OULUN AMMATTIKORKEAKOULU

Janne Keränen

MULTIPLATFORM-MOBIILISOVELLUKSEN KEHITTÄMINEN QT 5.2:LLA

MULTIPLATFORM-MOBIILISOVELLUKSEN KEHITTÄMINEN QT 5.2:LLA

Janne Keränen
Opinnäytetyö
Kevät 2014
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Janne Keränen
Opinnäytetyön nimi: Multiplatform-mobiilisovelluksen kehittäminen Qt 5.2:lla
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Kevät 2014 Sivumäärä: 23 + 4 liitettä

Opinnäytetyön aiheena oli muuntaa Symbianille tehty mobiilisovellus Qt:lla monialustaiseksi skaalautuvaksi sovellukseksi. Muunnettu ohjelmisto on Daisy-perhepäivähoitosovellus, jolla voidaan hoitaa lasten päivähoitoon liittyviä asioita mobiilisti. Tässä työssä sovellus muunnettiin toimimaan sekä Android- että iOS-käyttöjärjestelmissä.

Ohjelmointi toteutettiin Qt 5.2 -ohjelmointiympäristössä. Käyttöliittymät ohjelmoitiin QML:llä ja toiminnallisuus C++:lla. JNI-ominaisuudet ohjelmoitiin Javalla.

Työn aiheena ollut mobiilisovellus saatiin muunnettua monialustaiseksi skaalautuvaksi sovellukseksi ja sille asetetut tavoitteet saavutettiin.

Asiasanat: Qt 5.2, Android, mobiilisovellukset

ALKULAUSE

Kiitokset SpDesign Oy:lle opinnäytetyön aiheesta, sisällönohjaamisesta työn ohjaajalle Kari Laitiselle sekä työkavereille mukavasta työilmapiiristä.

26.5.2014

Janne Keränen

SISÄLLYS

SANASTO

Android	Googlen kehittämä käyttöjärjestelmä mobiililaitteille
Android NDK	Android Native Development Kitillä voidaan suorittaa Androidissa natiivikieliä kuten C/C++
Android SDK	Android Software Development kit on Android-sovelluskehityspaketti
Ant	Java-pohjainen työkalu automaattiseen kääntämiseen
C++	Ohjelmointikieli
Daisy	Varhaiskasvatuksen läsnäolonseurantajärjestelmä.
Java	Ohjelmointikieli
JDK	Java Development Kit on Java ohjelmistokehityspaketti
JNI	Java Native Interface
NFC	Near Field Communication mahdollistaa tiedonsiirron lyhyellä etäisyydellä
QML	Qt Modeling Language on Qt-käyttöliittymien ohjelmointikieli, joka perustuu JavaScriptiin
Qt	Alustariippumaton ohjelmistojen ja graafisten käyttöliittymien kehistysympäristö
QtCreator	Qt:n mukana tuleva IDE (integrated development environment) -ohjelmointiympäristö

1 JOHDANTO

Daisy on varhaiskasvatuksen kokonaisvaltainen läsnäolon seurantajärjestelmä. Järjestelmään kuuluvat Daisy Net, Daisy Manager, Daisy-päiväkoti-mobiilisovellus sekä Daisy-perhepäivähoito-mobiilisovellus. Tässä opinnäytetyössä oli työn kohteena perhepäivähoidon mobiilisovellus, joka toimii osana järjestelmää. Sillä tehdään lasten ja työntekijöiden kirjaukset ja siitä on nähtävissä tehdyt työaikakirjaukset, kertyneet tunnit, mahdolliset hyvitykset ja kulukorvaukset.

Tässä opinnäytetyössä muunnettiin Daisy-perhepäivähoitosovellus toimimaan monialustaisesti mobiililaitteissa. Daisy-perhepäivähoitosovellus oli tehty Qt:lla Symbianille ja Qt:lla pystyy myös ohjelmoimaan monialustaisia mobiili-sovelluksia, joten sama projekti toimi pohjana. Osa ominaisuuksista oli ohjelmoitu Symbianilla, ja ne poistettiin koodista. Qt:n versiossa 5.2 kaikkia vaadittavia ominaisuuksia ei ole tuettu, joten ne piti ohjelmoida uudestaan JNI:llä (Java Native Interface). Käyttöliittymä piti muuttaa skaalautuvaksi, koska Symbian-sovelluksessa se oli tehty toimimaan kiinteästi 640 x 360 pikselin resoluutiolla. Tavoitteena oli saada kaikki ominaisuudet toimimaan sekä käyttöliittymä skaalautumaan eri resoluutioille.

Qt on alustariippumaton sovellusten ja graafisten käyttöliittymien kehitysympäristö. Sillä voidaan tehdä natiiveja työpöytä-, sulautettuja ja mobiilisovelluksia. Työssä käytetty versio Qt:stä oli 5.2.

