

Neuvottelutilakalenterisovelluksen jatkokehitys

Case: Ambientia Group Oy



Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Hämeenlinna, Kevät 2017

Tatu Haimila

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Tekijä	Tatu Haimila	Vuosi 2017
Työn nimi	Neuvottelutilakalenterisovelluksen jatkokehitys	
Työn ohjaaja	Tommi Lahti	

TIIVISTELMÄ

Työn tavoitteena oli jatkokehittää olemassa olevaa Android-pohjaista neuvottelutilakalenterisovellusta lisäämällä siihen uusia toimintoja. Opinnäytetyön toimeksiantaja oli Ambientia Group Oy.

Työn teoriaosassa käsitellään Android-sovelluskehitystä ja käyttöliittymäsuunnittelua. Lisäksi tutustutaan REST-ohjelmointirajapintaan sekä alkuperäiseen neuvottelutilakalenterisovellukseen. Käytännön osassa käydään läpi uusien toimintojen lisääminen sovellukseen.

Neuvottelutilakalenterisovellukseen lisättiin kaksi erilaista asetusvalikkoa, näkymän automaattinen päivitys ja pikavarausmahdollisuus. Sovellus asennetaan kahdelle tabletille Ambientian Hämeenlinnan toimistoon neuvottelutilojen läheisyyteen.

Avainsanat Android, sovelluskehitys, REST, käyttöliittymäsuunnittelu

Sivut 25 s.

Degree Programme in Business Information Technology
Hämeenlinna

Author	Tatu Haimila	Year 2017
Subject	Advanced Development of Meeting Room Application	
Supervisor	Tommi Lahti	

ABSTRACT

The goal of this thesis was to add some new features to Android meeting room application. The client of this thesis was Ambientia Group Oy.

The theory section of this thesis examines Android application development and user interface design. It also views the REST application programming interface and the original meeting room application. The practical section goes through the process of adding the new features into the application.

Two different setting menus, automatic updating for the calendar view and quick reservation of a meeting room were successfully added to the application. In the future, the application will be installed on two Android tablets near meeting rooms in the office of Ambientia at Hämeenlinna.

Keywords Android, software development, REST, user interface design

Pages 25 p.

SISÄLLYS

1	JOHDANTO.....	1
2	ANDROID-SOVELLUSKEHITYS.....	2
2.1	Android.....	2
2.1.1	Android-versiohistoria.....	2
2.2	Android sovelluskehitys	3
2.2.1	App Manifest	3
2.2.2	Aktiviteetti	4
2.2.3	Aktiviteetin elinkaari.....	5
2.2.4	Fragmentti	6
2.2.5	Service	7
2.2.6	Tiedon tallentaminen	7
2.2.7	Sovelluksen käyttöliittymä	8
2.3	Android Studio	10
3	KÄYTTÖLIITTYMÄSUUNNITTELU.....	12
3.1	Käyttöliittymäsuunnittelu yleisesti	12
3.2	Käyttöliittymäsuunnittelu mobiililaitteelle	13
3.3	Käyttöliittymäsuunnittelu tässä työssä	13
4	REST-OHJELMOINTIRAJAPINTA	14
4.1	Resurssit	14
4.2	HTTP-metodit	15
5	NEUVOTTELUTILAKALENTERISOVELLUS.....	17
5.1	Sovelluksen toimintaperiaate	17
5.2	Tavoitellut uudet toiminnot	18
6	UUSIEN TOIMINTOJEN LISÄÄMINEN	19
6.1	Kalenterinäkömän automaattinen päivitys.....	19
6.2	Päivämäärä ja kellonaika toimintopalkkiin	20
6.3	Asetusvalikot.....	21
6.3.1	Näyttöasetukset	21
6.3.2	Kalenteriasetukset.....	22
6.4	Pikavarausmahdollisuus.....	22
7	YHTEENVETO	25
	LÄHTEET.....	26

1 JOHDANTO

Opinnäytetyön tavoitteena on lisätä Android-pohjaiseen neuvottelutilakalenterisovellukseen uusia toimintoja. Työn toimeksiantaja on Ambientia Group Oy, joka on hämeenlinnalaislähtöinen IT-alan yritys. Yhteyshenkilönä opinnäytetyössä toimii Tommi Kaisto.

Valitsin tämän aiheen koska olin kiinnostunut Android-sovelluskehityksestä ja halusin saada siitä lisää kokemusta. Aiempaa kokemusta oli vain koulussa aikaisemmin tehdyistä tätä työtä paljon pienemmistä harjoituksista ja projekteista. Opinnäytetyön alussa käydään ensin läpi taustatietoja Android-sovelluskehityksestä, jatkokehitettävästä neuvottelutilakalenterisovelluksesta, REST-ohjelmointirajapinnasta sekä käyttöliittymäsuunnittelusta. Tämän jälkeen siirrytään käytäntöön ja raportoidaan uusien toimintojen lisäämisestä sovellukseen.

Opinnäytetyön tutkimusongelmana on ”Uusien toimintojen lisääminen Android-pohjaiseen neuvottelutilakalenteriin” ja tutkimuskysymykset ovat seuraavat:

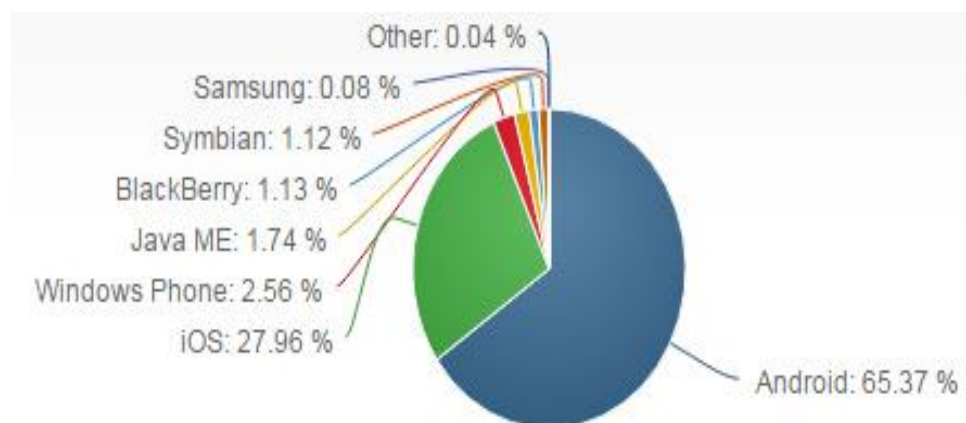
- Miten Neuvottelutilakalenterisovellus toimii?
- Kuinka tiedonsiirto sovelluksen ja palvelimen välillä tapahtuu?
- Kuinka uudet toiminnot toteutetaan?

2 ANDROID-SOVELLUSKEHITYS

2.1 Android

Android on Googlen kehittämä, pääasiassa kosketusnäytöllisille mobiililaitteille suunnattu vapaassa käytössä oleva Linux-pohjainen käyttöjärjestelmä. Sen ensimmäinen versio julkaistiin vuonna 2008, mistä lähtien se on kasvattanut suosiotaan maailman suosituimmaksi mobiilikäyttöjärjestelmäksi sadoilla miljoonilla laitteilla yli 190 eri maassa. Jotain sen suosiosta kertoo myös se, että joka päivä aktivoidaan ensimmäistä kertaa yli miljoona uutta Android laitetta. (Android Developers n.d.a.)

NetMarketShare-sivuston tilastojen mukaan vuonna 2016 Androidin osuus käytössä olevien älypuhelimien ja tablettien käyttöjärjestelmistä oli yli 65 % ja sitä lähimmäs pääsi iOS vajaan 28 %:n suosiollaan, kuten Kuva 1 on nähtävissä. Androidin suosio on kasvanut viime vuosien aikana runsaasti, sillä vielä vuonna 2014 sen osuus oli hieman alle 42 %. (NetMarketShare n.d.)



Kuva 1. Vuonna 2016 käytössä olleiden mobiililaitteiden käyttöjärjestelmien prosenttiosuudet. (NetMarketShare n.d.).

2.1.1 Android-versiohistoria

Vuonna 2008 julkaistiin ensimmäinen Android-versio, Android 1.0, josta on kirjoittamishetkellä päästy versioon 7.1. Viimeisimmät versiot ovat ilmestyneet noin vuoden välein, ja jokainen versio on tuonut korkeamman ohjelmistokehyksen API-tason ansiosta joukon uusia toimintoja. Android-versiot ovat taaksepäin yhteensopivia, joten esimerkiksi Android 4.4 –versiolle suunnattu sovellus toimii myös sitä uudemmissa versioissa. Tämän vuoksi tiheä julkaisutahti ei ole ongelma, vaan kaikki aiemmat sovellukset ovat uusien Android-versioiden käytettävissä. Vanhemmilla Android-versioilla ei kuitenkaan pysty ajamaan uudemmille versioille suunniteltuja sovelluksia. (Kurniawan & Perry 2015, 1–9.)

Yksittäisistä versioista eniten käytössä oleva työn kirjoittamishetkellä on Android 6.0 (Android Developers n.d.b). Käytössä olevien Android-versioiden käyttöosuudet ovat nähtävissä Kuva 2.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.1%
4.1.x	Jelly Bean	16	4.0%
4.2.x		17	5.9%
4.3		18	1.7%
4.4	KitKat	19	22.6%
5.0	Lollipop	21	10.1%
5.1		22	23.3%
6.0	Marshmallow	23	29.6%
7.0	Nougat	24	0.5%
7.1		25	0.2%

Kuva 2. Käytössä olevien Android-versioiden versionumero, koodinimi, API-taso ja osuus käytössä olevista laitteista. (Android Developers n.d.b).

