SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

ANDROID-SOVELLUKSEN KEHITYS

TEKIJÄ: Atte Ryynänen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Atte Ryynänen	
Työn nimi Android-sovelluksen kehitys	
Päiväys 20.5.2015	Sivumäärä/Liitteet 30/0
Ohjaaja(t) lehtori Jussi Koistinen, lehtori Sami Lahti	
Toimeksiantaja/Yhteistyökumppani(t) Suomalainen ohjelmistoyritys	

Tiivistelmä

Opinnäytetyön aiheena oli kehittää suomalaisen ohjelmistoyrityksen Android-pohjaista sovellusta. Sovellusta on kehitetty muutamien vuosien ajan, ja opinnäytetyön tarkoitus oli saada viimeisteltyä kyseinen sovellus. Sovellusta alettiin kehittämään uudelleen vanhan version päälle kesällä vuonna 2014, mutta sitä ei kuitenkaan ehditty saamaan valmiiksi sovitussa ajassa ja tarvitsi lisää jatkokehitystä, joten sovelluksen jatkokehityksestä saatiin hyvä aihe tälle opinnäytetyölle.

Sovellus on luotu ikäihmisten käyttöön ehkäisemään yksinäisyyttä, parantamaan yhteydenpitoa omaisiin sekä kotihoitajiin ja tuomaan kaikenlainen mahdollinen sisältö helposti yhden päätelaitteen alle. Sovellus on tarkoitettu käytettäväksi Android-tablettitietokoneille.

Työn aikana korjailtiin paljon ohjelmointivirheitä, paranneltiin ulkoasua, helpotettiin sovelluksen käyttöä, viimeisteltiin eri rajapinnat, otettiin käyttöön uusia Android-ominaisuuksia sekä selvitettiin ja suunniteltiin mahdollisesti tulevaisuudessa lisättäviä ominaisuuksia. Työ on ohjelmoitu Javalla, jonka apuna on käytetty erilaisia valmiita rajapintoja (esim. Facebookin ja YouTuben rajapinnat) ja Java-kirjastoja.

Lopputuloksena saatiin viimeistelty ja valmis versio sovelluksesta, joka on valmis käytettäväksi ja johon voidaan helposti jatkokehittää tulevaisuudessa jo työn aikana kehiteltyjä ominaisuuksia.

Avainsanat
Android, Java, SDK, API, rajapinta

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Atte Ryynänen			
Title of Thesis Android App development			
Date	20 May 2015	Pages/Appendices	30/0
Supervisor(s) Mr. Jussi Koistinen, Lecturer and Mr. Sami Lahti, Lecturer			
Client Organisation /Partners Finnish software company			

Abstract

The subject of this thesis was to develop further the company's Android application. The application has been under development for past few years and the purpose of this thesis was to finalize it. The old version of app was renewed in summer 2014, but it was not finished in time and the app needed more development, so it was a good subject for this thesis.

App has been created for senior citizens to prevent their feeling of loneliness, improve keeping in contact with their relatives and practical nurses and also to bring all kinds of contents to be easily used with one simple tablet. The application is meant to be used with Android tablets.

During the thesis, many of app's programming errors were fixed, the overall appearance was improved, the application usage was made easier, different APIs were finalized, and also potential features to be used in future were studied and designed. The app was programmed with Java, supported by different kind application programming interfaces (for example Facebook and Youtube APIs were used) and Java libraries.

The result of this thesis was a finalized and finished version of the application, which is ready for use and can be easily developed further with already designed features.

Keywords
Android, Java, SDK, API

ESIPUHE

Kiitos asiakasyrityksen toimitusjohtajalle ja kehitysjohtajalle mielenkiintoisesta opinnäytetyön aiheesta sekä kiitos Mastercom Oy:n toimitusjohtaja Antti Koposelle ja muulle henkilökunnalle neuvoista ja tuesta työn aikana. Lisäksi haluan kiittää ohjaavaa opettajaani, lehtori Jussi Koistista.

Kuopiossa 20.5.2015
Atte Ryynänen

SISÄLTÖ

TERMIT JA LYHENTEET

SDK	Software Development Kit, eli ohjelmistojen kehitykseen tarkoitettuja työkaluja, joiden avulla voidaan kehittää ohjelmistoja tietyille alustoille.
API	Application Programming Interface, eli ohjelmointirajapinta, jonka avulla ohjelmat voivat helposti keskustella keskenään ja liittää ne toisiinsa. API on usein osa jotakin SDK:ta.
Java	Oliopohjainen ohjelmointikieli, jota käytetään Android kehityksessä.
Android	Älypuhelimille suunniteltu mobiilialusta, joka käyttää Linuxia käyttöjärjestelmäytimenä. Ohjelmoidaan Java-kielellä.
Root	Linux-käyttöjärjestelmän pääkäyttäjä.
Roottaus / Roottaaminen	Prosessi, jolla murretaan puhelimen suojaukset jotta saadaan Root-käyttäjän (pääkäyttäjä) oikeudet.
Thread / Säie	Prosessi voi sisältää moni eri säikeitä, joilla voidaan tehdä monta eri asiaa yhtä aikaa, jumittamatta kuitenkaan muita toimintoja / ominaisuuksia.
IDE	Ohjelma, joka tarjoaa kaikki tarpeelliset toiminnot ohjelmoijalle ohjelmistojen kehittämiseen. Esimerkkejä IDE:istä: NetBeans, Visual Studio, Android Studio, Eclipse.
Manifest	Pakollinen tiedosto Android-sovelluksessa, joka pitää sisällään asetuksia ja tietoja.
Funktio	Funktioon voidaan lisätä koodia, joka voidaan helposti suorittaa kutsumalla kyseistä funktiota. Tämän avulla samojen koodien käyttö vähenee, ja samaa funktiota voidaan myös kutsua useita kertoja.

1 JOHDANTO

Tämän opinnäytetyön aiheena on jatkokehittää asiakasyrityksen Android-pohjaista sovellusta. Sovellus on suunniteltu ikäihmisten käyttöön ja sillä voidaan ehkäistä heidän yksinäisyyden tunnetta, helpottaa yhteydenpitoa omaisiin ja lähihoitajiin ja saada kaikenlainen sisältö helposti myös ikäihmisten ulottuville (esim. uutiset, omaisten kuvat, YouTube-videot, Facebook). Helpon käytön takaa selkeä ulkoasu ja isokokoinen HP Slate 21 Android tablettitietokone.

Opinnäytetyön tilaaja on suomalainen ohjelmistoyritys, joka on suunnitellut ja kehitellyt sovellusta jo muutamien vuosien ajan. Kesällä 2015 sovelluksen ulkonäköä uusittiin ja siihen lisättiin paljon uusia toimintoja, mutta sovellusta ei kuitenkaan saatu vielä täysin viimeisteltyä ja siitä puuttui toimintoja. Saman kaltaisia sovelluksia ei ole vielä monta markkinoilla, joten on mielenkiintoista nähdä sovelluksen kehittyminen ja suosio tulevaisuudessa.

Työn tavoitteena on saada toimiva kokonaisuus, joka voidaan viedä mahdollisimman monen ikäihmisen kotiin ja parantaa heidän terveyttään. Sovellus on ollut jo käytössä muutamilla henkilöillä, minkä pohjalta sovellukseen tehdä muutoksia ja parannuksia.

Nykypäivänä Android-pohjaisten sovellusten kysyntä on suuressa nousussa, minkä takia Android-ohjelmoinnin osaaminen on tärkeää. Android-käyttöjärjestelmät kehittyvät pikaista tahtia, ja sovellukset ovat entistä monipuolisempia ja monimutkaisempia. Tämän takia Android-sovelluksille on paljon kysyntää nyt ja tulevaisuudessa sekä Android-ohjelmoijille on varmasti tarvetta. Kehitystä myös edesauttaa kun kehittäjänä toimii yksi tietotekniikan isoimmista yrityksistä, Google.

Tässä opinnäytetyöraportissa esitellään sovelluksen toiminta, toteutetut toiminnot, eri rajapintojen toimivuus ja jatkokehitysideat. Raportissa käytetään koodia esimerkkeinä ja eritellään sovelluksessa olevat toiminnot. Lisäksi raportissa löytää kuvia sovelluksen ulkoasusta.