Tässä raportissa kaikkea ei voitu käydä läpi yksityiskohtaisesti, koska kyseessä oli asiakasprojekti ja näin ollen lähdekoodi on salattu.

2 QT 5.2 MULTIPLATFORM -SOVELLUSKEHITYS

2.1 Qt-ohjelmointiympäristö

Qt on alustariippumaton kehitysympäristö, jota käytetään laajasti mobiili- ja työpöytäsovellusten ohjelmoimiseen, joissa on graafinen käyttöliittymä. Qt:lla on myös mahdollista ohjelmoida komentorivityökaluja ja konsoleita palvelimille. Qt tukee kaikkia keskeisiä työpöytäkäyttöjärjestelmiä, kuten Windows, Mac OS X ja Linux. Käytettävä ohjelmointikieli on C++ ja QML (Qt Meta Language). Qt:stä on olemassa kaupallinen ja avoimen lähdekoodin versio. (Qt_(software). 2014.)

2.2 Qt 5.2:n tukemat mobiilialustat

2.2.1 Android-käyttöjärjestelmä

Android on Googlen kehittämä käyttöjärjestelmä, joka perustuu Linuxin ytimeen. Se on ensisijaisesti tarkoitettu kosketusnäytöllisiin mobiililaitteisiin. Androidia ohjelmoidaan ensisijaisesti Javalla. Virallisesti tuettu ohjelmointiympäristö (Integrated Development Environment) on Eclipse, johon on asennettu Android-ohjelmointityökalut (Android Development Tools).
(Android_(operationg_system). 2014)

Qt:ssä Android-sovelluksia voidaan kehittää Windowsilla, Linuxilla ja Mac OS X:llä. Jokaiselle alustalle löytyy Qt:n valmiit asennuspaketit. Toisin kuin Eclipsellä ohjelmoitaessa, Qt:ssa käytettävä ohjelmointikieli on C++. Qt:lla ohjelmoidut sovellukset on mahdollista laittaa Androidin Google Play -sovelluskauppaan. Android-ohjelmointiin Qt:lla tarvitaan Qt:n lisäksi vielä Android Software Development Kit, Android Native Development Kit, Ant sekä Java Development Kit, jotka määritetään Qt:n asetuksiin.

2.2.2 iOS-käyttöjärjstelmä

iOS on Applen kehittämä käyttöjärjestelmä Applen mobiililaitteille. iOS on alun perin julkaistu iPhonella vuonna 2007. Tämän jälkeen se on laajentanut myös muihin Applen laitteisiin, kuten iPod Touch, iPad, iPad Mini ja Apple TV. Lähdekoodi on suljettu, eikä käyttöjärjestelmää ole muiden valmistajien

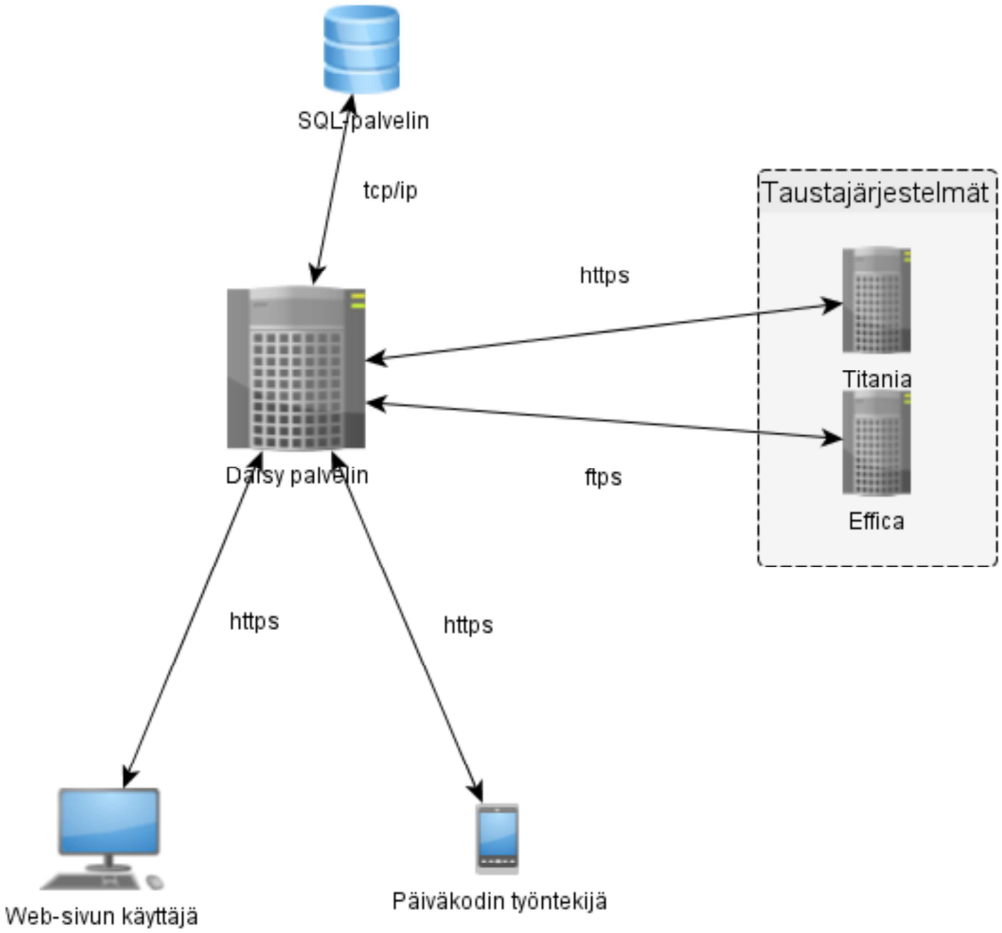
laitteissa. iOS-sovelluskehityksessä käytetään tavanomaisesti Xcodea. Xcode on Applen kehittämä ohjelmointiympäristö. (iOS. 2014.)

