta. Sovelluksesta lisätään uusi apk –tiedosto, ja sen ominaisuudet kerrotaan. Tämän jälkeen uusi ohjelmaversio näkyy sovelluksen käyttäjille saatavilla olevana päivityksenä. /22/

8 JATKOKEHITYS

Vaikka julkaistu versio sovelluksesta täytti sille tässä vaiheessa asetetut vaatimukset, se on kuitenkin vielä paikoin keskeneräinen. Sovellusta on tarkoitus jatkokehittää ajan saatossa, ja sille suunniteltiin ominaisuuksia sekä parannuksia, jotka tullaan lisäämään tulevissa ohjelman versioissa.

8.1 Käyttöliittymän parannukset

Julkaisuversion käyttöliittymä täytti sille asetetut vaatimukset ollen yksinkertainen ja selkeä. Se jäi kuitenkin hyvin pelkistetyn näköiseksi vielä tässä vaiheessa. Täysin tyylittelemätön käyttöliittymä saattaa antaa helposti sovelluksesta amatöörimäisen ensivaikutelman, ja se voi karkottaa käyttäjiä ennen kuin he ehtivät tutustumaan sovelluksen tarjoamiin ominaisuuksiin.

Käyttöliittymän ehostukset ovat korkealla prioriteetilla sovelluksen seuraavan version kehittämisessä. Käyttöliittymän asettelun ehostaminen vaatii tutkimista ja suunnittelemista, mutta graafinen ulkonäkö vaatii varmasti parantamista. Jo pelkällä taustakuvalla täysin valkoisen staattisen taustan sijaan saavutettaisiin paljon ammattimaisempi kuva.

Lisäksi sovelluksen kuvake jäi hyvin pienelle huomiolle kehityksen tässä vaiheessa. Vaikka kuvake ei välttämättä vaikuta sovelluksen latausmääriin tai sen saamiin arvosteluihin, on se kuitenkin oleellinen osa sovelluksen yleisilmettä. Hyvin suunniteltu kuvake erottuu joukosta ja herättää positiivisia mielikuvia. Tämä takaa sen, että sovellus käynnistetään toisenkin kerran ensilatauksen jälkeen.

8.2 Käyttäjän sijainnin tarkka kyseleminen

Tällä hetkellä ohjelma lukee laitteen sijainnin käyttäen Androidin getLastKnownLocation –metodia. Tällä menetelmällä sijainti saadaan kyseltyä nopeasti eikä virtaa kuluttavaa GPS –yhteyttä tarvitse käyttää. Tämä menetelmä ei kuitenkaan ole tarkka, sillä käyttäjä on saattanut liikkua pitkänkin matkan viimeisen sijainnin selvittämisen jälkeen. Tämän lisäksi laite ei välttämättä muista viimeisintä sijaintitietoa, esimerkiksi uudelleenkäynnistyksen jälkeen. Tällainen tilanne aiheuttaa

ohjelmassa virhetilan, ja se on ehdottomasti korjattava seuraavaan julkaisuversioon.

8.3 Laitteen yhteystietojen käyttäminen

Uutta profiilia luodessa käyttäjä syöttää sovellukseen lämmityslaitteen tiedot. Lopputestauksessa havaittiin oikean puhelinnumeron löytämisen vaikeus. Oikea puhelinnumero on luultavasti käyttäjällä tallennettuna laitteen muistiin, ja sitä syöttäessä käyttäjä joutuu vaihtelemaan sovellusta tämän ohjelman sekä puhelinluettelon välillä.

Android tarjoaa kirjaston laitteen yhteystietojen käyttämiseen, ja oikea numero olisi näin mahdollista etsiä puhelinluettelosta poistumatta tästä sovelluksesta. Kärsimättömälle käyttäjälle tällainen puute saattaa olla ylitsepääsemätön, ja voi päättyä sovelluksen poistamiseen vaikeakäyttöisenä.