2 KÄYTETYT TEKNIIKAT

Tässä luvussa tarkastellaan opinnäytetyössä käytettyjä tekniikoita ja sovelluksia.

2.1 Android Studio

Android Studio on Googlen kehittämä IDE, joka on suunniteltu Android-ohjelmistojen tekoon, jonka on tarkoitus korvata aiemmin Android-ohjelmoinnissa käytetty Eclipse, jota taas käytetään yleisesti kaikenlaisten ohjelmistojen kehittämiseen. Android Studion kehitys aloitettiin vuonna 2013, ja sen ensimmäinen vakaa julkaisu ilmestyi vuoden 2014 joulukuussa, jolloin se myös hyväksyttiin Androidin viralliseksi IDE:ksi. Android Studiossa käytetään Gradle-pohjaista järjestelmien rakentamista ja uusimpiin versioihin on lisätty myös versionhallinta. Android Studio toimii Windowsilla, Mac OS X:llä ja Linuxilla. (Android Developer 2015a.)

2.2 Gradle

Gradle on Android Studioon lisätty liitännäinen, jolla voidaan hallita ohjelmiston riippuvuuksia ja määritellä kustomoidut asetukset järjestelmien rakentamiselle. Gradle toimii myös ilman Android Studiota, joten Android-ohjelmia voidaan kääntää myös ilman Android Studion asennustakin. (Android Developer 2015b.)

Gradle on hyvin kätevä ohjelmiston riippuvuuksien hallintaan, koska sillä voidaan helposti hakea ja ottaa käyttöön joko tietokoneelta tai ulkopuolisista tiedostovarastoista esim. Java-kirjastoja. Kaikki Gradlen asetukset ovat yhdessä tiedostossa, josta niitä on helppo muokata.

2.3 Android SDK

SDK sisältää paljon erilaisia työkaluja Android-ohjelmien kehitykseen. Nämä työkalut on jaettu kahteen ryhmään: SDK-työkalut ja platform-työkalut. SDK-työkalut ovat pakollisia, ja ne eivät ole alustariippuvaisia. SDK sisältää mm. seuraavia työkaluja:

- virtuaalisten laitteiden emulaattorin
- kehitystyökaluja:
    - koodin analysointi, SDK-pakettien hallinta, SQLite
- debug-työkaluja:
    - Android Debug Bridge, jolla voidaan keskustella emulaattorin tai Android-laitteen kanssa
    - Device Monitor, joka tarjoaa graafisen käyttöliittymän Android-ohjelmien debuggaukseen ja analysointiin
- build-työkalut:
    - APK-tiedostojen luonti, koodin optimointi.

Platform-työkalut tulevat SDK:n mukana, ja niitä ei yleensä tarvitse erikseen käyttää, koska SDK osaa automaattisesti kutsua niitä tarvittaessa. Yleisin platform-työkalu on logcat, joka näyttää ja kerää tietoja laitteen ja sovelluksen toiminnasta (esimerkiksi virheilmoitukset näytetään tässä). (Android Developer 2015c.)

2.4 Tortoise SVN

Tortoise SVN on Windowsille tehty versionhallintaohjelma, joka pohjautuu Apachen Subversion-ohjelmaan. Tortoise SVN tarjoaa helppokäyttöisen käyttöliittymän Subversionille, ja sitä voidaan käyttää minkä tahansa IDE:n versionhallintaohjelmana. (Tortoise SVN 2015.)

Versionhallintaohjelman avulla voin helposti ylläpitää eri projekteja, säilöä ja varmuuskopioida projektit palvelimelle ja työskennellä helposti useiden eri henkilöiden kanssa samassa projektissa.

2.5 Java

Java on yleiskäyttöinen, luokka- ja olio-pohjainen ohjelmointikieli. Se on suunniteltu hyvin yksinkertaiseksi, jotta mahdollisimman moni ohjelmoija pystyy helposti oppimaan sen. Java on yksi maailman suosituimmasta ohjelmointikielistä, ja sitä käytetään hyvin paljon kaikkialla. (Oracle 2015.)

2.6 JSON

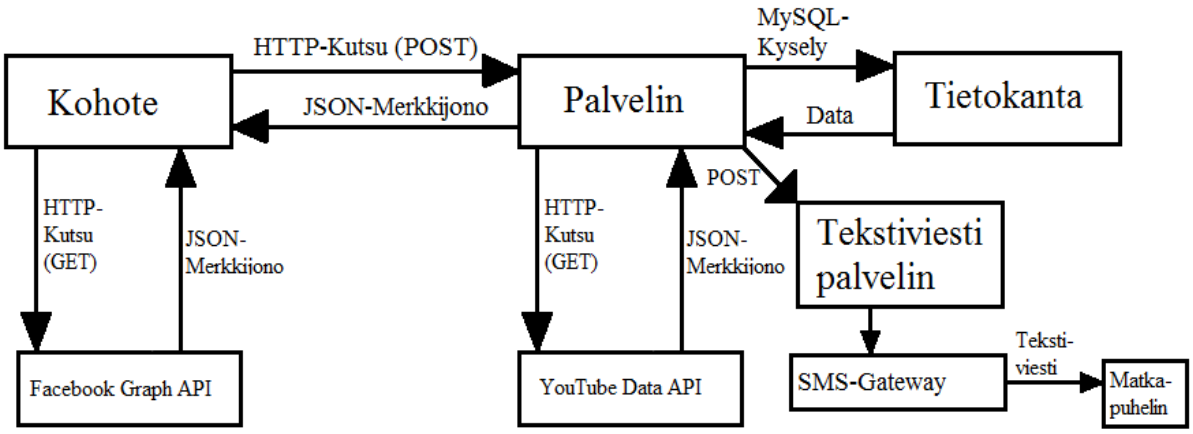
JSON eli JavaScript Object Notation on kevyt, tietojen siirtoon tarkoitettu formaatti. JSON on aivan tavallista tekstiä, ja siksi myös sitä on helppo lukea ja kirjoittaa. Sitä voidaan myös käyttää kaikissa mahdollisissa ohjelmointikielissä, mutta se tukee samoja ohjelmointikäytäntöjä kuten muutkin suositut ohjelmointikielet. (JSON 2015.)

3 TOIMINTA

Tässä luvussa tutkitaan tarkemmin sovelluksen yleistä toimintaa ja sitä, kuinka eri rajapinnat on toteutettu. Luvussa on myös kuvia sovelluksesta, joista nähdään, kuinka sovellus toimii käytännössä.

3.1 Ulkoinen toiminta

Sovelluksen ulkoinen toiminta koostuu yksinkertaisesta palvelinten ja laitteen välisistä kommunikoinnista.



KUVA 1. Sovelluksen ulkoinen toiminta (Atte Ryynänen 2015-05-12.)

Kuvassa 1 kuvataan sovelluksen eri rajapintojen ja palvelimien välinen toiminta. Suurin osa sovelluksen HTTP-kutsuista tehdään suoraan sovelluksen omalle palvelimelle, mutta Facebookin kutsut tehdään suoraan Facebookin palvelimille. Kaikki sovelluksesta lähtevä ja palvelimelta tuleva tieto näytetään JSON-muodossa, minkä vuoksi tietoa on helppo käsitellä.

Palvelimella on myös ulkoinen käyttöliittymä, jolla on helppo hallita eri käyttäjien asetuksia ja sisältöä. Käyttöliittymässä omaiset voivat jakaa kuvia, videoita (esim. Youtube-kanavat), RSS-syötteitä ja muuta sisältöä. Palvelin osaa automaattisesti muokata tämän sisällön Android-sovellukselle sopivaksi, joten Android-laitteessa ei tarvita muuta kuin muuttaa data oikeaan muotoon ja näyttää se sovelluksessa.

Tekstiviestien lähetyksessä käytettiin erillistä SMS-Gatewaytä, jonka avulla voidaan lähettää tekstiviestejä. Sovellus lähettää tekstiviestistä pyynnön palvelimelle, joka lähettää uuden pyynnön erilliselle tekstiviestipalvelimelle. Tekstiviestipalvelin vastaanottaa pyynnön ja lähettää parametrina tulleeseen puhelinnumeroon tekstiviestin.