Qt:n iOS-versiolla voidaan kääntää Qt-sovelluksia iPhonelle ja iPadille. Qt:lla ohjelmoidut sovellukset on mahdollista laittaa Applen App Store -sovelluskauppaan. Qt:n lisäksi iOS-ohjelmointiin tarvitaan vielä myös Xcode.

3 DAISY-JÄRJESTELMÄ

Daisy on WhileOnTheMove Oy:n kehittämä varhaiskasvatuksen tietojärjestelmä. Järjestelmään kuuluvat mobiilisovellukset päiväkodeille ja perhepäivähoitajille sekä web-sovellukset Daisy Net ja Daisy Manager. Daisy Net on lasten vanhemmille tarkoitettu web-sovellus, josta voidaan seurata lapsen hoidossa oloaikoja, tehdä hoitovarauksia, ilmoittaa lasten poissaoloista ja lähettää viestejä päivähoidon ja vanhempien välillä. Daisy Manager on perhepäivähoidon hallinnolle ja johdolle tarkoitettu nettisovellus, josta löytyvät hoito- ja palvelusopimukset, erilaisia raportteja, hoitoaikavaraukset ja viestintä lasten vanhempien kanssa. (Daisy – Enemmän aikaa lapselle. 2014.)

Daisy-mobiilisovelluksissa tehdään työ- ja hoitoaikakirjaukset. Kirjauksien tekemisessä hyödynnetään NFC-tekniikkaa (Near Field Communication). Mobiilisovelluksista nähdään ajantasainen tieto paikalla olevista työntekijöistä ja lapsista sekä kirjausten kelloajat. Lisäksi nähdään myös lasten allergiat ja vanhempien yhteystiedot. Opinnäytetyön aiheena olleesta perhepäivähoidon mobiilisovelluksessa nähdään edellä mainittujen lisäksi lapsien kertyneet hoitoajat, ateriakulut sekä lomat ja poissaolot. Perhepäivähoitaja näkee omista tiedoistaan suunnitellut- ja toteutuneet työajat. Mobiilisovellus on yhteydessä tietokantaan mobiililaitteen Internet-yhteyden avulla. Daisy-järjestelmän arkkitehtuurikuvaus on kuvassa 1. (Daisy – Enemmän aikaa lapselle. 2014.)



KUVA 1. Daisy-arkkitehtuurikuvaus (Keränen 2013, 9)