2.2 Android sovelluskehitys

Android-sovellusten pääasiallinen ohjelmointikieli on Java, joka on vuonna 1995 Sun Microsystemsin julkaisema oliopohjainen ohjelmointikieli ja -alusta (Java n.d.). Toinen Android-sovelluskehityksessä paljon käytetty kieli on eXtensible Markup Language eli XML-merkintäkieli, jota käytetään tiedon tallettamiseen ja siirtämiseen. Se on suunniteltu niin ihmisen kuin koneen luettavaksi. (w3schools n.d.a.)

2.2.1 App Manifest

Jokaisella Android-sovelluksella tulee olla App Manifest –XML-tiedosto (AndroidManifest.xml). Siinä määritetään sovellukselle kriittisiä tietoja, joita käyttöjärjestelmä tarvitsee pyörittääkseen sovellusta. Näitä tietoja ovat mm. sovelluksen tunniste, käytetyt ohjelmistokomponentit, pienin API-taso, jonka sovellus vaatii toimiakseen, sovelluksen käynnistyessä ensimmäisenä avattava aktiviteetti, sekä sovelluksen tarvitsemat käyttöoikeudet, jotka käyttäjän tulee hyväksyä API-tasosta riippuen joko asentaessaan sovellusta tai kun käyttöoikeutta tarvitaan ensimmäisen kerran. Tällaisia käyttöoikeuksia voivat olla esimerkiksi lupa luoda tiedostoja tai käyttää kameraa. (Android Developers n.d.c.)

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tatu.exampleapplication">

    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission android:name="android.permission.CAMERA"></uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Example Application"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity"></activity>
    </application>

</manifest>

```

Kuva 3. Esimerkki App Manifest -tiedostosta.

Kuva 3 on Example Application –sovelluksen App Manifest –tiedosto. Sovellukselle on asetettu oikeus käyttää internetyhteyttä ja kameraa. Lisäksi tiedostosta nähdään kaikki sovelluksen aktiviteetit, joista MainActivity-aktiviteetti on asetettu avautumaan ensimmäisenä sovelluksen käynnistyessä. Tiedostossa on myös määritetty sovelluksen käyttämä kuvake ja ulkoisuuden teema.

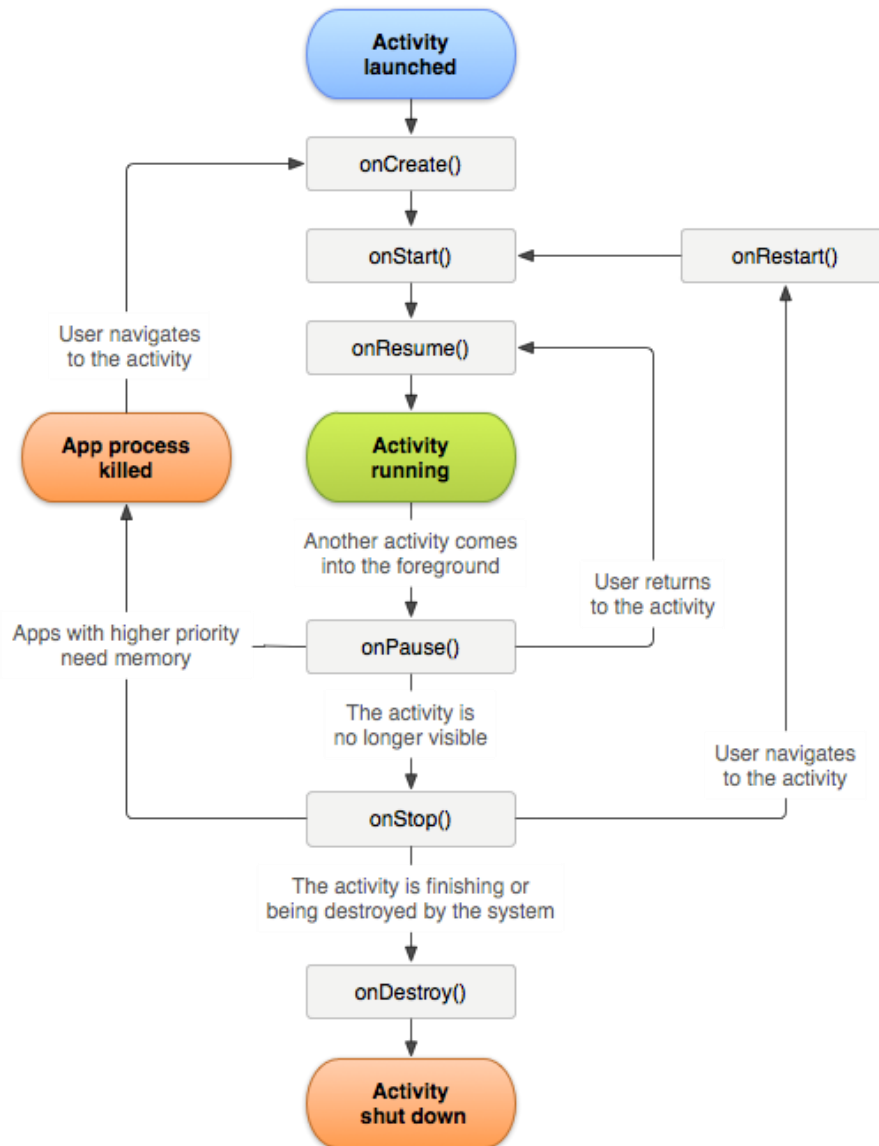
2.2.2 Aktiviteetti

Aktiviteettia voidaan sanoa näkymäksi, jonka avulla käyttäjä toimii sovelluksen kanssa. Useimmissa sovelluksissa on monta eri aktiviteettia ja niitä voidaan muokata käyttöliittymäkomponenteilla tai -kontrolleilla. Aktiviteetit ovat Android-sovelluksen tärkeimpiä ja oleellisimpia rakennuspalikoita, joita pitkin käyttäjä navigoi sovelluksessa ja joiden avulla käyttäjä käyttää sovellusta. Sovelluksen kaikkien aktiviteettien tulee olla rekisteröitynä App Manifest –tiedostossa. Vaikka aktiviteetit luovat yhdessä käyttäjälle yhtenäisen käyttökokemuksen, ei niillä ole suurempia riippuvuuksia toistensa välillä. (Android Developers n.d.d.)

2.2.3 Aktiviteetin elinkaari

Koska käyttäjät käyttävät monia sovelluksia samaan aikaan hyppien sovelluksesta toiseen ja jättäen niitä auki taustalle, on sovellusten ja niiden aktiviteettien osattava toimia sen mukaan. Tätä varten aktiviteetillä on joukko elinkaarimetodeja, joita se kutsuu tietyssä elinkaarensa mukaisessa vaiheessa. Näitä metodeja ovat *onCreate*, *onStart*, *onResume*, *onPause*, *onStop*, *onDestroy*, sekä *onRestart*. (Android Developers n.d.f.)

Aktiviteetin elinkaarimetodien avulla voidaan määrittää, miten aktiviteetti käyttäytyy sen elinkaaren eri vaiheissa, kuten mitä aktiviteetti tekee ennen kuin se tulee näkyviin tai miten aktiviteetti reagoi, kun se poistuu näkyvistä ja palaa hetken päästä takaisin. Metodien avulla voidaan esimerkiksi poistaa internetyhteys sovelluksen käytöstä, jos kyseinen sovellus ei ole sillä hetkellä aktiivisena käytössä. Tällöin saadaan säästettyä käytetyn laitteen virtaa ja vaaditut resurssit vapautuvat muiden, sillä hetkellä tärkeämpien sovellusten käyttöön. Elinkaarimetodien avulla voidaan myös varmistaa, etteivät sovellukset kaadu tai käyttäjät menetä edistymistään, kuten esimerkiksi täytettyjä lomaketietoja, mikäli käyttäjä siirtyy toiseen sovellukseen tai aktiviteetti siirtyy muuten yllättäen pois näkyvistä esimerkiksi käyttäjän vastaanottaessa puhelun. (Android Developers n.d.f.)



Kuva 4. Aktiviteetin elinkaari. (Android Developers, n.d.f).

Kuva 4 on nähtävissä, missä järjestyksessä aktiviteetin elinkaarimetodeja kutsutaan ja miten aktiviteetti reagoi niiden avulla käyttäjän liikkeisiin. Aktiviteetti tulee käyttäjälle näkyviin kutsuttuaan ensin kolme ensimmäistä elinkaarimetodia (`onCreate`, `onStart`, `onResume`). Kaikki tarvittava valmistelu tulee tehdä näiden metodien aikana, jotta aktiviteetti on valmis käytettäväksi. Kun käyttäjä siirtyy aktiviteetista toiseen aktiviteettiin tai sovellukseen, kutsuu aktiviteetti `onPause`-metodia, jossa voidaan esimerkiksi tallentaa tarvittavat tiedot ja asettaa pois ominaisuudet joita ei tarvita, kun sovellus ei ole näkyvissä. Nämä ominaisuudet voidaan asettaa taas käynnistyväksi `onResume`-metodissa, kun sovellus palaa takaisin näkyviin.

2.2.4 Fragmentti

Aktiviteetin eli näkymän sisällä voi olla ns. alinäkymiä eli fragmentteja. Fragmentti voi olla muokattavissa oleva käyttöliittymän osa tai se voi ku-

vata jotakin toimintoa aktiviteetissa ja niiden avulla käyttöliittymistä voidaan tehdä dynaamisempia. Yhdessä aktiviteetissa voi olla useampia fragmentteja ja niitä voidaan kontrolloida yksitellen. Lisäksi samaa fragmenttia voidaan käyttää useammassa eri aktiviteetissa. Fragmentilla on oma elinkaarensa, mutta se on riippuvainen sen isäntäaktiviteetin, eli sen aktiviteetin jonka sisälle fragmentti on luotu, elinkaaresta. Jos aktiviteetti tuhoetaan, niin myös kaikki sen sisältämät fragmentit tuhoutuvat. (Android Developers n.d.e.)

2.2.5 Service

Service eli palvelu on ohjelmistokomponentti, jonka avulla voidaan suorittaa pitkäkestoisia toimintoja tausta-ajona ilman omaa käyttöliittymää. Jokainen palvelu tulee esitellä App Manifest -tiedostossa ja niitä voidaan käyttää esimerkiksi musiikin tai äänen toistamiseen. Toinen ohjelmistokomponentti, esimerkiksi aktiviteetti, voi käynnistää palvelun ja sen voidaan määrittää jatkavan toimintaansa, vaikka sen käynnistänyt ohjelmistokomponentti tuhoettaisiin. Palvelu voidaan myös yhdistää toiseen ohjelmistokomponenttiin, jolloin ne voivat olla vuorovaikutuksessa keskenään. Tällöin palvelu myös tuhoutuu samalla siihen yhdistetyn ohjelmistokomponentin mukana, eikä jää turhaan pyörimään taustalle vieden resursseja muilta sovelluksilta. (Android Developers n.d.i.)

2.2.6 Tiedon tallentaminen

Monille sovelluksille jonkinlaisen tiedon tallentaminen on välttämätöntä. Android-sovelluksissa siihen on kolme pääasiallista tapaa. Tallentamiseen voidaan käyttää avain – arvo -pareja, tietoa voidaan tallentaa erilliseen tiedostoon tai siihen voidaan käyttää SQLite-tietokantaa. Näistä mahdollisuuksista voidaan valita käyttöön sopivin, kun otetaan huomioon tallennettavan tiedon määrä ja sen käyttötarkoitus. (Android Developers n.d.j.)

SharedPreferences tarjoaa tavan tallentaa yleisimpiä tietotyyppisiä helposti avain – arvo-parien avulla. Se toimii siten, että määritetään avain, esimerkiksi "Käyttäjänimi", sekä tallennettava arvo, esimerkiksi "Tatu". Tallennuksen jälkeen tallennettu arvo voidaan hakea SharedPreferences-tiedoista avaimen nimellä. (Android Developers n.d.k.)

Monissa sovelluksissa pitää tallentaa käyttäjälle tärkeitä tietoja, kuten käyttäjätietoja ja salasanoja, jotta niitä ei tarvitse syöttää uudestaan jokaisella käyttökerralla. Tällöin on tärkeää huolehtia tietoturvasta. Yleisimmin tietoturvaongelmia muodostuu liian turvattomasti laitteeseen tallennetun tiedon vuoksi. Optimaalisin ratkaisu olisi se, ettei tietoja tarvitsisi tallentaa laitteeseen ollenkaan. Se ei kuitenkaan ole useimmiten mahdollista, joten tiedot täytyy tallentaa mahdollisimman tietoturvallisesti. (Nolan 2015.)

Yksinkertaisin ja samalla tietoturvallisesti huonoin ratkaisu on tallentaa salasana ja muut tärkeät tiedot suoraan laitteeseen, käytti sitten mitä tallennustapaa tahansa. Tietoturvaa parantaa jo huomattavasti se, jos salana ensin salataan jollakin salausmenetelmällä ja tallennetaan laitteeseen vasta salauksen jälkeen. Tällöin mikäli joku saa salasanan käsiinsä, ei se ole vielä luettavassa muodossa, vaan myös salaus tulee murtaa. (Nolan 2015.)

Android-sovelluksissa kaikista paras ratkaisu on käyttää Android Keystore systeemiä. Sinne voidaan tallentaa salattuja avaimia, jotka ovat erityisen vaikeasti saatavissa ulos laitteesta. Android Keystore lisättiin vasta Android 4.3 –versioon, joten se ei ole käytettävissä sitä aiemmissa Android-versioissa. (Android Developers n.d.l.)

2.2.7 Sovelluksen käyttöliittymä

Android sisältää monia erilaisia käyttöliittymäelementtejä, joita voi käyttää käyttöliittymän ulkoasun luomiseen ja muokkaamiseen. Tällaisia elementtejä ovat esimerkiksi erilaiset tekstiruudut, painikkeet tai kuvat. Android Studioon käyttöliittymäeditorin avulla ulkoasun muokkaaminen on melko nopeaa, sillä erilaisia elementtejä voi lisätä käyttöliittymään haluttuun paikkaan helposti hiirellä raahaamalla. Jokaisen käyttöliittymäelementin ominaisuudet tallentuvat XML-merkintäkielellä kirjoitettuun käyttöliittymän ulkoasusta vastaavaan layout-tiedostoon, josta käsin käyttöliittymän ja ulkoasun muokkaaminen on myös mahdollista. (Kurniawan & Perry 2015, 51–55.)

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="#c2c2c2"
    tools:context="com.example.tatu.exampleapplication.MainActivity">

    <TextView
        android:text="Example"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView_title"
        android:layout_marginTop="53dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="60sp" />

    <Button
        android:text="Start"
        android:layout_width="150dp"
        android:layout_height="80dp"
        android:id="@+id/button_start"
        android:textSize="16sp"
        android:layout_marginTop="50dp"
        android:layout_below="@+id/textView_title"
        android:layout_centerHorizontal="true" />

    <Button
        android:text="Exit"
        android:layout_width="150dp"
        android:layout_height="80dp"
        android:id="@+id/button_exit"
        android:textSize="16sp"
        android:layout_marginTop="20dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/button_start" />

</RelativeLayout>

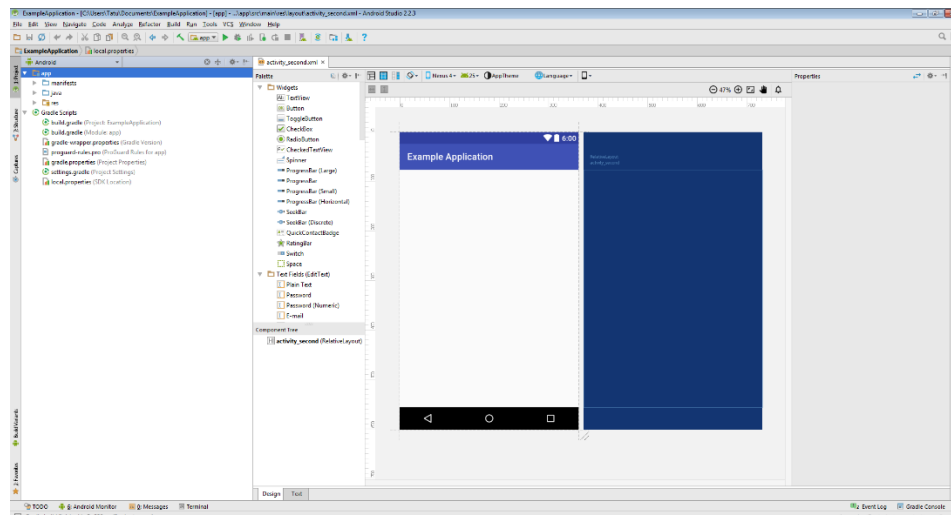
```