Seuraavissa luvuissa eritellään tarkemmin eri osien toimivuutta.

3.2 JSON

Sovelluksessa kaikki mahdollinen tieto siirretään JSON:n avulla. Googlen tarjoaman Java-kirjaston, GSON, ansiosta voidaan helposti muuttaa JSON-merkkijono suoraan Java-objekteiksi ja objekteja JSON:ksi.

```
[{
    "time":1450908000,
    "name":"Jouluaatto",
    "holiday":false,
    "flagday":false,
    "celebrated_names":["Aatami","Eeva","Eevi","Eveliina"]
}]
```

KUVA 2. Esimerkki JSON-merkkijonosta (Atte Ryynänen 2015-05-12.)

Kuten kuvasta 2 huomataan, JSON on hyvin helppolukuista. Aaltosuluilla merkitty alue tarkoittaa yhtä objektia ja hakasulut tarkoittavat taulukkoa. JSON:ssa muuttujan nimi on ensimmäisissä lainausmerkeissä oleva merkkijono (”name”), jonka arvo on eroteltu kaksoispisteellä (”Jouluaatto”). Pilkku erottaa parit aina toisistaan ja yhdellä muuttujalla on yksi arvo (arvo voi olla myös null eli tyhjä).

```
import com.google.gson.annotations.SerializedName;

public class DayInfo {

    @SerializedName("time")
    public String Timestamp;

    @SerializedName("name")
    public String HolidayName;

    @SerializedName("holiday")
    public boolean Holiday;

    @SerializedName("flagday")
    public boolean Flagday;

    @SerializedName("celebrated_names")
    public String[] CelebratedNames;

}
```

KUVA 3. JSON-objekti (Atte Ryynänen 2015-05-12.)

JSON-merkkijonosta voidaan helposti luoda objekti GSON-kirjaston avulla. Luokkaa tutkimalla nähdään, että jokaista muuttujaa edeltää @SerializedName(”teksti”), joka kertoo, minkä parin arvo tulee mihinkin muuttujaan, eli esimerkiksi HolidayName muuttujaan tulisi ”Jouluaatto”.

Eli JSON:sta voidaan luoda objekti vaikkapa nimellä dayInfo (DayInfo dayInfo) ja kyseisen objektin tietoihin päästään käsiksi muuttujan avulla (dayInfo.HolidayName), jolloin voidaan esimerkiksi tulostaa konsoliin ”Jouluaatto” -teksti.

3.3 HTTP-Kutsut

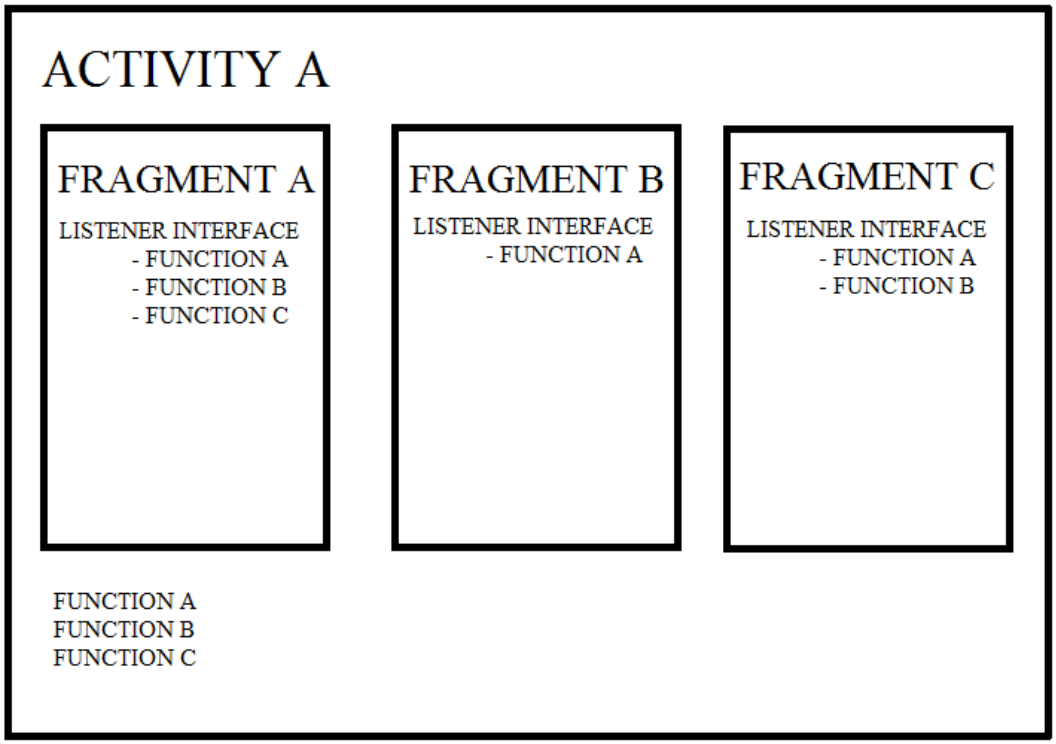
HTTP-kutsuilla tarkoitetaan käytännössä samaa asiaa kuin nettisivun avaamista selaimeen. Kun palvelimelle lähetetään kutsu ja palvelin palauttaa vastauksen, selain osaa käsitellä vastauksen ja näyttää nettisivuna. Kutsuihin voidaan myös antaa parametreja (POST ja GET), joiden mukana voidaan lähettää tietoa palvelimelle.

Sovelluksessa kutsuja käytetään, kun halutaan saada tietoa palvelimelta, esimerkiksi kun halutaan ladata ja näyttää uutisten RSS-syöte sovelluksessa. Tietoa hakiessa kutsutaan tiettyä URLia oikeilla POST-parametreilla, minkä jälkeen (jos kutsu onnistui) palvelin katsoo parametrit läpi ja palauttaa halutut tiedot vastauksena. Koska kutsua ei tehty selaimella, laitetta käytettäessä vastausta ei nähdä missään, mutta koodi osaa käsitellä vastauksen oikein ja näyttää uutiset laitteessa.

Kaikki kutsut toimivat taustalla, ja ne tapahtuvat eri säikeessä kuin missä käyttöliittymä pyörii, joten ne eivät jumita laitetta. Kutsut ovat myös asynkronisia eli niitä ei lähetetä missään tietyssä järjestyksissä eikä yhtä aikaa, vaan jokainen kutsu on oma kutsunsa.

3.4 Sisäinen toiminta

Sovelluksen sisäinen toiminta on toteutettu FragmentActivityjen avulla.



KUVA 4. Sisäinen toiminta (Atte Ryynänen 2015-05-12.)

Kuvasta 4 huomataan, kuinka sovellus on ohjelmoitu käyttäen FragmentActivityjä. Activityä voidaan miettiä yhtenä kokonaisuutena, jonka alla on Fragmenteja. Fragmenteilla ja Activitylla on omat ulkoasunsa ja Fragmentteja on helppo vaihtaa ja avata Activityn sisällä.

Activityn funktioita voidaan kutsua suoraan Fragmentista, tässä olevilla Interface Listenereiden avulla. Activity implementoi Fragmentin Listenerin ja yli kirjoittaa siinä olevan funktion. Tämä helpottaa koodin lukua ja vähentää turhaa, useasti toistuvaa koodia.

Fragmentteja on käytetty todella paljon sovelluksessa, koska niillä on helppoa ylläpitää sovelluksen toimivuutta ja eri Fragmentteja voidaan käyttää eri paikoissa, joten samaa asiaa ei tarvitse koodata useampaa kertaa.

Sovellus käyttää yhtä pää-Activityä, jonka sisälle kaikki Fragmentit avataan. Activityllä on oma ulkoasunsa, johon on määritelty alue, mihin avataan uudet Fragmentit, kun esimerkiksi sivua vaihdetaan sovelluksessa. Ulkoasussa on myös paljon staattisia elementtejä, jotka eivät muutu. Näin ei jouduta luomaan aina kokonaan uutta ulkoasua, vaan Fragmentteja voidaan ylläpitää helposti yhden alueen sisällä, johon voidaan nopeasti ladata kaikki käyttäjälle näytettävät tiedot.