4 DAISY-MOBIILISOVELLUKSEN MUUNTAMINEN

4.1 Projektin muokkaaminen

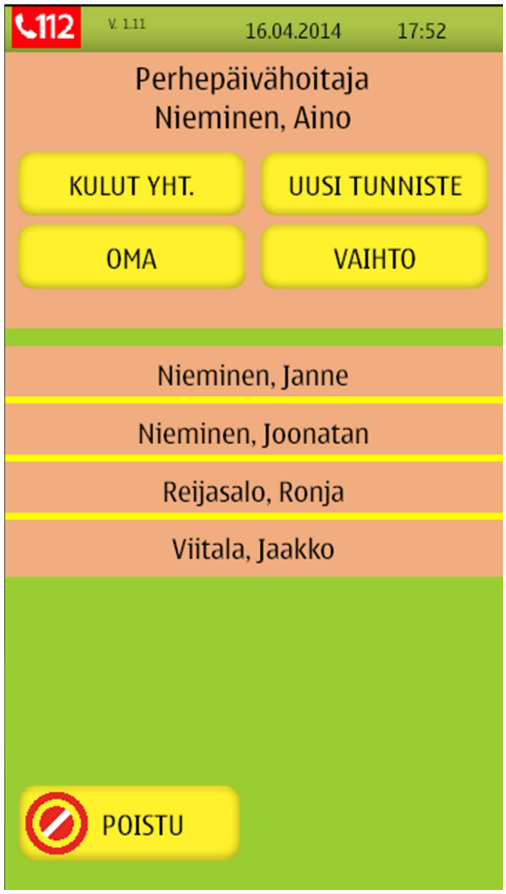
Aluksi tehtiin resurssitiedosto ja lisättiin sinne sovelluksessa käytetyt äänet, kuvat ja käyttöliittymän QML-tiedostot. Sitten muokattiin otsikkotiedostojen sisällytykset kuntoon ja poistettiin Symbian-koodit sekä ominaisuudet, joita ei ole tuettu 5.2-versiossa muun muassa NFC ja värinä.

Mobiilisovellus käytti QtQuick 1:tä ja se piti muuttaa käyttämään uudempaa QtQuick 2:ta. Tämä tehtiin Internetistä löytyneiden muuntamisohjeiden mukaan (Porting QML Applications to Qt 5. 2013).

Tämän jälkeen sovellus oli mahdollista kääntää Android-laitteelle, mutta käyttöliittymä ei skaalautunut koko näytölle, koska ohjelma oli tehty kiinteästi 640 x 360 pikselin resoluutiolle ja käytössä olevan Android-laitteen resoluutio oli 800 x 480 pikseliä.

4.2 QML-käyttöliittymän muokkaaminen

Opinnäytetyöni työläin vaihe oli käyttöliittymän muokkaaminen skaalautuvaksi. Daisy-perhepäivähoitosovelluksessa käyttöliittymä oli toteutettu QML:llä. Jokaisesta näkymästä oli tehty oma sivunsa, jotka oli ohjelmoitu toimimaan kiinteästi 640x360 resoluutiolla. Käyttöliittymä muokattiin sivu kerrallaan skaalautuvaksi. Käyttöliittymän muokkaaminen aloitettiin päänäkymästä (kuva 2).



KUVA 2. Daisy-perhepäivähoito-sovelluksen pääsivu Symbin-puhelimessa.

Päänäkymä koostuu periaatteessa neljästä osiosta. Yläreunassa on yläpalkki, joka sisältää hätäpuhelunapin, versionumeron, päivämäärän ja kellonajan. Sen alapuolella on vaaleanpunainen suorakaide, jossa on valitun perhepäivähoitajan nimi ja neljä nappia. Seuraavana alapuolella on Listview-komponentti, jossa on lueteltuna kyseisen perhepäivähoitajan lapset. Alareunassa on alapalkki, jossa on kaksi painiketta. Pääsivulla toinen nappula on tarpeeton, joten se on piilotettu.

Yläpalkista tehtiin Header-niminen komponentti (kuva 3). Komponentti on tietyn kokoinen suorakulmio, jonka leveys määriteltiin koko ruudun levyiseksi ja korkeus 5 % ruudusta pystysuunnassa ja ankkuroitiin ylälaitaan. Tämän jälkeen suorakulmion sisään laitettiin kuva, joka määritettiin täyttämään koko suorakulmio. Seuraavaksi luotiin hätäpuhelunappi, joka ankkuroitiin suorakulmion vasempaan laitaan. Versionumero ankkuroitiin hätäpuhelunapin

oikeaan reunaan jos se on näkyvissä, muulloin kuvan vasempaan reunaan. Päivämäärä ankkuroitiin puolestaan versionumeron oikeaan laitaan. Tämän jälkeen oli vuorossa kellonaika, joka ankkuroitiin kuvan oikeaan laitaan. Kellonajan ja päivämäärän päivittämistä varten tehtiin ajastin, joka päivittää kellonajan ja päivämäärän 10 sekunnin välein. Tekstin fonttikoot ovat myös skaalautuvia ja fontin kooksi pikseleinä määritettiin 2 % ruudun koosta pystysuunnassa. (Kuva 3.)

```
import QtQuick 2.1
import "comfunctions.js" as ComFunctions

Rectangle {
    width: page.width
    height: page.height*.05
    property string versionNumberText
    property bool versionNumberVisible: false
    Item{
        Timer {
            interval: 10000; running: true; repeat: true
            onTriggered: {
                dateText.text = Qt.formatDateTime(ComFunctions.getDate2(), "dd.MM.yyyy");
                timeText.text = Qt.formatDateTime(ComFunctions.getDate2(), "HH:mm");
            }
        }
    }

    Image {
        id: image1
        anchors.fill: parent
        source: "header.png"
        AlertButton{
            id: alertButton
        }

        Text{
            id: versionNumber
            text: versionNumberText
            visible: versionNumberVisible
            font.pixelSize: page.height*.02
            anchors.verticalCenter: parent.verticalCenter
            anchors.left: alertButton.visible ? alertButton.right : parent.left
            anchors.leftMargin: page.height*.015
        }
        Text {
            id: dateText
            text: Qt.formatDateTime(ComFunctions.getDate2(), "dd.MM.yyyy")
            font.pixelSize: page.height*.02
            anchors.left: versionNumber.right
            anchors.leftMargin: page.height*.015
            anchors.verticalCenter: parent.verticalCenter
        }
        Text {
            id: timeText
            text: Qt.formatDateTime(ComFunctions.getDate2(), "HH:mm")
            font.pixelSize: page.height*.02
            anchors.right: parent.right
            anchors.rightMargin: page.width*.11
            anchors.verticalCenter: parent.verticalCenter
        }
    }
}
```