8.4 Webaston Thermo Call -sovelluksen viestikomennot

Webaston Thermo Call on autoon asennettava laitteisto Webaston ohjaamista varten. Laitteeseen asennetaan SIM –kortti, ja tämän jälkeen se on ohjattavissa GSM –verkon kautta matkapuhelimella. Webaston Thermo Call tarjoaa paljon lisätoimintoja pelkän etäkäynnistämisen lisäksi. Näitä toimintoja käytetään tekstiviestinä lähetettävillä komennoilla joihin laitteisto vastaa, mikäli hyväksyy ne.

Tällaisia komentoja ovat esimerkiksi ”TEMPSTATUS”, jolla laitteisto saadaan palauttamaan mm. sen hetkisen lämpötilan, ja ”1234NBANK:XXXX”, jolla saadaan lisättyä laitteen hyväksymiä puhelinnumeroita. Tällaisten komentojen muistaminen on hankalaa, ja suurella osalla käyttäjistä ei ole kiinnostusta perehtyä niihin. /14/

Nämä komennot voidaan lisätä sovellukseen valmiiksi lyhyen selityksen kera. Käyttäjä saisi valita näiden valmiiden komentojen väliltä, mikäli hän on Thermo Call laitetta käyttävä. Näin Thermo Callin tarjoamat ominaisuudet eivät menisi niin helposti hukkaan, jos käyttäjä päätyy ohjaamaan sitä tässä opinnäytetyössä tehdyllä sovelluksella. Suunnittelutyöhön kuuluneen kilpailijoiden tuotteisiin pe-

rehtymisen jälkeen myös todettiin, että Webaston oma ohjelma on erittäin vaikeakäyttöinen ja sekava. Olisi järkevää tarjota kallista Thermo Call –laitetta käyttäville vaihtoehtoinen sovellus, jolla Webaston oma sovellus voitaisiin korvata.

8.5 Kaatumisilmoitusten kerääminen

On mahdollista, että sovellukseen on jäänyt ohjelmointivirheitä huolellisesta testaamisesta huolimatta. Myös jatkokehityksessä lisättävät ominaisuudet saattavat aiheuttaa arvaamattomia ongelmia.

Tähän ongelmaan voidaan jatkokehityksen myötä vastata implementoimalla ACRA –kirjasto. ACRA –kirjastolla Android –laite saadaan raportoimaan sovelluksen kaatumisilmoitus GoogleDoc –muodossa halutulle palvelimelle. Tästä tiedostosta saadaan kerättyä arvokasta tietoa siitä, mikä sovelluksen kaatumisen on aiheuttanut. Tämä helpottaa ohjelman kehittämistä stabiiliksi, ja ammattimaiseksi kokonaisuudeksi.

9 YHTEENVETO

Työn suorittaminen oli hyvin opettavainen kokemus, ja sen myötä taidot sovelluskehittämisen parissa kasvoivat merkittävästi. Applikaatiota kehitettäessä törmättiin useisiin ennestään tuntemattomiin ongelmiin. Näiden selättäminen kehitti ongelmanratkaisukykyä, sekä taitoja itsenäiseen tiedonhankintaan. Työn suorittamisen myötä ohjelmistokehityksen eri työvaiheiden merkitys selveni käytännössä entisestään.

Tässä opinnäytetyössä saatiin kehitettyä julkaisukelpoinen sovellus Android –käyttöjärjestelmälle. Sovellus on tarkoitettu autojen lisälämmittimien etäkäyttöön, ja se käyttää kommunikointiin GSM –verkkoa. Sovelluksen ominaisuudet hyödyntävät useita erilaisia tekniikoita, kuten laitteen sijaintitietoja, tekstiviestien lähettämistä ja puheluiden soittamista.

Sovellus toteutti sille asetetut tärkeimmät vaatimukset, ja työ saatiin suoritettua aikataulussa. Sovellus saatiin julkaistua, vaikka siihen jäikin vielä jatkokehittämisen varaa. Tulevat ominaisuudet ja parannukset ovat kuitenkin jo pitkälle suunniteltuina, ja tämän myötä niiden implementoiminen tulee helpottumaan merkittävästi.