Android-sovelluksia kehitetään yleisesti ActivityFragmenteilla, niiden helppokäyttöisyyden ja selkeyden takia.

3.5 Roottaus

Yksi tämän opinnäytetyön osuuksista oli tutkia, kuinka HP Slate 21 voidaan onnistuneesti rootata eli murtaa puhelimen suojaukset, jotta pääkäyttäjän tunnukset saadaan käyttöön. Koska Android käyttää Linux-käyttöjärjestelmäydintä, laite voidaan murtaa ja ottaa pääkäyttäjän tunnukset käyttöön. Roottaus ei ole laitonta Euroopassa, eikä se myöskään mitätöi laitteen takuuta ellei myyjä voi todeta, että roottaus on vahingoittanut laitetta (Šuklje ja Piana 2012-11-06). Maailmalla on kiistelty pitäisikö roottauksen olla laitonta tai pitäisikö sen mitätöidä takuu. Roottauksen pitäisi olla mahdollista ja laillista, niin kauan kun se on käyttäjän vastuulla ja käyttäjä ymmärtää riskit.

TAULUKKO 1. Roottauksen hyvät ja huonot puolet (Phelps 2015.)

Hyvät puolet	Huonot puolet
Epävirallisten käyttöjärjestelmäversioiden asennus	Tietoturvariskit ja virukset
Erikoisohjelmien käyttäminen. Jotkut ohjelmat ja puhelimen toiminnot vaativat root-oikeudet toimiakseen	Joissakin maissa takuu mitätöityy
Tilan vapauttaminen. Ohjelmat on mahdollista siirtää puhelimessa olevalle ulkoiselle muistikortille	Laitteen hajoaminen kokonaan
	Laite saattaa hidastua roottauksen jälkeen
	Viralliset päivitykset eivät välttämättä enää toimi rootatussa laitteessa

Root-sana tulee UNIX-käyttöjärjestelmien pääkäyttäjän nimestä. Tällä pääkäyttäjällä on täydet oikeudet kaikkiin toimenpiteisiin järjestelmän sisällä. Voidaan ajatella, että roottaamattomassa

älypuhelimessa on vain olemassa ”vieras” käyttäjä, joka ei pääse käsiksi kaikkiin järjestelmän tiedostoihin, kun taas root-käyttäjällä on täydet oikeudet.

3.5.1 Roottaus käytännössä

Internetistä löytyy paljon ohjeita ja tietoa kuinka eri laitteita pystytään roottamaan. Rootatessa kannattaa aina kuitenkin muistaa että se voi vahingoittaa laitetta väärintehtynä ja roottauksen yhteydessä asennetaan kolmannen osapuolen (ei virallisia) käyttöjärjestelmiä, joissa voi olla tietoturvariskejä. Ohjeet vaihtelevat eri laitteilla, joten ei ole olemassa mitään yleistä ohjetta eikä käyttöjärjestelmää joka toimii kaikissa laitteissa.

Ennen kuin laite voidaan rootata se pitää ensin unlockata, joka tyhjentää ja asentaa laitteen käyttöjärjestelmän uusiksi. Tämä on välttämätöntä ennen kuin roottaus on mahdollista. Tämän jälkeen kolmannen osapuolen järjestelmä voidaan asentaa ja pääkäyttäjä saadaan käyttöön.

3.5.2 Root-komennot

Android ei salli normaalisti laitteen alapalkin piilottamista pysyvästi, mutta rootatulla laitteella tämä onnistuu ja siksi opinnäytetyön aikana piti selvittää kuinka Slate 21 voidaan rootata. Tulevaisuudessa sovellukseen saatetaan tulla lisäämään muitakin root-oikeuksia vaativia toimintoja.

```
1  Runtime.getRuntime().exec("service call activity 42 s16 com.android.systemui")
```

KUVA 5. Root-komento navigointivalikon piilottamiseen (Atte Ryynänen 2015-05-12.)

Root-oikeudet mahdollistavat Android-shellin komentojen suorittamisen suoraan koodista tai erikseen ladatulla terminal-ohjelmalla. Kuvasta nähdään kuinka shell-komentoja voidaan suorittaa suoraan koodista. Tämä komento piilottaa Androidissa alhaalla olevan navigointivalikon, joka on mahdoton piilottaa kokonaan normaalisti ilman root-oikeuksia.

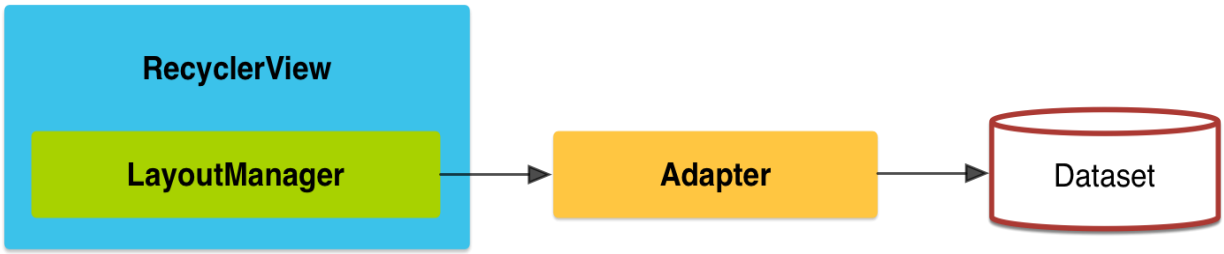
3.6 RecyclerView

Uusimmassa Android-versiossa (Lollipop) yhtenä uutena ominaisuutena tuli RecyclerView, jonka tarkoitus on korvata vanha ListView näkymä. Uusi RecyclerView tarjoaa monipuolisempi tapoja näyttää listoja ja helpottaa joidenkin asioiden toteuttamista. Esimerkiksi ListView’llä horisontaalinen lista oli todella hankala toteuttaa ilman erillisiä kirjastoja, kun taas uuteen RecyclerView’iin se on tehty valmiiksi. RecyclerView’llä on muutamia valmiita pohjia listoille:

- Lineaarinen eli LinearLayout
- Ruudukko eli GridLayout
- Porrastettu ruudukko eli StaggeredGridLayout

Nämä kaikki pohjat toimivat pysty- sekä vaakasuunnassa. Suurin osa sovelluksen listoista korvattiin RecyclerView’llä ja kaikkia valmiita pohjia on käytetty eri listoissa. Lollipop-versiossa tuli myös

mukana uusi CardView, jolla voidaan helposti luoda kortin näköisiä ulkoasuja listaan. Muutamissa listoissa on käytetty tätä.



KUVA 6. RecyclerView (Android Developer 2015d.)

Tarvitaan siis kolme eri asiaa RecyclerView'n toteukseen:

- Fragmentin, jossa luodaan ja näytetään RecyclerView
- Datasetin eli jotakin ainestoa mitä halutaan näyttää listassa
- Adapterin, jossa määritellään mm. seuraavat asiat:
    - yhden listan rivin ulkoasun
    - mahdolliset listan Eventit (eli esimerkiksi mitä tapahtuu kun listan riviä painetaan)
    - minkälaisen aineiston Adapter voi ottaa vastaan

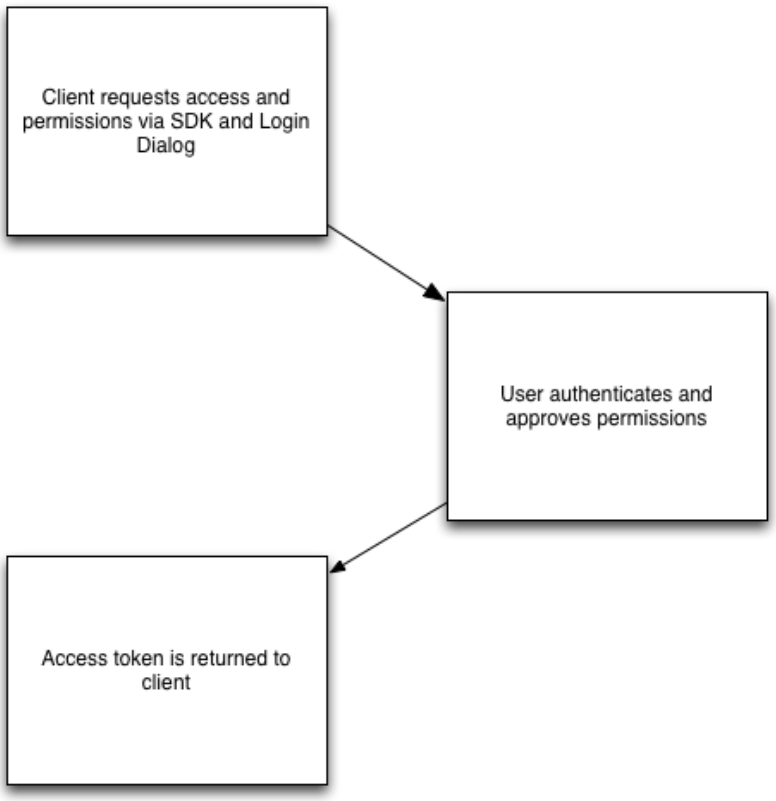
Adapterin avulla voidaan siis "yhdistää" Dataset johonkin ulkoasuun, joka näytetään RecyclerView'ssä.