KUVA 3. Header-komponentin QML-lähdekoodi.

Kun komponentti oli saatu tehtyä, se piti vielä näyttää pääsivulla. Näyttäminen tapahtui lisäämällä kuvan 4 mukainen koodi pääsivulle. (Kuva 4.)

```
Header {
    id: header1
    anchors.right: parent.right
    anchors.left: parent.left
    anchors.top: parent.top
    versionNumberVisible: true
    versionNumberText: getDaisyVersion()

}
```

KUVA 4. Header-komponentin käyttäminen

Seuraavana oli vuorossa suorakulmio, jossa on neljä painiketta ja perhepäivähoitajan nimi. Suorakulmiossa on neljä samanlaista painiketta, joten painikkeesta tehtiin BasicButton-komponentti, jota voidaan käyttää tekemättä jokaista painiketta kokonaan erikseen eli turha toistuvuus koodissa vähenee. Painike koostuu kuvasta ja kuvan keskellä olevasta tekstistä. Painiketta painettaessa välittyy ”clicked”-signaali. (Kuva 5.)

```
import QtQuick 2.1

Image {
    id: button
    width: page.width*.47
    height: page.height*.07
    source: imageSource
    property string buttonText
    property string imageSource: "nappula.png"
    signal clicked()
    MouseArea {
        id: buttonMouseArea
        anchors.fill: parent
        onClicked: {
            button.clicked()
        }
    }
    Text {
        text: buttonText
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
        wrapMode: Text.WordWrap
        font.pixelSize: parent.height*.4
    }
}
```

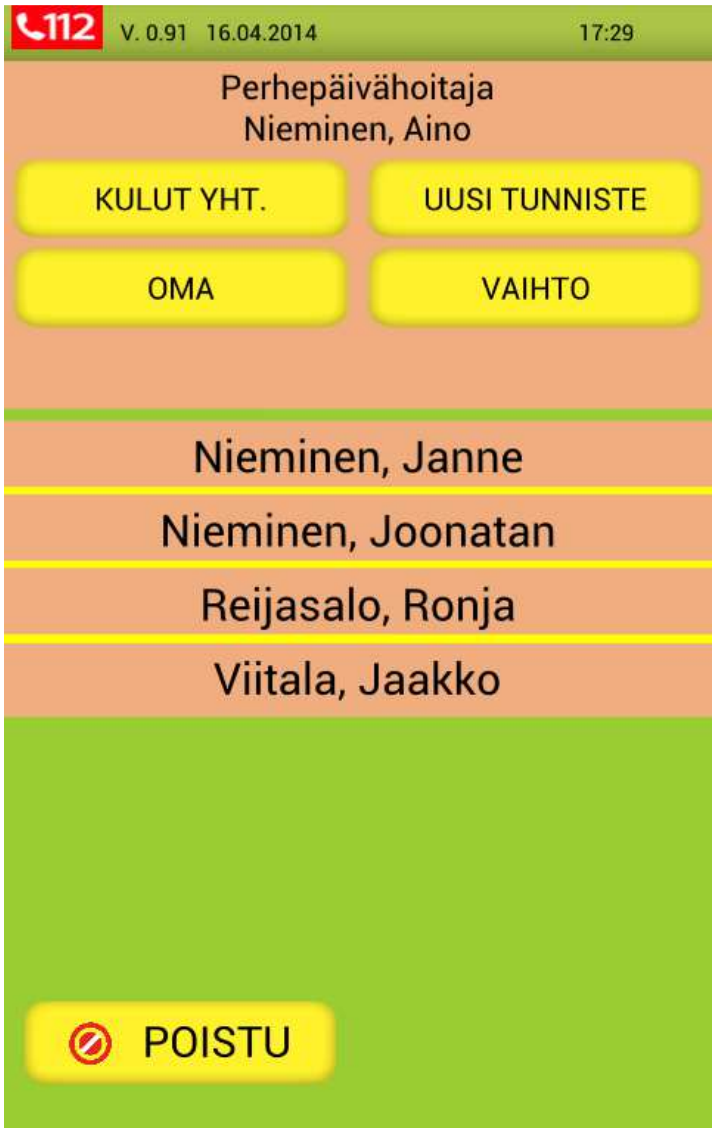
KUVA 5. BasicButton-lähdekoodi.