Kuva 5. Esimerkki layout-tiedostosta.

Kuva 5 on yksinkertaisen käyttöliittymän ulkoasusta vastaava layout-tiedosto. Tiedosto on kirjoitettu XML-kielellä. Käyttöliittymään on lisätty kolme käyttöliittymäelementtiä, yksi tekstikenttä ja kaksi painiketta. Jokaiselle elementille on määritetty mm. id, koko, näytettävä teksti sekä sen sijainti käyttöliittymässä. Se, miltä käyttöliittymä näyttää oikeasti puhelimen ruudulla, on nähtävissä Kuva 7.

2.3 Android Studio

Android Studio on ilmaiseksi ladattavissa oleva virallinen ohjelmointiympäristö Android-sovelluskehitykseen, joka sisältää kaikki tarvittavat työkalut Android-sovellusten rakentamiseen. Android Studio (Kuva 6) on rakennettu IntelliJ IDEA Java –ohjelmointiympäristön pohjalle ja sen ensimmäinen stabiili versio julkaistiin loppuvuodesta 2014. (Android Developers n.d.a.)



Kuva 6. Yleiskuva Android Studiosta.

Android Studion sisältämä Android Virtual Device Manager sisältää laajan valikoiman erilaisia Android virtuaalilaitteita, kuten älypuhelimia, tabletteja, sekä TV- ja Wear-laitteita. Virtuaalilaitteella tarkoitetaan virtuaalisessa muodossa olevaa laitetta, joka kuitenkin sisältää kaikki oikean laitteen ominaisuudet ja määrytykset. Android Virtual Device Managerissa laitteita pystyy myös muokkaamaan vapaasti, esimerkiksi muistin ja näytön koon osalta, jos juuri haluttua mallia ei löydy valikoimasta. (Android Developers n.d.g.)