3.7 Facebook Android SDK 4.0 ja Graph API 2.3

Facebook tarjoaa monille eri alustoille valmiita kirjastoja ja rajapintoja, joiden avulla on helppo integroida Facebookin tarjoamia toimintoja sovelluksiin. SDK:n mukana tulevat kirjastot Graph API:n kutsuihin ja Facebookin kirjautumiseen. SDK myös sisältää Facebookista otetut värit, teemat yms. ulkoasuun liittyvät asiat. Tässä luvussa käydään läpi SDK:n toimintoja ja Graph-rajapinnan toimintaa

3.7.1 Käyttöoikeustunnus

Facebook käyttää tunnistamiseen käyttöoikeustunnusta eli Access tokenia. Näitä tunnuksia on muutamia erilaisia, mutta yleisin on käyttäjäkäyttöoikeustunnus eli User Access Token, jota käytetään tässä sovelluksessa. Käyttöoikeustunnus luodaan yleensä kirjautumisen yhteydessä, ja sitä voidaan käyttää Graph API:n kyselyissä. Ilman käyttöoikeustunnusta kyselyitä ei voida tehdä. Toisin sanottuna tarvitaan käyttöoikeustunnus, jotta Facebook tunnistaa kyselyn oikeaksi ja osaa palauttaa oikeita tietoja.



KUVA 7. Käyttöoikeustunnusten luonti (Facebook 2015.)

Käyttöoikeustunnukset vanhentuvat tietyn ajan kuluttua, mutta ne voidaan uusia koodin avulla. Näitä tunnuksia voidaan myös käyttää muissa sovelluksissa. Jos tunnukset haetaan sovelluksessa, voidaan käyttää samaa tunnusta myös selaimesta tehdyissä kutsuissa, vaikka selaimessa ei olisikaan erikseen kirjauduttu sisään. Käyttöoikeustunnus myös määrittelee mitä kaikkea sovellus pystyy Graph API:sta hakemaan. Eli jos halutaan hakea käyttäjän Facebook seinä, sovellus pyytää oikeudet käyttäjän seinän lataamiseen. Näin käyttäjä tietää aina mitä oikeuksia tarvitaan ja mitä tietoja sovellus lataa Facebookista.

3.7.2 Graph API 2.3

Facebook kehitystyökaluilla voidaan suoraan tehdä ja testata Graph API-kutsuja. Kuvassa on kutsuttu käyttäjän tietoja ”me” parametrin avulla. Vastauksena saadaan JSON-muodossa profiilitiedot.

KUVA 8. Graph API:sta on haettu käyttäjän profiili JSON-muodossa. (Atte Ryynänen 2015-05-12.)

3.7.3 Facebook Android SDK 4.0

SDK:ssa on saatavilla valmiina Facebookiin kirjautuminen, eli kirjautuminen voidaan lisätä suoraan Fragmentin ulkoasuun. Kirjautumisen onnistuttua saadaan vastauksena käyttöoikeustunnus ja käyttäjän profiili (profiili sisältää perustietoja käyttäjästä, esimerkiksi nimen). Sovellus palaa takaisin Fragmentiin, ja koska käyttöoikeustunnus saatiin käyttöön, Facebookista voidaan alkaa hakea tietoa suoraan Graph API:lta.

```
GraphRequest request = GraphRequest.newMeRequest(
        accessToken,
        new GraphRequest.GraphJSONObjectCallback() {
            @Override
            public void onCompleted(
                    JSONObject object,
                    GraphResponse response) {

            }
        });
Bundle parameters = new Bundle();
parameters.putString("fields", "id,name,link");
request.setParameters(parameters);
request.executeAsync();
```

KUVA 9. Graph API:n kutsu Androidista (Atte Ryynänen 2015-05-12.)

Kuvassa 9 on esimerkki, kuinka voidaan luoda uusi GraphRequest-objekti, lisätä siihen parametreja ja toteuttaa pyyntö. Pyynnössä lähetetään käyttöoikeustunnus ja saadaan vastauksena GraphResponse-objekti. Pyynnön valmistuttua onCompleted()-funktiota kutsutaan automaattisesti ja funktion sisältämät koodit ajetaan.

GraphResponse on SDK:n luokka, missä tulee kaikki Graph API:n palauttamat tiedot, kuten käyttäjän hakemat tiedot JSON-muodossa ja mahdolliset virheviestit. JSON voidaan helposti muuttaa Java-objekteiksi GSON-kirjaston avulla (katso luku 2.1). Tämä tieto voidaan sitten näyttää halutulla tavalla sovelluksessa.

```
GraphRequestBatch graphRequestBatch = new GraphRequestBatch(); //Luodaan uusi pyyntöjoukko

GraphRequest request = GraphRequest.newMeRequest( ... ); // Luodaan uusi pyyntö

graphRequestBatch.add(request); //Lisätään pyyntö joukkoon

batch.addCallback(new GraphRequestBatch.Callback() {
    @Override
    public void onBatchCompleted(GraphRequestBatch graphRequests) {
        // Toteuttavat koodit pyyntöjoukko kutsun valmistuttua.
    }
});
batch.executeAsync();
```

KUVA 10. Pyyntöjoukon suorittaminen (Atte Ryynänen 2015-05-12.)

4.0-versiossa myös mukana tuli uusi ominaisuus suorittaa useampi pyyntö yhtä aikaa. Tällä tavoin pyyntöjä ei tarvitse tehdä yksitellen, vaan voidaan lähettää kaikki tiedot yhtenä joukkona.

Aluksi luodaan pyyntöjoukko (GraphRequestBatch), johon lisätään pyyntöjä (Pyyntö on samanlainen kuin kuvassa 11). Pyyntöjoukolla on myös onBatchCompleted()-funktio. joka toimii samoin kuin yksittäisen pyynnön onCompleted()-funktio. Yksittäisten pyyntöjen onCompleted()-funktiot toimivat normaalisti, vaikka ne tehdäänkin yhdessä erässä.

Kirjautumisen onnistuttua Graph API:n palauttama JSON muutetaan objekteiksi ja näytetään listassa Adapterin avulla.

3.8 Äänensäätönapit

Yhtenä toiveena oli lisätä sovellukseen napit, joiden avulla on helppo säätää äänenvoimakkuutta. Vaikka Slatessa itsessään on napit tähän tarkoitukseen niitä on hieman hankala käyttää.

Androidissa on valmiina kirjastot ja funktiot, joiden avulla voidaan helposti säätää äänenvoimakkuutta. Ääntä voidaan säätää painamalla nappia kerran tai pitämällä pohjassa sovelluksessa olevaa kaiuttimen kuvaa. Napit toimivat yksinkertaisilla OnClickListener'llä ja OnTouchListener'llä, joiden avulla voidaan määrittää mitä koodia ajetaan missäkin tilanteessa.