Tämän jälkeen tehtiin suorakulmio jonka koko määritettiin koko sivun levyiseksi ja 31 % pystysuunnassa ja ankkuroitiin Header-komponentin alareunaan. Seuraavaksi lisättiin perhepäivähoitajan nimi. Nimi keskitettiin sivuttaissuunnassa ja ankkuroitiin suorakulmion ylälaitaan. Tämän jälkeen lisättiin vielä neljä BasicButton-komponenttia ja määritettiin niille ankkurointi, tekstit sekä toiminnallisuus. (Liite 1.)

Seuraavaksi tehtiin BottomBar-komponentti. Tässä komponentissa on kaksi nappia, mutta toinen on tällä sivulla piilotettu. Komponentin kooksi määritettiin koko ruutu sivuttaissuunnassa ja 15 % pystysuunnassa. Tässä komponentissa on kaksi samanlaista nappia, joten niistä tehtiin BottomButton-komponentti (liite 2). Napit sijoitettiin BottomBar-komponentin molempiin laitoihin ja määritettiin niille toiminnallisuus. (Liite 3.)

Pääsivun viimeisenä oli lapsilistan muokkaaminen. Lapset ovat listattuna ListView-komponettiin, joka on valmiina Qt:ssa. ListView'n koko määritettiin sivuttaissuunnassa koko ruudun levyiseksi ja pystysuunnassa ankkuroitiin StatusBarin alareunaan ja BottomBarin yläreunaan. Kuvakaappaus muunnetusta pääsivusta kuvassa 6.

Pääsivun jälkeen muokattiin loputkin QML-sivut samaan tyyliin.



KUVA 6. Daisy-perhepäivähoito-sovelluksen päänäkymä Samsung Galaxy S2 Plus Android-puhelimessa.

4.3 JNI-ominaisuudet

Qt 5.2 -versio ei tukenut kaikkia tarvittavia ominaisuuksia ja tämän vuoksi NFC ja värinä toteutettiin JNI:llä (Java Native Interface). JNI:n avulla on mahdollista kutsua Javalla kirjoitettuja metodeita. JNI:llä toteutettavat ominaisuudet ohjelmoitiin erilliseen Java-luokkaan, joita sitten kutsuttiin C++-luokasta. JNI:llä toteutattavat ominaisuudet lisättiin if-lauseiden sisälle, jotta sama projekti kääntyisi suoraan iOS:lle (kuva 7).

```
#ifdef Q_OS_ANDROID

    //JNI-ominaisuudet

#endif
```

KUVA 7. Android-laitteissa suoritettavat JNI-ominaisuudet

Ensimmäiseksi tehtiin DaisyActivity Java-luokka, joka perii sovelluksen pääluokan. Sinne ohjelmoitiin metodi, joka palauttaa kontekstin (kuva 8). AndroidManifest.xml-tiedostoon myös määritettiin activityksi DaisyActivity.

```
package org.qtproject.qt5.android.bindings;

import org.qtproject.qt5.android.bindings.QtActivity;
import android.app.Activity;

public class DaisyActivity extends QtActivity {

    private static DaisyActivity m_instance;

    public DaisyActivity()
    {
        m_instance = this;
    }
    public static DaisyActivity getQtActivityInstance()
    {
        return m_instance;
    }
}
```

KUVA 8. DaisyActivity-luokan lähdekoodi.

Tämän jälkeen ruvettiin ohjelmoimaan värinää. Värinätoiminnon käyttäminen Androidissa vaatii käyttöoikeudet, jotka määritettiin Androidmanifest.xml-tiedostoon. Värinäominaisuudelle tehtiin oma luokka, jonka muodostimessa haettiin konteksti DaisyAcvitity-luokasta ja sitten luotiin vibrator (kuva 9). Värinän käyttämistä varten luotiin vibrate-funktio, jota kutsumalla värinää voidaan käyttää. Esimerkkinä käytettiin JNI:llä toteutettua QSimpleAudioPlayer-esimerkkiä (JNI. 2013).

```
package org.kde.necessitas.origo;

import android.content.Context;
import android.os.Vibrator;
import org.qtproject.qt5.android.bindings.DaisyActivity;

public class QVibrator {

        Vibrator vibrator;
        Context mContext;

        public QVibrator()
        {
                mContext = DaisyActivity.getQtActivityInstance();
                vibrator = (Vibrator) mContext.getSystemService(mContext.VIBRATOR_SERVICE);
        }

        public void vibrate()
        {
                vibrator.vibrate(500);
        }

}
```

KUVA 9. Vibrator-luokan lähdekoodi.