Jotta virtuaalilaitetta voidaan käyttää tietokoneella, tarvitaan siihen emulaattori. Emulaattorin tehtävä on simuloida virtuaalilaitteen toimintaa, jolloin sitä päästään varsinaisesti käyttämään. Android Studiossa emulaattorina toimii Android Emulator, jonka avulla Androidin virtuaalilaitteita voidaan ajaa tietokoneella. Tämä auttaa paljon sovellusten kehityksessä testauksen osalta, sillä kehitettyjä Android-sovelluksia voidaan testata helposti ja nopeasti eri laitteilla ilman, että laitteita tarvitsee omistaa fyysisesti. Android Emulatorilla pystyy testaamaan lähes kaikkia laitteen ominaisuuksia aina puhelun vastaanottamisesta lämpötilasensoreihin. (Android Developers n.d.h.)



Kuva 7. Esimerkkisovellus avattuna Nexus 5 –virtuaalilaitteella Android Emulatorissa.

3 KÄYTTÖLIITTYMÄSUUNNITTELU

3.1 Käyttöliittymäsuunnittelu yleisesti

Käyttöliittymällä tarkoitetaan tietokoneohjelman visuaalista, eli näkyvää puolta, jonka avulla käyttäjä kontrolloi ohjelmaa. Käyttöliittymäsuunnittelulla puolestaan tarkoitetaan käyttöliittymien suunnittelua ja sen tarkoituksena on tehdä käyttäjän ja tietokoneen kanssakäymisestä mahdollisimman luontevaa.

Hyvän käyttöliittymän tulisi olla selkeä ja esteettisesti miellyttävä. Siinä käytettyjen visuaalisten elementtien tulisi olla helposti ymmärrettäviä ja opittavia, sekä johdonmukaisesti sijoitettuja, aina samaan paikkaan tai samalla tavalla aseteltuna, jos mahdollista. Mikäli käyttäjälle aiemmin tuttuja elementtejä tai konsepteja on mahdollista käyttää, tulisi näin tehdä. Värejä olisi hyvä käyttää vaatimattomasti, mutta kuitenkin tehokkaasti, ja näkyvien elementtien tulisi olla linjattu ja ryhmitelty sopivan kontrastin luomiseksi. (Galitz 2007, 45–58.)



Kuva 8. Esimerkki yleisistä kuvakkeista.

Kuva 8 on usein käytettyjä kuvakkeita, joista käyttäjä pystyy päättämään niiden ulkoasun sekä aikaisempien kokemusten perusteella mitä niistä painamalla tapahtuu. Vasemmalta katsottuna ensimmäisenä on kuvake, jota käytetään usein erilaisten asetusten kuvakkeena. Seuraavana on eri muodoissa paljon käytetty oikein-merkki, joka on myös yleensä vihreään väriin. Sen nähdessään tai sitä painaessa käyttäjä tietää käyttötapauksesta riippuen esimerkiksi antaneensa oikean vastauksen, vastaavansa myöntyvästi tai vahvistavansa antamansa tiedot. Kolmantena on pallon sisällä oleva kysymysmerkki. Mikäli käyttäjä tarvitsee lisää tietoja, apua tai ohjeita, osaa hän painaa kuvakkeesta tai etsiä tietoa kuvakkeen läheisyydestä. Kaikille merkeille on yhteistä se, että ne ovat paljon käytettyjä eli käyttäjällä on todennäköisesti aikaisempaa kokemusta niistä, jolloin niiden käyttötarkoituksen päättely helpottuu.

Kaikkien toimintojen, jotka käyttäjän on mahdollista suorittaa, tulisi olla aina saatavilla ja näkyvissä. Kuten visuaalisten elementtien, myös toimintojen tulisi olla ennustettavia ja helposti opittavia, sekä loogisesti eteneviä ja yhteneväisesti toteutettuja. Jotta käyttäjälle tulisi tunne, että hän kont-

rolloi laitetta, jokaisesta hänen liikkeestään tulisi seurata jokin efekti, kuten näkymän muutos tai näppäinääni. Hyvä käyttöliittymä antaa anteeksi inhimilliset virheet kuten virhepainallukset. Siinä on mahdollista palata aikaisempaan näkymään tai tilaan ja käyttöliittymä ilmoittaa käyttäjän mahdollisista virheistä. On myös syytä muistaa, että käyttäjän edut menevät aina teknisten ratkaisujen etujen edelle. (Galitz 2007, 45–58.)

3.2 Käyttöliittymäsuunnittelu mobiililaitteelle

Kun käyttöliittymää suunnitellaan mobiililaitteelle, on siinä useita huomioitavia yksityiskohtia verrattuna esimerkiksi tietokoneelle suunniteltaessa. Erityistä huomiota tulee asettaa kuvaruutujen yleisesti pienempään koon ja mobiililaitteiden näyttöjen koon vaihteluun. Mobiililaitteen ruudulle ei mahdu niin paljoa tekstiä tai muuta sisältöä kuin tietokoneen ruudulle. Lisäksi mobiilikäyttöliittymän olisi hyvä olla responsiivinen eli mukautua näytön koon mukaan, jotta käyttöliittymän sisältö olisi aina mahdollisimman sopivan kokoista. (Cohen & Wang 2014, 2–7.)

Suurin osa mobiililaitteista on kosketusnäytöllisiä, mikä tarkoittaa sitä, että valinta ei ole yhtä tarkka ja nopea kuin tietokoneella hiirtä käytettäessä. Kun epätarkka valinta yhdistetään pienempään näyttöön voi se aiheuttaa ongelmia esimerkiksi käytettäessä tietokoneelle suunnattua käyttöliittymää mobiililaitteella. Hyvä ratkaisu on selkeyttää käyttöliittymää poistamalla siitä kaikki turha sisältö ja tehdä painettavista kuvakkeista hieman suurempia. (Cohen & Wang 2014, 2–7.)

Myös käytettävissä oleva näppäimistö on virtuaalinen ja se sisältää yleensä vain tärkeimmät näppäimet tai vähemmän käytetyt merkit ovat vähintään useamman painalluksen takana. Tämä yhdistettynä epätarkempaan valintaan tarkoittaa sitä, että kirjoittaminen vie enemmän aikaa ja virhepainallusten määrä on suurempi. Ratkaisuna voidaan luoda mukautettuja virtuaalisia näppäimistöjä, joissa on valittavana vain tarvittavia näppäimiä, kuten numeroita. (Cohen & Wang 2014, 2–7.)

3.3 Käyttöliittymäsuunnittelu tässä työssä

Tämän työn käyttöliittymäsuunnittelu eroaa normaalista siinä mielessä, että käyttöliittymää ei suunnitella alusta lähtien, koska sovelluksella on jo olemassa käyttöliittymä. Käyttöliittymäsuunnittelu koostuukin uusien toimintojen myötä tulevista uusien elementtien, kuten valikoiden ja ilmoitusten toteutustavan suunnittelusta. Niiden toteutuksessa pyritään huomioimaan edellisissä kappaleissa esiteltyjä käyttöliittymäsuunnittelun pääkohdista, sekä mobiilikäyttöliittymäsuunnittelussa huomioitavia asioita.

Normaaliin käyttöliittymäsuunnitteluprosessiin kuuluu aina olennaisena osana myös käyttöliittymätestaus, mutta erillistä testausta ei opinnäytetyössä järjestetä.

4 REST-OHJELMOINTIRAJAPINTA

Ohjelmointirajapinta, eli API (Application Programming Interface), määrittää tavan, jolla ohjelmistokomponentit pystyvät kommunikoimaan keskenään (Tutorialspoint n.d.).

REpresentational State Transfer eli REST on Roy Fieldingin vuonna 2000 esittelemä joukosta rajoitteita muodostuva tapa siirtää dataa asiakasohjelman ja palvelimen välillä. Pääpiirteissään rajoitukset menevät siten, että järjestelmän tulee käyttää standardoituja metodeja ja mediatyyppejä, sekä yhteydenpidon asiakasohjelman ja palvelimen välillä tulisi olla tilatonta, eli jokaisen pyynnön tulisi olla itsenäinen eikä yksittäinen pyyntö saa sisältää tietoa muista pyynnöistä. Lisäksi jokaisella käytetyllä resurssilla tulee olla uniikki ID ja resurssien tulee olla linkitettyinä toisiinsa, luoden niiden välille yhteyksiä. Kun kaikki vaaditut rajoitteet toteutuvat järjestelmässä, voidaan järjestelmää kutsua REST-pohjaiseksi. REST-pohjaisten järjestelmien hyötyjä ovat mm. helppo ylläpidettävyys ja laajennettavuus. (Fielding 2000; Abeysinghe 2008, 13–14; Sandoval 2009, 7–11)

4.1 Resurssit

Resurssilla tarkoitetaan tässä tapauksessa mitä tahansa dataa, mitä voidaan siirtää asiakasohjelman ja palvelimen välillä. Resurssi voi esimerkiksi olla uutisraportti tai lista työntekijöistä. Jokaisella resurssilla tulee määrittysten mukaa olla oma URI (Uniform Resource Identifier) eli osoite, josta kyseinen resurssi on palvelimelta haettavissa. Resurssilla voi olla monta erilaista esitysmuotoa. Web-selainten kohdalla esitystapa on yleensä HTML-sivu, koska ihmisten on tarkoitus lukea sitä. Automatisoitujen pyyntöjen kohdalla luettavuus ei ole yhtä tärkeässä asemassa, joten tehokkaampia esitystapoja resurssille ovat esimerkiksi tiedonvälitykseen soveltuvat XML-merkintäkieli (Kuva 9) ja JSON (Kuva 10). (Sandoval 2009, 7-11.) JavaScript Object Notation eli JSON on tiedostomuoto, joka on XML-merkintäkielen tavoin luotu tiedon tallettamiseen ja siirtämiseen ja jota niin ihmiset kuin koneet pystyvät lukemaan (w3schools n.d.b).

```

<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>

```

Kuva 9. Esimerkki XML-merkintäkielestä. (w3schools n.d.c).

```

{"employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}

```

Kuva 10. Esimerkki JSON-tiedostomuodosta. (w3schools n.d.c).