LÄHTEET

/1/ Acra. GitHub. Viitattu 7.11.2015. https://github.com/ACRA/acra

/2/ Activity. Android. Viitattu 15.10.2015. http://developer.android.com/reference/android/app/Activity.html

/3/ Android. Android. Viitattu 2.10.2015. https://www.android.com/

/4/ Android Architecture. EasyTutz. Viitattu 5.10.2015. http://www.eazytutz.com/android/android-architecture/

/5/ Android – Architecture. Tutorialspoint. Viitattu 5.10.2015. http://www.tutorialspoint.com/android/android_architecture.htm

/6/ Android Studio Overview. Android. Viitattu 3.10.2015. http://developer.android.com/tools/studio/index.html

/7/ Android 5.0, Lollipop. Android. Viitattu 3.10.2015. http://www.android.com/versions/lollipop-5-0/ - features

/8/ google-gson. GitHub. Viitattu 8.10.2015. https://github.com/google/gson

/9/ Intent. Android. Viitattu 24.10.2015. http://developer.android.com/reference/android/content/Intent.html

/10/ Introducing JSON. JSON. Viitattu 8.10.2015. http://www.json.org/

/11/ Java 20 years. Oracle. Viitattu 6.10.2015. http://oracle.com.edgesuite.net/timeline/java/

/12/ Joda-Time. Joda-Time. Viitattu 11.10.2015. http://www.joda.org/joda-time/

/13/ Just-in-time compiler (JIT). WhatIs.com. Viitattu 6.10.2015. http://whatis.techtarget.com/definition/just-in-time-compiler-JIT

/14/ Käyttöohje. Thermo Call TC3. Webasto. Viitattu 7.11.2015. http://www.webasto.com/fileadmin/webasto_files/documents/country-folder/finland/car/technical-documentation/car-technical-documentation-thermo-call_fi.pdf

/15/ Lisävarusteiden hinnasto. Eberspächer. Viitattu 1.10.2015. http://www.eeperi.com/eberspacher_lisavarusteet.html

/16/ Parviainen, H. TM Vertailu: Polttoainekäyttöiset lämmittimet. Tekniikan maailma. Tiivistelmä TM:n 17/07 artikkelista, s. 134-138. Viitattu 1.10.2015. http://tekniikanmaailma.fi/autot/vertailut/tm-vertailu-polttoainekayttoiset-lammittimet

/17/ Platform Versions. Android. Viitattu 3.10.2015. https://developer.android.com/about/dashboards/index.html - Platform

/18/ Polttoainelämmittimet. KHP –myynti. Viitattu 1.10.2015. http://www.khp-myynti.fi/32-Polttoainelämmittimet

/19/ Porter, A. 2010. Intoduction. University of Maryland. Viitattu 6.10.2015. https://www.cs.umd.edu/class/fall2010/CMSC498G/CMSC498G/Slides_files/Introduction.pptx

/20/ Services. Android. Viitattu 24.10.2015. http://developer.android.com/guide/components/services.html

/21/ Supporting Different Languages. Android. Viitattu 7.11.2015. http://developer.android.com/training/basics/supporting-devices/languages.html

/22/ Ten Google Play alternatives. Appflood. Viitattu 18.11.2015. http://appflood.com/blog/ten-alternative-android-app-stores

/23/ TL –verkkotoimitus. Lämmitinlaitteiden käyttö pakkasella. Tuulilasi. Viitattu 2.10.2015. http://www.tuulilasi.fi/uutiset/pakkanen-kyykyttaa-akun-webaston-polttoaineen-ja-osaamattoman-korjaamon

/24/ What is dex file and how do I open a dex file? Openthefile. Viitattu 5.10.2015. http://www.openthefile.net/extension/dex

/25/ What is Java technology and why do I need it? Java. Viitattu 5.10.2015. http://java.com/en/download/faq/whatis_java.xml

/26/ What is OpenWeatherMap service. OpenWeatherMap. Viitattu 11.10.2015. http://openweathermap.org/about