OnTouchListener'ssä oli ongelmana että siinä ei voida yksinkertaisesti kutsua jatkuvasti äänensäätö funktiota, vaan tähän piti lisätä Handler. Handler'n avulla voidaan luoda uusi säie, joka toistuvasti tietyn väliajoin suorittaa koodia. Uudessa säikeessä kutsutaan äänensäätö funktiota niin kauan kun käyttäjä painaa kaiutinta.

Eli kun käyttäjä painaa kaiutin nappia → luodaan uusi Handler ja säie → säie kutsuu jatkuvasti funktiota → käyttäjä lopettaa kaiuttimen painamisen → sovellus tunnistaa Eventiksi, jossa Handler ja säie tuhotaan.

Nappeja painamalla myös ruudulle ilmestyy Androidin oma äänenohjausvalikko, jonka avulla voidaan helposti säätää äänenvoimakkuutta.

3.9 Receiver

Receiver'n avulla voidaan luoda ohjelmiston komponentteja, jotka mahdollistavat koodin suorittamista kun Android tai jokin muu sovellus lähettää käskyjä. Receiver asetetaan kuuntelemaan tiettyjä käskyjä, jolloin kun sovellus saa käskyn voidaan suorittaa koodia vaikka itse sovelluksen muut osat eivät olisi käynnissä. (Android Developer 2015e.)

Androidissa on monia erilaisia tälläisiä käskyjä, joiden avulla voidaan suorittaa koodia eri tilanteissa. Esimerkiksi käyttäjä haluaa jakaa kuvan laitteesta, voidaan asettaa sovellus huomaamaan että käyttäjä haluaa jakaa kuvan, milloin Android tarjoaa kyseistä sovellusta kuvan jakoon. Receiver'n

avulla voidaan käynnistää sovellus heti kun laite on käynnistynyt. Tämä onnistuu kun Receiver luodaan, mikä käynnistää sovelluksen ja lisäämällä sovelluksen Manifestiin asetuksen, joka kuuntelee laitteen käynnistymistä.

Receiverin ja roottauksen avulla voidaan varmistaa että sovellus on kokoajan päällä. Sovelluksesta toki voidaan poistua jos sovellus asetetaan huoltotilaan, mutta oletusarvona sovellus aukeaa automaattisesti ja siitä on mahdoton poistua.

3.10 Näyttökuvat

Tässä luvussa tarkastellaan sovelluksen näyttökuvia sekä lisäksi selvitetään, kuinka viime luvuissa käydyt asiat on toteutettu käytännössä.

3.10.1 Etusivu

Sovelluksen etusivu toimii yleissivuna, joka sisältää mm. RSS-syötteen.

KUVA 11. Sovelluksen etusivu

Kaikki sivut koostuvat kahdesta eri ulkoasusta: Activityn ulkoasusta ja Fragmentin ulkoasusta. Activityn ulkoasu on koko ruudun kokoinen, ja siinä on valkoinen tausta. Vasemmalla puolella on valikko, ja Activityn keskelle avataan aina sivujen omat Fragmentit.

Vasemmassa yläkulmassa näkyy kello ja päivämäärä. Nurkassa on myös ruksi, jota klikkaamalla sovelluksen päälle aukeaa musta-taustainen ruutu, mikä tarkoittaa, että sovellus on ”nukkumatilassa”. Tämä tila myös avataan automaattisesti muutaman minuutin kuluttua. Tilan tarkoituksena on tummentaa ruutua ja himmentää siitä tulevaa valoa.

Vasemmassa reunassa on valikko, jolla voidaan vaihtaa sivua. Sivu aukaisee aina uuden Fragmentin keskelle ja sulkee entisen. Tässä huomataan, kuinka FragmentActivityt toimivat käytännössä. Sovelluksessa Activityä ei ladata ikinä uudelleen vaan vain Fragmentit vaihtuvat ja ladataan uudelleen.

Valikon alla on äänensäätönapit, joilla on helppo säätää ääntä (Katso kappale 3.8). Vasemmanpuoleinen nappi hiljentää ja oikeanpuoleinen lisää ääntä.

Valikoiden alla on hälytyskello, jolla käyttäjä voi hälyttää hoitajan. Hoitajalle lähtee tekstiviesti, jossa pyydetään ottamaan yhteyttä. Tämä lisää ikäihmiselle turvaa, koska hädän sattuessa voidaan helposti hälyttää apua paikalle. Hälytyksessä ei ole mitään varmistusta, vaan se lähtee muutaman sekunnin kuluttua, kun nappia painetaan. Hälytys kuitenkin voidaan peruttaa tämän muutaman sekunnin sisällä painamalla ”peruuta”-nappia ruudulle aukeavasta viestistä.

Edellä mainitut asiat ovat Activityn osia ja ne ovat aina näkyvillä, vain keskellä oleva sisältö muuttuu. Etusivun Fragmentissa näkyy uutissyöte, nimipäivät ja juhlapäivät. Myös omaisiin voidaan ottaa yhteyttä etusivulta.

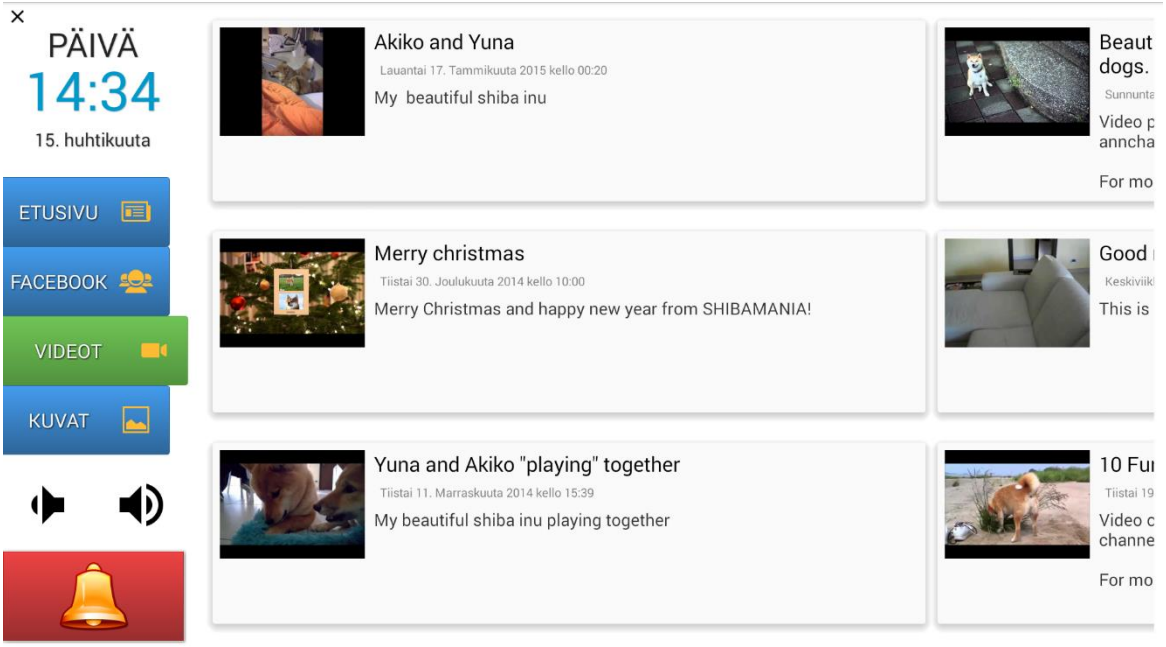
Uutisissa on käytetty porrastettua ruudukkolistaa ja sitä selataan vertikaalisesti. Uutiset on toteutettu CardView’llä, josta tulee uutisten korttimainen ulkoasu. Kortit ovat nopea tapa tehdä tyylikkäitä ja nykyaikaisia listoja, joilla on muutamia helposti muutettavia ominaisuuksia, kuten varjot.

Uutista klikkaamalla avataan uusi Fragment, joka sisältää Androidin oman selaimen. Selaimessa näytetään itse uutinen, eli jokaisessa kortissa on vain suora linkki uutiseen, joka avataan selaimeen. Nimipäivät ja juhlapäivät tulevat yksinkertaisesti palvelimelta, jotka näytetään tekstikentässä. Jos juhlapäivää ei ole, kyseinen teksti on piilossa.