4.2 HTTP-metodit

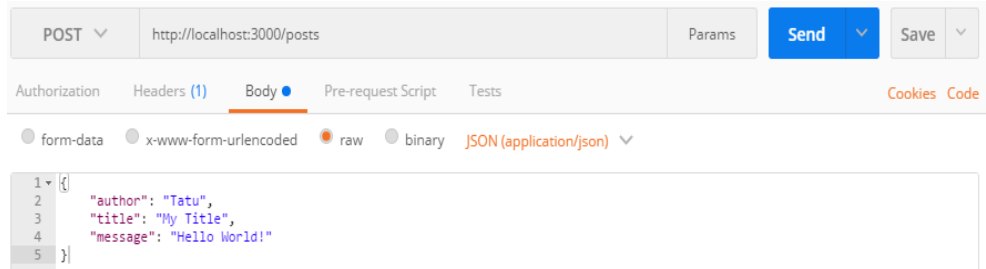
REST-pohjaisten ohjelmointirajapintojen avulla sovellukset ja järjestelmät pystyvät kommunikoimaan keskenään HTTP-metodien avulla. Yleisimmin käytetyt HTTP-metodit ovat GET, POST, PUT ja DELETE. (Sandoval 2009, 7–11).

GET-metodia käytetään tiedon lukemiseen, sillä sen avulla vastaanotetaan resurssi valitussa esitysmuodossa. Koska metodilla ainoastaan luetaan tietoa, eikä muuteta mitään, on sen käyttö riskitöntä datan muuttumisen tai korruptoitumisen kannalta. POST-metodia puolestaan käytetään pääsääntöisesti uusien resurssien luomiseen. (Gulabani 2014, 16–18).

PUT-metodilla voidaan korvata olemassa olevia resursseja uudella tiedolla, joten sitä käytetään suurimmaksi osin tietojen päivittämiseen. Joissakin tapauksissa PUT-metodia voidaan myös käyttää uusien resurssien luomiseen POST-metodin ohella. Resurssien poistaminen suoritetaan DELETE-metodilla. (Gulabani 2014, 16–18)

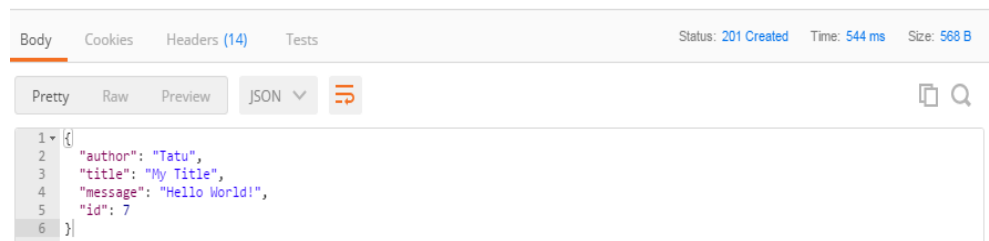
Seuraavassa esimerkissä käytetään paikallista REST-ohjelmointirajapintaa, jonka avulla voidaan lisätä viestejä tietokantaan, sekä hakea viestejä tietokannasta ID-numeron perusteella. Viesti sisältää tiedon viestin luojasta, otsikosta ja itse viestin sisällöstä. HTTP-pyyntöjen lähettämiseen käytettiin siihen tarkoitettua Postman-sovellusta.

Kuva 11 tietokantaan lisätään POST-pyyntöllä viesti "Hello World!". Lähetettävä sisältö on JSON-tiedostomuodossa.



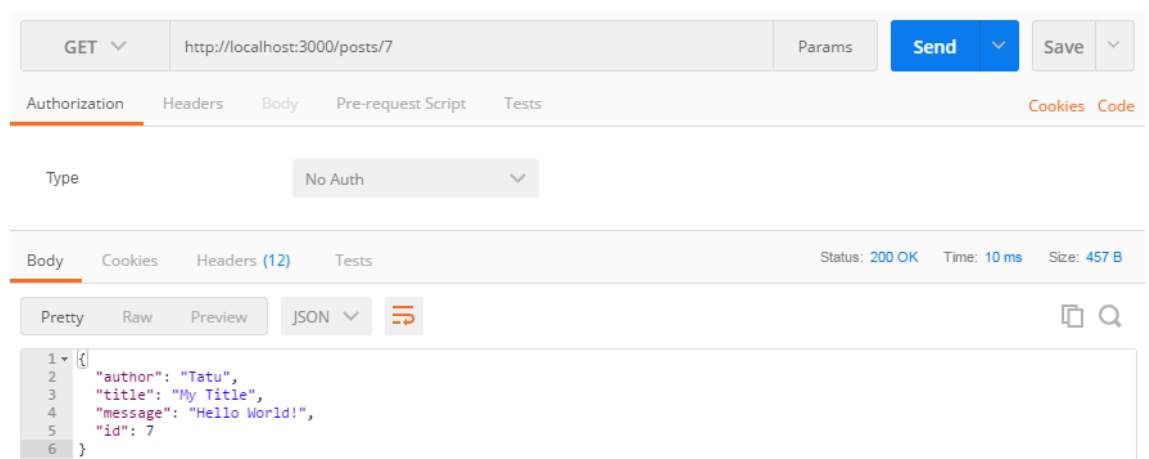
Kuva 11. POST-pyyntö Postman-sovelluksessa.

Kuva 12 on nähtävissä Kuva 11 POST-pyyntöstä saatu vastaus. Oikeassa yläreunassa on statuskoodi "201 Created", joka tarkoittaa sitä, että resurssi luotiin onnistuneesti tietokantaan. Statuskoodeilla ilmoitetaan pyynnön tila ja niiden avulla saadaan tietoon ongelmatapauksissa melko tarkasti missä vaiheessa virhe tapahtui. Saadun vastauksen sisältö on sama, kuin lähetetyssä POST-pyyntössä, mutta vastauksessa näkyy myös juuri luodun viestin ID-numero tietokannassa.



Kuva 12. Postman-sovelluksessa suoritetusta POST-pyyntöstä saatu vastaus.

Kun viestin ID-numero on tiedossa, voidaan kyseinen viesti hakea tietokannasta GET-pyyntön avulla (Kuva 13). Statuskoodi 200 tarkoittaa onnistunutta pyyntöä ja saatu vastaus on täysin sama kuin resurssia luodessa POST-pyyntöstä saatu vastaus.



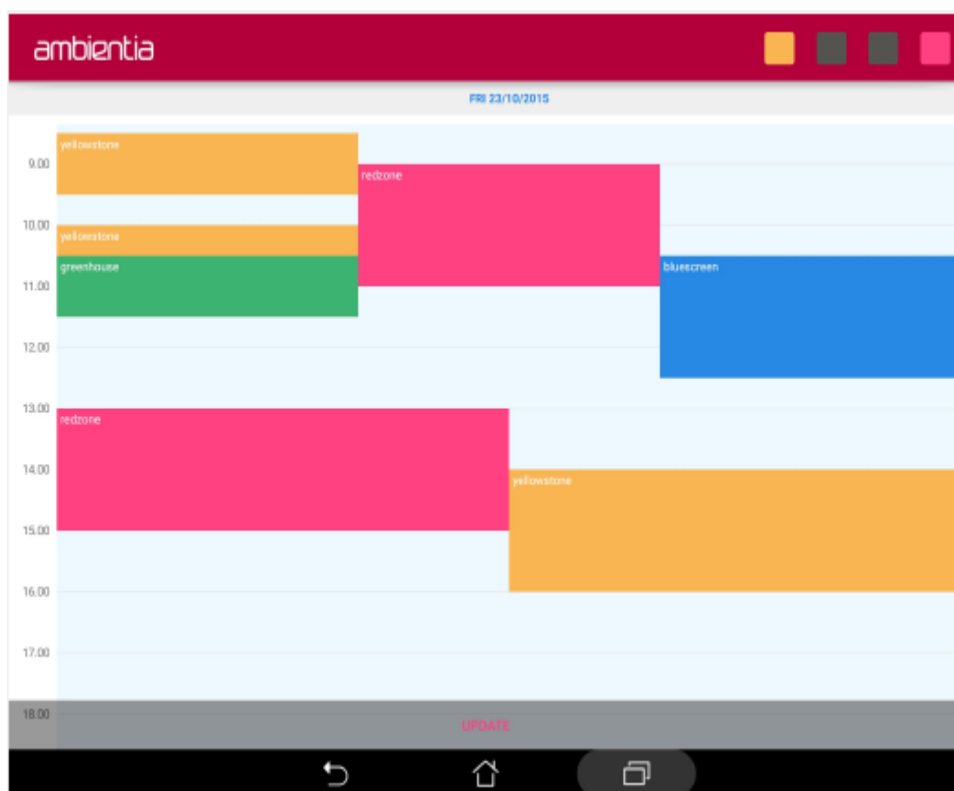
Kuva 13. Postman-sovelluksessa suoritettu GET-pyyntö ja siitä saatu vastaus.