Vibrator-luokan tekemisen jälkeen C++-luokassa määritettiin ID:t luokalle, muodostimelle ja metodille (kuva 10).

```
jclass vibratorclazz = env->FindClass("org/kde/necessitas/origo/QVibrator");
if (!vibratorclazz)
{
    qDebug()<<"Can't find QVibrator class";
    return -1;
}

s_vibratorClassID = (jclass)env->NewGlobalRef(vibratorclazz);

s_vibratorConstructorMethodID = env->GetMethodID(s_vibratorClassID, "<init>", "()V");
if (!s_vibratorConstructorMethodID)
{
    qDebug()<<"Can't find QVibrator class contructor";
    return -1;
}

s_vibratorVibrateMethodID = env->GetMethodID(s_vibratorClassID, "vibrate", "()V");
if (!s_vibratorVibrateMethodID)
{
    qDebug()<<"Can't find vibrate method";
    return -1;
}
```

Kuva 10. Java-luokan metodien määritys.

Seuraavaksi luotiin m_jvibratorObject (kuva 11).

```
JNIEnv* env;

if (s_javaVM->AttachCurrentThread(&env, NULL)<0)
{
    qDebug()<<"AttachCurrentThread failed";
    return;
}
m_jvibratorObject = env->NewGlobalRef(env->NewObject(s_vibratorClassID,
                                                     s_vibratorConstructorMethodID));

if (!m_jvibratorObject)
{
    qDebug() << "can't create vibrator object";

} else
{
    qDebug() << "vibrator object created";
}
s_javaVM->DetachCurrentThread();
```

KUVA 11. Vibrator-objectin luominen.

Tämän jälkeen värinäominaisuutta voitiin käyttää kuvan 12 tavalla.

```
JNIEnv* env;

if (s_javaVM->AttachCurrentThread(&env, NULL)<0)
{
    qDebug() << "Attach java env failed!";
    return;
}

env->CallVoidMethod(m_jvibratorObject,s_vibratorVibrateMethodID);

s_javaVM->DetachCurrentThread();
```

KUVA 12. Vibrate-metodin kutsuminen.

4.4 Testaaminen

Testaaminen suoritettiin useilla eri mobiililaitteilla. Testilaitteet olivat puhelimia ja tabletteja, joissa oli useita erikokoisia näyttöjä, sekä eri resoluutioita. Testaamisessa käytettyjä tabletteja oli Google Nexus 7 sekä Apple iPad ja puhelimia Samsung Galaxy S2 plus ja Samgung Galaxy S3. Aluksi testattiin käyttöliittymän skaalautuminen eri laitteisiin. Käyttöliittymän testaamisessa verrattiin uutta versiota vanhaan, joka oli asennettu Nokia 700 -puhelimeen. Sovellus käytiin läpi näkymä kerrallaan, ja todettiin, että käyttöliittymän skaalautuu hyvin jokaisella laitteella.

Käyttöliittymän kuntoon saamisen jälkeen, siirryttiin testaamaan toiminnallisuutta. Testissä testattiin kaikki erilaiset kirjausvariaatiot, viestien lähetys ja kulujen lisääminen. Testissä ilmeni, että uusi versio toimii kuin vanha versio.

5 POHDINTA

Opinnäytetyön aiheena oli Symbian-laitteissa toimivan Daisy-perhepäivähoito-mobiilisovelluksen muuntaminen monialustaiseksi skaalautuvaksi sovellukseksi. Työn tarkoituksena oli saada kyseinen mobiilisovellus tukemaan enemmän käyttöjärjestelmiä. Työ oli mielenkiintoinen projekti, jossa opin uusia asioita Qt-ohjelmoinnista, JNI:stä sekä ohjelmistoprojektista. Työ saatiin tehtyä ja sille asetetut tavoitteet saavutettiin.

Työssä käytetystä Qt:stä oli aikaisempaa kokemusta työpöytä- ja Symbian-sovelluksien ohjelmoinnista. Aikaisempi kokemus vaikutti sovelluksen muuntamiseen positiivisesti, sillä kaikkea ei joutunut opettelemaan uutena asiana. JNI-ominaisuuksia tehdessä oli välillä vaikeuksia, koska aikaisempaa kokemusta ei ollut, mutta internetistä löytyvien esimerkkien sekä työkavereiden avulla JNI-ominaisuudet saatiin tehtyä. Käyttöliittymän muokkaamisessa onnistuttiin ja se skaalautui erikokoisillle näytöille hyvin.