Oikeassa reunassa on lista omaisista, joille käyttäjä voi lähettää yhteydenottopyynnön. Yhteydenotto toimii täysin samalla tavalla kuin hälyttäminen, mutta kontaktit näkyvät ainoastaan etusivulla kun taas hälytyskello näkyy jokaisella sivulla. Kontakteille on mahdollista antaa myös kuvat, jotka näkyvät nimen yläpuolella.

3.10.2 Videot

Videoiden listauksessa on käytetty horisontaalista ruudukkolistaa, minkä ulkoasuna myös käytetään CardView'tä. Videot on toteutettu horisontaalisesti, koska näin voidaan myös selata videon kuvausta. Lisäksi tämä toteutus säästää tilaa.



KUVA 12. Sovelluksen videot

Videot tulevat YouTube-kanavilta (myös YouTube streamit näkyvät listassa). Jokainen kortti sisältää videon otsikon, päivämäärän, kuvauksen ja kuvan videosta. Nämä kaikki tulevat sovelluksen palvelimelta, joka lataa ne YouTube-rajapinnan avulla YouTubesta. Videoita ei ladata valmiiksi, vaan ne katsotaan Androidin omalla YouTube-rajapinnalla.

Videota klikkaamalla ne avautuvat YouTuben-rajapinnan avulla omaan videosoittimeen. Videosoitimelle riittää, että annetaan videon ID:n (tunnistemerkkijono, joka videolla on omansa), ja rajapinta lataa kyseisen videon tai streamin uuteen Fragmenttiin.

3.10.3 Kuvat

Kuvat-sivu on toteutettu lineaarisella RecyclerView'llä ja ViewPager'lla.

KUVA 13. Sovelluksen kuvat-sivu

Ylhäällä olevat pienet kuvat on toteutettu RecyclerView'llä ja keskellä oleva kuva on tehty ViewPager'illa. ViewPager'lla on mahdollista selata erilaista tietoa helposti pyyhkäisemällä oikealle tai vasemmalle. Kuvien selaus siis onnistuu joko painamalla ja selaamalla ylhäällä olevia kuvia tai pyyhkäisemällä keskellä olevaa kuvaa. Kuva voidaan suurentaa ruudulla painamalla keskellä olevaa kuvaa. Sovellus alkaa selaamaan kuvia automaattisesti, jos laitteella ei tehdä mitään vähään aikaan. Kuvat ladataan laitteen muistiin, joten niitä ei ladata aina uudelleen, kun tämä sivu avataan.

3.10.4 Facebook

Facebook-seinä on toistaiseksi toteutettu vanhalla ListView'llä. Seinälle ladataan päivityksiä Facebookista samalla tavalla, kuin käytettäisiin Facebookin sivuja.

KUVA 14. Sovelluksen Facebook

Rivin vasemmalla puolella näkyy päivityksen tekijännimi ja profiilikuva. Oikealla puolella näkyy itse päivitys. Päivityksessä näkyy tietoa siitä, mitä päivityksen tekijä on tehnyt (esimerkiksi käyttäjä X tykkää käyttäjän Y:n kuvasta), päiväys, tykkääjät ja sisältö. Sisällössä voi olla tekstiä, kuvia (yksittäinen tai albumi) tai videoita.

Päivitystä painamalla sovellus avaa kuvan isompaan ruutuun tai mikäli päivityksessä on video, avataan video uuteen Fragmenttiin ja näytetään se videosoittimessa. Mikäli päivityksessä on useampi kuin yksi kuva, kuvia klikkaamalla voidaan avata ne isommiksi.

Videota katsottaessa käytetään Androidin omaa videosoitinta, jonka näppäimien ulkoasu on kuitenkin koodattu uusiksi, koska ne olivat perinteissä videosoittimessa liian pienet. Uudet näppäimet ovat isokokoisempia ja selvempiä.

3.11 Käännökset

Sovelluksen tulevia käännöksiä varten kaikki koodien merkkijonot muutettiin lukemaan merkkijonot suoraan strings.xml-tiedostoista. Näin voidaan luoda jokaiselle käännökselle oma strings.xml-tiedostonsa, jonka avulla voidaan helposti ylläpitää sovelluksen käännöksiä.

Androidiin on luotu jo valmiiksi hyvät pohjat käännöksille, koska Android tunnistaa automaattisesti laitteessa käytössä olevan kielen ja osaa vaihtaa myös sovellusten kielen samalla, jos sovellukseen

on luotu strings.xml tiedosto kyseiselle kielelle. Kansiot, joissa strings.xml-tiedosto on, täytyy nimetä tietyllä tavalla. Esimerkiksi jos halutaan lisätä ruotsin kieli, kansion nimeksi tulisi values-sv.

Käännöksiä tehdessä täytyy ottaa huomioon sovelluksen ulkoasu. Eri kielten käännösten merkkijonot ovat yleensä eripituisia, joten ulkoasu ei välttämättä näytä samalta kaikissa kielissä. Siksi ulkoasu täytyy suunnitella mahdollisimman dynaamisiksi tai mahdollisesti ulkoasut rakennetaan valmiiksi eri kielille, mitkä ladataan valitun kielen perusteella.

4 JATKOKEHITYS

4.1 Pelit

Yksi toivotuimmista ominaisuuksista on ollut pelien lisääminen sovellukseen. Lisäyksessä on kuitenkin ongelmia, jotka pitäisi pystyä ratkaisemaan. Koska sovellus on tarkoitettu suljetuksi järjestelmäksi, pelejä täytyisi pystyä käynnistämään tai pelaamaan suoraan sovelluksen sisältä.

- Pelejä täytyisi pystyä helposti jakamaan kaikille käyttäjille.
- Tietyille käyttäjille pitää pystyä aktivoimaan tietyt pelit.
- Mahdolliset pelit täytyisi ladata laitteelle järkevästi.
- Pelejä ei voida suoraan käynnistää sovelluksesta.

Ensimmäisenä täytyy ratkaista tapa, jolla pelit tehdään: Voidaanko ne ohjelmoida suoraan selaimella toimiviksi vai pitääkö ne tehdä erillisinä sovelluksina. Androidille on mahdollista luoda selainpohjaisia pelejä käyttämällä esimerkiksi HTML5:sta. Selainpohjaiset pelit ratkaisivat useita ongelmia kerralla, mutta peleistä todennäköisesti puuttuisi paljon ominaisuuksia verrattaessa peleihin, jotka on tehty suoraan Androidille.

Yksi suosittu pelimoottori pelien tekoon on Unity, jossa on valmiiksi myös plugin, joka mahdollistaa pelien pelaamisen suoraan selaimesta. Valitettavasti tätä pluginia ei ole vielä tehty Androidille. Lisäksi Unityn ongelmina olisi, kuinka pelit voitaisiin käynnistää sovelluksen sisältä ja miten pelit jaettaisiin käyttäjille.

Ongelmia voitaisiin ratkoa root-tunnuksilla. Rootin avulla voidaan käynnistää pelejä suoraan sovelluksesta ja mahdollisesti ladata pelit palvelimelta taustalla ja asentaa suoraan laitteelle. Tämä mahdollistaisi Unity-pelimoottorin käytön. Eli käyttäjällä olisi peleistä lista, johon voidaan lisätä pelejä nettikäyttöliittymän kautta. Sovellus saisi ilmoituksen, että käyttäjälle on lisätty uusi peli, joka ladattaisiin ja asennettaisiin. Asennuksen onnistuttua käyttäjä voisi yksinkertaisesti käynnistää pelin listasta.

Selainpohjaiset pelit olisivat kaikkein helpoin ratkaisu, koska niitä olisi helppo jakaa käyttäjille ja mitään latausta ei tarvittaisi, vaan käyttäjille annettaisiin vain suora linkki, joka avaisi pelin sovelluksessa olevan Fragmentin sisään.

Yksi mahdollinen ratkaisu olisi myös sisällyttää pelit suoraan sovellukseen, mutta tämä todennäköisesti nostaisi liikaa sovelluksen kokoa, jos pelejä olisi paljon. Hyvänä puolena olisi, että pelit voitaisiin suoraan jakaa Google PlayStoren kautta sovelluksen mukana eli pelien toimivuuden ohjelmoinnissa säästettäisiin paljon aikaa.