5 NEUVOTTELUTILAKALENTERISOVELLUS

Neukkari-Seppo nimellä kulkeva neuvottelutilakalenterisovellus on alun perin Ambientian Tommi Kaiston kehittämä Android-sovellus neuvottelutilojen varausten hallintaan. Sen avulla Ambientian työntekijät näkevät kulloinkin vapaana olevat neuvotteluhuoneet, sekä niiden varaukset.

5.1 Sovelluksen toimintaperiaate

Opinnäytetyön aloittamishetkellä sovellus toimii siten, että se hakee Update-painiketta painamalla palvelimelta tiedot varauksista ja luo niistä siten lukujärjestyksellisen kalenterinäkymän ((w3schools n.d.c). Kuva 14). Tietojen haku palvelimelta tapahtuu REST-ohjelmointirajapinnan avulla. Jokainen neljästä eri väristä kuvaa eri neuvotteluhuonetta. Oikeassa yläreunassa olevista neuvotteluhuonekuvakkeista eli ns. liikennevaloista näkee sillä hetkellä vapaana olevat neuvotteluhuoneet, varattujen ollessa harmaana. Sovelluksessa voi liikkua eri päivien välillä pyyhkäisemällä näytöstä vasemmalle tai oikealle.



(w3schools n.d.c). Kuva 14. Neukkari-Seppo-sovellus.

Sovelluksen tärkeimpiä komponentteja on MainActivity-aktiiviteetti, joka vastaa sovelluksen oletusnäkyä ((w3schools n.d.c). Kuva 14). Se on määritetty App Manifest -tiedostossa avattavaksi ensimmäisenä, kun sovellus käynnistetään. MainActivity-aktiiviteetissa hallitaan myös toiminto-

palkissa olevia neuvotteluhuoneiden kuvakkeita, sekä varausten näyttämistä ja päivittämistä, vaikka yhteydenpito sovelluksen ja palvelimen välillä tapahtuukin ReservationService-luokassa.

5.2 Tavoitellut uudet toiminnot

Opinnäytetyön tavoitteena on lisätä uusia toimintoja sovellukseen. Tavoiteltuja toimintoja ovat kalenterinäkymän automaattinen päivitys, neuvottelutilan pikavarausmahdollisuus sekä kaksi erilaista asetusvalikkoa, näyttöasetukset ja kalenteriasetukset. Lisäksi sovellukseen haluttiin lisätä kellonaika ja päivämäärä toimintopalkkiin helpottamaan varausten lukemista ja luomista.

Kalenterinäkymän automaattisella päivittämisellä tarkoitetaan sitä, että näkymä päivittyisi automaattisesti tietyn ajan välein, jotta kalenteri pysyy ajan tasalla ilman että näkymää tarvitsee päivittää manuaalisesti. Näkymän päivitysväli määritetään näyttöasetuksissa, jossa myös valitaan näytettävät neuvotteluhuoneet, jolloin sovelluksen voi halutessaan asettaa näyttämään esimerkiksi vain yhden neuvotteluhuoneen varauksia. Toisessa asetusvalikossa, eli kalenteriasetuksissa, asetetaan käytetyn kalenterin URL-osoite ja sen käyttöön vaaditut käyttäjätiedot.

Neuvottelutilan pikavaraus tapahtuu painamalla halutun neuvotteluhuoneen liikennevalosta, jolloin avautuu varausdialogi. Varausdialogissa näytetään mihin neuvotteluhuoneeseen varausta ollaan tekemässä ja siinä määritetään varauksen kesto. Varauksen alkamisajaksi asetetaan se hetki, kun käyttäjä painaa varausdialogista ”Tee varaus”-painiketta. Dialogissa tulisi myös olla ”Peruuta”-painike, jos käyttäjä ei haluakaan tehdä varausta. Varaus tehdään palvelimelle REST-ohjelmointirajapinnan avulla, kuten varausten lukeminenkin.

6 UUSIEN TOIMINTOJEN LISÄÄMINEN

6.1 Kalenterinäköymän automaattinen päivitys

Käytännön osuus aloitettiin työskentelemällä kalenterinäköymän automaattisen päivityksen parissa. Projektiin lisättiin TimerService-palveluluokka, jonka tarkoituksena on hallita kalenterinäköymän päivitykseen tarvittavaa ajastinta. Luokkaan lisättiin näköymän päivitystä hallitseva ajastin, sekä metodit ajastimen käynnistämiseksi ja pysäyttämiseksi. Lisäksi ennen ajastimen käynnistämistä tarkistetaan, onko näköymää päivittäviä ajastimia jo käynnissä, jolloin vältetään mahdollisuudelta asettaa useampi ajastin samanaikaisesti.

Jotta TimerService-luokassa sijaitsevia ajastimen hallinnasta vastaavia metodeita päästiin kutsumaan MainActivity-luokasta, tarvittiin Bound Service –rajapintaa. Sen avulla voidaan yhdistää komponentteja palveluihin. Sen hyvä puoli on myös se, että palvelu voidaan asettaa käynnistymään samalla, kun yhteys sen ja komponentin välille luodaan ja tuhoutumaan kun yhteys puretaan. Näin ollen palvelu ei jää turhaan taustalle pyörimään, kun sitä ei enää tarvita.

Koska TimerService-luokasta tarvitsee myös kutsua MainActivity-luokan metodeja ajastimen ajan täytyessä, luotiin sitä varten TimerServiceCallbacks-rajapinta. Se sisältää metodin, joka on asetettu kutsuttavaksi, kun ajastimeen määritetty aika täyttyy. Kun MainActivity-luokka toteuttaa TimerServiceCallbacks-rajapinnan ja ylikirjoittaa sen metodit, voidaan niiden avulla MainActivity-luokassa toteuttaa asioita TimerService-luokasta käsin, eli tässä tapauksessa päivittää kalenterinäköymä.

Ajastimelle voidaan ajastaa TimerTask-olio, jonka avulla ajastinta ei tarvitse käynnistää uudestaan jokaista päivitystä varten, vaan siinä määritetään viive ennen kuin haluttu toiminto suoritetaan sekä aikaväli siitä seuraavaan suorituskertaan. Ajastin hakee aina ennen käynnistymistään päivitysvälin SharedPreferences-tiedoista, joihin haluttu päivitysväli on tallennettu näytöasetuksissa.

Ajastin on asetettu käynnistymään ensimmäisen kerran, kun yhteys MainActivity-aktiviteetin ja TimerService-luokan välille luodaan. Tämä tapahtuu heti sovelluksen käynnistyessä. Ajastin vuorostaan pysähtyy silloin, kun MainActivity-aktiviteetti siirtyy pois näkyvistä taustalle, eli MainActivity-aktiviteetin kutsuessa onPause-elinkaarimetodia. Ajastin vuorostaan käynnistyy uudelleen ja päivittää näköymän MainActivity-aktiviteetin onResume-elinkaarimetodia kutsuttaessa eli kun käyttäjä palaa sovellukseen. Yhteys TimerService-luokan ja MainActivity-aktiviteetin välillä puretaan jälkimmäisen onDestroy-elinkaarimetodissa, jolloin TimerService-palvelu tuhoutuu ja ajastin pysähtyy.

6.2 Päivämäärä ja kellonaika toimintopalkkiin

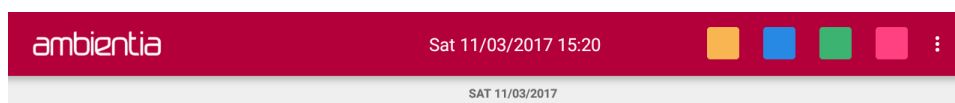
Sovellukseen haluttiin lisätä päivämäärä ja kellonaika niiden ollessa oleellisia tietoja varauksia tehdessä ja lukiessa. Koska MainActivity-aktiviteetissa ei ollut muualla juuri tilaa, päätettiin aika ja päivämäärä asettaa keskelle toimintopalkkia. Toimintopalkille luotiin oma layout-tiedosto ja sinne lisättiin tekstikenttä, jonne päivämäärä ja aika asetettiin. Jotta aikaa saataisiin myös päivitettyä, luotiin sitä varten TimerService-luokkaan uusi ajastin. Ajastin toteutettiin muuten täysin samalla tavalla ja se käyttäytyy samoin tavoin kuin kalenterinäköymän päivytyksestä vastaava, mutta ajan täytyessä kutsuttava metodi on eri ja päivitysväliseksi asetettiin minuutti. Näin ollen myös TimerServiceCallbacks-rajapintaan täytyi luoda uusi metodi, jota ajastin kutsuisi ajan täytyessä ja sitä kautta päivittäisi ajan.

Toimintopalkissa olevan tekstikentän kanssa ongelmaksi tuli se, että kun tekstikenttä määritettiin asettuvaksi keskelle, asettui se keskelle sitä tilaa, mikä jäi vasemman reunan Ambientia-tekstin ja neuvotteluhuonekuvakkeiden väliin (Kuva 15), vaikka oikeasti tavoiteltiin asettumista koko toimintopalkin keskelle. Siirtämälle tekstikenttää manuaalisesti toimintopalkin layout-tiedostossa hieman oikealle, saatiin se asettumaan vaakasuorassa landscape-tilassa halutulle paikalle. Kuitenkin kun tabletti käännettiin pystysuoraan portrait-tilaan, jäi tekstikenttä manuaalisten säätöjen mukaisesti liiaksi oikealle piiloon neuvotteluhuonekuvakkeiden alle, eikä ollut ollenkaan näkyvässä. Tämä korjattiin siten, että projektiin luotiin uusi layout-port-kansio, jonne lisättiin layout-tiedosto portrait-tilaa varten. Näin sovellus osaa hakea oikean laitteen tilaa vastaavan layout-tiedoston, kun aktiviteetti luodaan.



Kuva 15. Kellonaika ja päivämäärä toimintopalkissa väärällä paikalla.

Ongelmaksi jäi vielä se, että kun laite käännettiin esimerkiksi vaakasuorasta landscape-tilasta pystysuoraan portrait-tilaan aktiviteetin luonnin jälkeen, jäi landscape-tilaan suunniteltu ulkoasu voimaan. Tämän ratkaisemiseksi MainActivity-aktiviteettiin lisättiin onConfigurationChanged-metodi, jota kutsutaan aktiviteetin tilan muuttuessa. Metodiin asetettiin layout-tiedoston uudelleen haku. Näin ollen aina laitetta kääntäessä se osaa hakea layout-tiedoston uudelleen, jolloin käytössä on aina oikea layout-tiedosto ja ulkoasu on toivottu (Kuva 16).



Kuva 16. Kellonaika ja päivämäärä toimintopalkissa oikealla paikalla.

6.3 Asetusvalikot

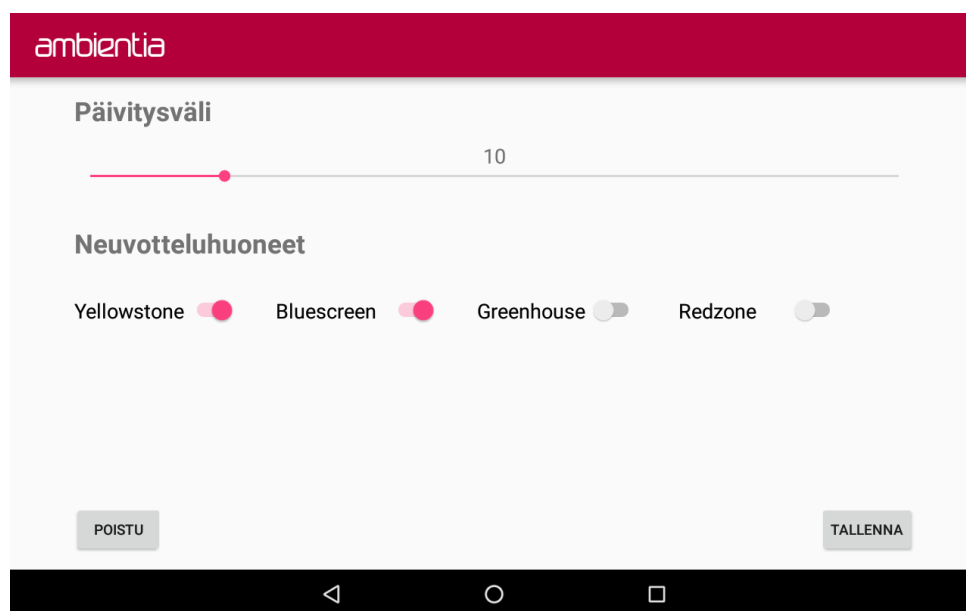
Sovelluksen toimintopalkkiin lisättiin Androidin oletus valikkokuvake, josta avautuu valinta, haluaako siirtyä näyttöasetuksiin vai kalenteriasetuksiin (Kuva 17). Näyttöasetuksissa valitaan kalenterinäkymän automaattisen päivityksen aikaväli ja määritetään kalenterinäkymässä näytettävät neuvotteluhuoneet. Kalenteriasetuksissa vuorostaan määritetään käytettävän kalenterin osoite ja sen käyttöön tarvittavat käyttäjätunnus ja salasana. Molemmille asetuksille luotiin omat aktiviteetit.



Kuva 17. Valikko avattuna toimintopalkissa.

6.3.1 Näyttöasetukset

Näyttöasetuksiin (Kuva 18) lisättiin SeekBar-elementti, jonka avulla valitaan haluttu päivitysväli kalenterinäkymän päivitykselle minuutin tarkkuudella. SeekBar voi saada arvoja 0:n ja 60:n väliltä, joista nolla asettaa automaattisen päivityksen pois käytöstä. Lisäksi näyttöasetukset sisältävät jokaiselle neuvotteluhuoneelle oman Switch-elementin, jolla määritetään, näkyykö kyseinen kalenteri kalenterinäkymässä. Näin kalenterinäkymään saadaan vain halutut kalenterit ja sovellusta voidaan käyttää esimerkiksi vain yhden neuvotteluhuoneen varausten seurantaan.



Kuva 18. Näyttöasetukset-aktiviteetti.

Tietojen tallentamista ja paluuta takaisin MainActivity-aktiviteettiin varten näkymässä on Tallenna- ja Poistu-painikkeet. Tallenna-painike tallentaa valinnat SharedPreferences-tietoihin, josta sovellus myös hakee sinne tallennetut tiedot näyttöasetusaktiviteetin onCreate-elinkaarimetodissa eli

aktiviteettia luodessa. Tietojen tallennuttua tästä myös ilmoitetaan käyttäjälle ilmoituksen avulla, kuten hyvän käyttöliittymän tulee tehdä.

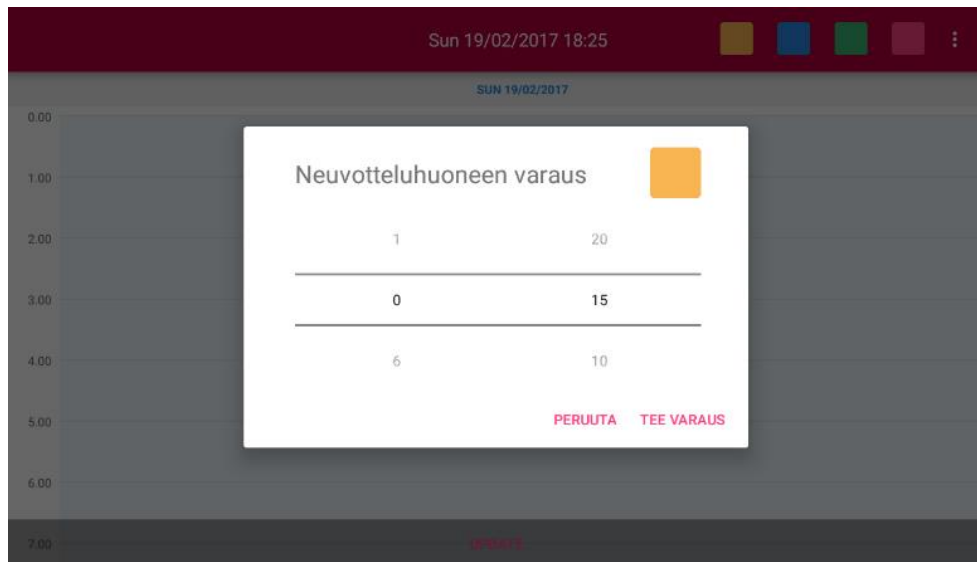
6.3.2 Kalenteriasetukset

Kalenteriasetuksia varten luotiin näyttöasetusten tavoin uusi aktiviteetti ja sinne lisättiin kolme tekstikenttää, joista yhdessä määritetään kalenterin URL-osoite, jota käytetään kalenterien varausten hakemiseen ja uusien varausten tekemiseen. Lisäksi kalenteriasetuksissa on tekstikentät käyttäjätunnukselle ja salasanelle, jotka vaaditaan kalenteritietojen hakemiseen (Kuva 19). Näkymään lisättiin Tallenna- ja Poistu-painikkeet, samaan kohtaan ja samoilla toiminnoilla kuin näyttöasetuksiin, eli Poistu-painike vie käyttäjän takaisin MainActivity-aktiviteettiin ja Tallenna-painike tallentaa tekstikenttiin asetetut tiedot SharedPreferences-tietoihin, ilmoittaen siitä käyttäjälle. URL-osoite tallennetaan normaalisti, mutta käyttäjänimi ja salasana enkoodataan Base64-menetelmällä ennen tallennusta.

Kuva 19. Kalenteriasetukset-aktiviteetti.

6.4 Pikavarausmahdollisuus

Pikavarausta varten luotiin varausdialogi, joka toteutettiin luomalla ReservationDialogFragment-fragmentti, joka tulee näkyviin painamalla liikennevaloista halutun neuvotteluhuoneen kuvakkeesta. Dialogissa (Kuva 20) on oikeassa yläreunassa näkyvillä sen neuvotteluhuoneen kuvake, johon varausta ollaan tekemässä, sekä NumberPicker-elementit tunneille ja minuuteille, joiden avulla määritetään varauksen kesto. Tunnit voidaan valita 0:n ja 8:n väliltä ja minuutit viiden välein 0:n ja 55:n väliltä. Lisäksi dialogissa on Peruuta- ja Tee varaus -painikkeet. Tee varaus -painike lähettää tiedot eteenpäin ReservationService-luokalle, josta itse varaus palvelimelle tehdään.



Kuva 20. Varausdialogi.

Kun varaus on tehty, näytetään käyttäjälle ilmoitus, jossa on näkyvissä varattu neuvotteluhuone, varauksen alkamisaika, sekä varauksen päättymisaika. Alkamisaika on se hetki, kun käyttäjä on painanut Tee varaus-painiketta ja päättymisaika valitut tunnit ja minuutit lisättyinä alkamisaikaan.