4.2 Videopuhelut

Jatkokehityksen yksi tärkeimmistä asioista oli uuden videopuhelurajapinnan toteuttaminen sovellukseen, mutta ajan puutteen ja muiden ongelmien vuoksi sitä ei ehditty toteuttamaan, vaan se jäi suunnitteluasteelle.

Videopuheluille on olemassa ilmainen, vapaaseen lähdekoodiin pohjautuva WebRTC-niminen rajapinta. WebRTC tukee selaimia sekä mobiilisovelluksia. Rajapinta on projekti, jota tukee mm. Google, Mozilla ja Opera. Googlen Chrome-tiimi ylläpitää rajapinnan nettisivuja, joilla on paljon ohjeita ja dokumentaatioita rajapintaa varten. (Google Chrome Team 2015.)

WebRTC:n kehitys vaatii Linuxin, mikä oli myös yksi ongelmista muiden ongelmien ohella miksi rajapintaa ei ole vielä lisätty sovellukseen. Ongelmille koitetaan vielä löytää ratkaisuja, mutta toistaiseksi ei ole varmaa, mikä olisi paras tapa toteuttaa rajapinta sovellukseen.

Videopuhelut toimisivat niin, että käyttäjä että omainen voisivat soittaa suoraan sovellukseen tai sovelluksesta. Käyttäjä pystyisi soittamaan esimerkiksi kontaktilistasta omaisilleen, ja sovellus ottaisi vastaan kaikki puhelut, joita käyttäjä voisi hyväksyä.

Videopuhelut ovat hyvä tapa lisätä ikäihmisten yhteydenpitoa omaisiin ja hoitajiin. Hoitajien ei tarvitsisi aina lähteä paikalle käymään, vaan hoitajat voisivat käyttäjän kanssa helposti selvittää videopuhelun avulla kaikki mahdolliset ongelmat.

Tämän hetkisessä Skype-sovelluksessa käyttäjä ei pysty soittamaan muille, käyttäjä pystyy ainoastaan vastaanottamaan puheluita. Siksi uusi videopuhelurajapinta olisi erittäin hyvä lisä sovellukseen.

5 YHTEENVETO

Sovellus kehittyi opinnäytetyön aikana huomattavasti, koska melkein kaikki puuttuvat ominaisuudet saatiin toteutettua ja virheet korjattua. Sovellusta kehitetään vielä hyvin paljon ja sen lopullinen versio muuttuu tulevaisuudessa, mutta tämänhetkiseen versioon ollaan tyytyväisiä. Sovelluksen jatkokehitys onnistui hyvin eikä ongelmia ilmennyt opinnäytetyön aikana.

Facebookin integrointi sovellukseen onnistui mainiosti, ottaen huomioon kuinka paljon ongelmia siinä oli. Ongelmia oli Facebookin rajapinnassa (paljon erilaisia ohjelmointivirheitä, jotka Facebook onneksi sai korjattua) sekä sovelluksessa, johon oli jäänyt ohjelmointivirheitä viime versiosta.

Uusin Facebookin SDK-versio paransi rajapinnan toimivuutta, koska HTTP-kyselyt voidaan tehdä nyt helposti yhtenä joukkona. Koodi myös lyheni ja järkevöityi huomattavasti. Sovelluksessa olevaan Facebookiin ei kuitenkaan olla täysin tyytyväisiä, koska uudet versiot tuovat tulevaisuudessa paljon muutoksia ja kaikki pienetkin ongelmatilanteet pitää ottaa huomion, että sovellus ei kaadu. Tämä vaatii vielä paljon testailua ja korjaamista, mutta nykyinen versio on hyvä pohja. Joitakin ominaisuuksia voitaisiin vielä lisätä, esimerkiksi nappi jolla voidaan tykätä henkilön julkaisuista.

Myös RecyclerView oli hyvä uudistus, koska vanhat listat olivat vanhentunutta teknologiaa. Uusi RecyclerView helpottaa todella paljon eri ominaisuuksien toteuttamista, jotka olivat todella hankalia vanhalla ListView'llä, esimerkiksi niinkin yksinkertainen asia kuin horisontaalinen lista. Google onnistui uuden RecyclerView'n kehityksessä ja toteuksessa, vaikka ListView oli yksi Androidin tärkeimpiä elementtejä ja sen korvaaminen uudella ei ole helppoa.

Koodin rakennetta päivitetään tulevaisuudessa ja sovelluksen eri toimintoja mietitään uusiksi. Ulkoasu uusitaan todennäköisesti jossakin vaiheessa vielä kokonaan, kunhan uusi ulkoasu saadaan kehiteltyä.

Tulevaisuudessa ongelma on, kuinka voidaan hallita kaikkia käyttäjien tunnuksia, koska monet palvelut vaativat omat kirjautumistunnuksensa. Nykyisessä versiossa on vielä käytössä Skype videopuheluna, mutta tämän toimivuus on ongelmallinen nykyisessä laitteessa ja uusi videopuhelu pitäisi saada integroitua itse sovellukseen. Skype on siis ulkopuolinen sovellus, ja sitä on mahdoton suoraan integroida täysin ohjelman sisälle.

Muutamia ideoita on syntynyt jatkokehityksen aikana, mitkä voitaisiin toteuttaa. Esimerkiksi kun käyttäjä huutaisi ”Apua”, sovellus osaisi automaattisesti lähettää hätäkutsun hoitajalle. Sovellusten toimivuus olisi myös hyvä tarkistaa suoraan netin kautta. Palvelin voisi lähettää kyselyitä kaikille laitteille, joiden avulla voitaisiin tarkistaa, onko niissä ongelmia (esimerkiksi laite sammunut jostakin syystä).

Sovellus lisätään myös Google Playhin, koska sen avulla on helppo ylläpitää eri versioita ja päivittää kaikki laitteet kerralla. Kokonaisuutena jatkokehitys onnistui hyvin, ja toivottavasti mahdollisimman moni pääsee käyttämään sovellusta. On mielenkiintoista nähdä sovelluksen tulevaisuus.

LÄHTEET JA TUOTETUT AINEISTOT

ANDROID DEVELOPER 2015a. Android Studio Overview. [Viitattu 2015-04-20.] Saatavissa: http://developer.android.com/tools/studio/index.html

ANDROID DEVELOPER 2015b. Android Plug-in for Gradle. [Viitattu 2015-04-20.] Saatavissa: http://developer.android.com/tools/building/plugin-for-gradle.html

ANDROID DEVELOPER 2015c. Android Tools. [Viitattu 2015-04-22.] Saatavissa: http://developer.android.com/tools/help/index.html

ANDROID DEVELOPER 2015d. Creating Lists and Cards. [Viitattu 2015-04-15.] Saatavissa: https://developer.android.com/training/material/lists-cards.html

ANDROID DEVELOPER 2015e. Receiver. [Viitattu 2015-05-03.] Saatavissa: http://developer.android.com/guide/topics/manifest/receiver-element.html

FACEBOOK 2015. Access Token. [Viitattu 2015-04-14.] Saatavissa: https://developers.facebook.com/docs/facebook-login/access-tokens

GOOGLE CHROME TEAM 2015. WebRTC. [Viitattu 2015-05-08.] Saatavissa: http://www.webrtc.org/

JSON 2015. JSON. [Viitattu 2015-04-22.] Saatavissa: http://json.org/

ORACLE 2015. Java Language Specification. [Viitattu 2015-04-22.] Saatavissa: http://docs.oracle.com/javase/specs/jls/se8/html/jls-1.html

PHELPS, Thomas 2015. To Root or Not to Root. [Viitattu 2015-04-14.] Saatavissa: http://google.about.com/od/socialtoolsfromgoogle/a/root-android-decision.htm

ŠUKLJE, Matija ja PIANA, Carlo 2012-11-06. Does rooting your device (e.g. an Android phone) and replacing its operating system with something else void your statutory warranty, if you are a consumer? [Viitattu 2015-04-14.] Saatavissa: http://fsfe.org/freesoftware/legal/flashingdevices.en.html

TORTOISE SVN 2015. Tortoise SVN. [Viitattu 2015-04-22.] Saatavissa: http://tortoisesvn.net/about.html