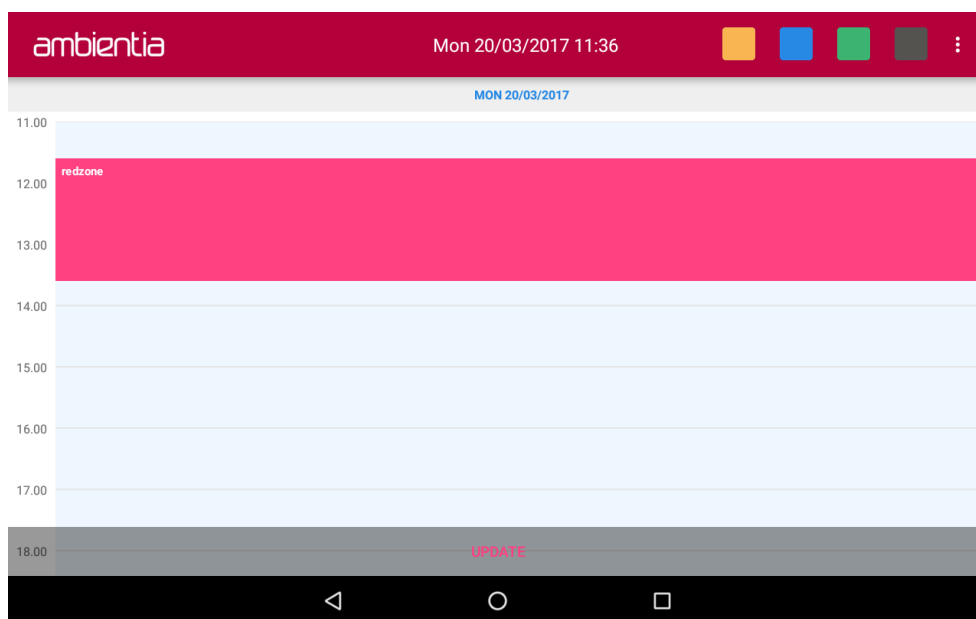
Varaus toteutetaan palvelimelle sen REST-rajapinnan avulla siten, että ensin otetaan kalenteriasetuksissa määritettyyn osoitteeseen yhteys ja suoritetaan tunnistautuminen asetuksissa määritetyin käyttäjätiedoin. Tämän jälkeen palvelimelle tulee lähettää POST-metodin avulla iCalendar-tiedosto, jonka avulla palvelin luo varauksen. iCalendar on Z Contentin kehittämä standardimenetelmä kalenteritietojen siirtämiseen järjestelmien välillä (iCalendar n.d.).

ReservationService-luokassa on metodi joka luo varausdialogissa valittujen tietojen mukaisen iCalendar-tiedoston (Kuva 21). Tiedosto luodaan hyödyntäen projektiin lisättyä biweekly-kirjastoa, joka on luotu iCalendar-tiedostojen tekemistä varten. Kun tiedosto on luotu, lähetetään se palvelimelle, jolloin palvelin luo tiedoston tietojen mukaisen varauksen.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Michael Angstadt//biweekly 0.6.1//EN
BEGIN:VEVENT
UID:0f8526b6-25e2-42a1-aa4e-358747ee6e47
DTSTAMP:20170320T113552Z
SUMMARY:Testivaraus
LOCATION:REDZONE
DTSTART:20170320T113552Z
DTEND:20170320T133552Z
END:VEVENT
END:VCALENDAR
```

Kuva 21. Esimerkki iCalendar-tiedostosta.

Kun varaus palvelimelle on tehty, päivittää sovellus kalenterinäkymän ja varausta voidaan tarkastella sovelluksen kalenterinäkymässä (Kuva 22). Kun neuvotteluhuone on sillä hetkellä varattuna, on myös sen neuvotteluhuoneen kuvake pimennettynä toimintopalkissa, jolloin kyseiselle neuvotteluhuoneelle ei voi luoda varausta. Varauksen loppuessa kuvake palautuu takaisin värilliseksi kalenterinäkymän päivityksen yhteydessä.



Kuva 22. Varaus sovelluksen kalenterinäkymässä.

7 YHTEENVETO

Opinnäytetyön tavoitteena oli lisätä neuvottelutilakalenterisovellukseen uusia toimintoja. Tavoiteltuja uusia toimintoja olivat kalenterinäkömän automaattinen päivitys, kaksi erilaista asetusvalikkoa, neuvottelutilan pika-varausmahdollisuus ja ajan ja päivämäärän näyttäminen toimintopalkissa. Kaikki toiminnot lisättiin onnistuneesti sovellukseen työn aikana, eikä ylitysepäsemättömiä ongelmia ilmaantunut.

Opinnäytetyössä saatiin vastaukset asetettuihin tutkimuskysymyksiin ja tutkimusongelmaan löydettiin ratkaisu. Samalla opinnäytetyön tekijä sai paljon kokemusta ja oppi kokonaan uusia asioita Android-sovelluskehityksestä, käyttöliittymäsuunnittelusta ja REST-ohjelmointirajapinnasta.

Työn jälkeen neuvottelutilakalenterisovellus, Neukkari-Seppo, tullaan asentamaan Ambientian Hämeenlinnan toimistoon kahdelle neuvottelutilojen läheisyyteen sijoitettavalle tabletille. Ambientian työntekijät voivat myös asentaa sovelluksen omille Android-laitteilleen.

LÄHTEET

Abeysinghe, S. (2008). *RESTful PHP Web Services*. Olton: Packt Publishing.

Android Developers. (n.d.a). Android, the world's most popular mobile platform. Haettu 17.1.2017 osoitteesta <https://developer.android.com/about/index.html>

Android Developers. (n.d.b). Dashboards. Haettu 24.1.2017 osoitteesta <https://developer.android.com/about/dashboards/index.html>

Android Developers. (n.d.c). App Manifest. Haettu 24.1.2017 osoitteesta <https://developer.android.com/guide/topics/manifest/manifest-intro.html>

Android Developers. (n.d.d). Introduction to Activities. Haettu 26.1.2017. <https://developer.android.com/guide/components/activities/intro-activities.html>

Android Developers. (n.d.e). Fragments. Haettu 26.1.2017 osoitteesta <https://developer.android.com/guide/components/fragments.html>

Android Developers. (n.d.f). The Activity Lifecycle. Haettu 26.1.2017 osoitteesta <https://developer.android.com/guide/components/activities/activity-lifecycle.html>

Android Developers. (n.d.g). Creating and Managing Virtual Devices. Haettu 19.1.2017 osoitteesta <https://developer.android.com/studio/run/managing-avds.html>

Android Developers. (n.d.h). Run Apps on the Android Emulator. Haettu 19.1.2017 osoitteesta <https://developer.android.com/studio/run/emulator.html>

Android Developers. (n.d.i). Services. Haettu 3.2.2017 osoitteesta <https://developer.android.com/guide/components/services.html>

Android Developers. (n.d.j). Saving Data. Haettu 16.2.2017 osoitteesta <https://developer.android.com/training/basics/data-storage/index.html>

Android Developers. (n.d.k). Saving Key-Value Sets. Haettu 16.2.2017 osoitteesta <https://developer.android.com/training/basics/data-storage/shared-preferences.html>

Android Developers. (n.d.l). Android Keystore System. Haettu 3.3.2017 osoitteesta <https://developer.android.com/training/articles/keystore.html>

- Cohen, R. & Wang, T. (2014). *GUI Design for Android Apps*. Apress.
- Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures. Chapter 5: Representational State Transfer*. Väitöskirja. Haettu 5.2.2017 osoitteesta https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- Galitz, W. (2007). *The Essential Guide to User Interface Design*. John Wiley & Sons.
- Gulabani, S. (2014). *Developing RESTful Web Services with Jersey 2.0*. Olton: Packt Publishing.
- iCalendar. (n.d.). Introduction. Haettu 16.3.2017 osoitteesta <https://icalendar.org/>
- Java. (n.d.) What is Java technology and why do I need it. Haettu 22.1.2017 osoitteesta https://www.java.com/en/download/faq/whatis_java.xml
- Kurniawan, B. & Perry, D. (2015). *Android Application Development: A Beginner's Tutorial*. Waterloo: Brainy Software.
- NetMarketShare. (n.d.). Mobile/Tablet Operating System Market Share. Haettu 22.1.2017 osoitteesta <https://www.netmarketshare.com/operating-system-market-share.aspx>
- Nolan, G. (2015). Where is the best place to store a password in your Android app. Haettu 3.3.2017 osoitteesta <http://www.androidauthority.com/where-is-the-best-place-to-store-a-password-in-your-android-app-597197/>
- Sandoval, J. (2009). *RESTful Java Web Services: Master Core REST Concepts and Create RESTful Web Service in Java*. Olton: Packt Publishing.
- Tutorialspoint. (n.d.). API Testing. Haettu 22.1.2017 osoitteesta https://www.tutorialspoint.com/software_testing_dictionary/api_testing.htm
- w3schools. (n.d.a.). XML tutorial. Haettu 16.3.2017 osoitteesta <https://www.w3schools.com/xml/>
- w3schools. (n.d.b.). JSON – Introduction. Haettu 17.3.2017 osoitteesta https://www.w3schools.com/js/js_json_intro.asp
- w3schools. (n.d.c.). JSON vs XML. Haettu 20.4.2017 osoitteesta https://www.w3schools.com/js/js_json_xml.asp