Qt:lla samoista lähdekoodeista voidaan sovellukset kääntää usealle eri alustalle, joten se on nopeampi, kuin jos tehtäisiin erikseen sovellukset joka alustalle.

LÄHTEET

Android_(operationg_system). 2014. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Android_(operating_system). Hakupäivä 1.4.2014.

Daisy – Enemmän aikaa lapselle. 2014. WhileOnTheMove. Saatavissa: http://whileonthemove.com/Tuote_Daisy.html. Hakupäivä 18.5.2014.

iOS. 2014. Wikipedia. Saatavissa: http://fi.wikipedia.org/wiki/IOS. Hakupäivä 5.4.2014

JNI. 2013. KDE Community Wiki. Saatavissa: http://community.kde.org/Necessitas/JNI. Hakupäivä 2.5.2014.

Keränen, Mikko 2013. Web-palvelun skaalaus isoille käyttäjämäärille. Opinnäytetyö. Oulu: Oulun seudun ammattikorkeakoulu, tietotekniikan koulutusohjelma. Saatavissa: https://publications.theseus.fi/handle/10024/57882. Hakupäivä 18.5.2014.

Porting QML Applications to Qt 5. Qt-project, 2013. Saatavissa: http://qt-project.org/doc/qt-5.0/qtquick/qtquick-porting-qt5.html. hakupäivä: 11.3.2014.

Qt_(software). 2014. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Qt_(software). Hakupäivä: 1.4.2014.

```
Rectangle {
    id: statusBar
    width: page.width
    height: page.height * .31
    color: "#4cc4f0"
    anchors.top: header1.bottom

    Text {
        id: textCentreName
        text: ComFunctions.familyNurseText
        anchors.top: parent.top
        anchors.topMargin: page.height*.005
        anchors.horizontalCenter: parent.horizontalCenter
        horizontalAlignment: Text.AlignHCenter
        font.pixelSize: page.height*.03
    }

    BasicButton {
        id: expensesButton
        anchors {
            left: parent.left;
            top: textCentreName.bottom;
            margins: parent.height*.03
        }
        buttonText: ComFunctions.expensesTotalText
        onClicked: myQmlObject.setQml(1)
    }

    BasicButton {
        id: newTagButton
        anchors {
            right: parent.right
            top: textCentreName.bottom
            margins: parent.height*.03
        }
        buttonText: ComFunctions.newTagText
        onClicked: myQmlObject.setQml(4)
    }

    BasicButton {
        id: ownInfoButton
        anchors { left: parent.left
            top: expensesButton.bottom
            margins: parent.height*.03 }
        buttonText: ComFunctions.ownInfoText
        onClicked: myQmlObject.setQml(3)

        Image {
            id: message
            width: page.width*.086
            height: page.width*.086*.58
            anchors.right: parent.right
            anchors.rightMargin: 5
            anchors.verticalCenter: parent.verticalCenter
            source: "viesti.jpg"
            visible: messagevisible
        }
    }

    BasicButton {
        id: changeButton
        anchors {
            right: parent.righ
            top: newTagButton.bottom
            margins: parent.height*.03
        }
        buttonText: ComFunctions.changeText
        onClicked: myQmlObject.setQml(5)
    }
}
```

```
import QtQuick 2.1

Image {
    id: root
    property string buttonText: ""
    property string buttonImage: ""
    property bool buttonTextCenter
    property double buttonTextSize: root.height*.5
    property double buttonIconSize: root.height*.5
    signal clicked
    source: "nappula.png"

    Row {
        spacing: root.width*.1
        anchors{top:parent.top; bottom:parent.bottom; horizontalCenter: parent.horizontalCenter}

        Image {
            id: buttonIcon
            width:height
            visible: !buttonTextCenter
            height: buttonIconSize
            source: root.buttonImage
            anchors.verticalCenter: parent.verticalCenter
        }

        Text {
            id: buttonText
            text: root.buttonText
            visible: !buttonTextCenter
            font.pixelSize: buttonTextSize
            anchors.verticalCenter: parent.verticalCenter
        }
    }
    Text {
        id: buttonText2
        text: root.buttonText
        visible: buttonTextCenter
        font.pixelSize: buttonTextSize
        anchors.verticalCenter: parent.verticalCenter
        anchors.horizontalCenter: parent.horizontalCenter
    }

    MouseArea {
        anchors.fill: parent
        onClicked: {
            root.clicked();
        }
    }
}
```

```
import QtQuick 2.1
import "comfunctions.js" as ComFunctions

Rectangle {
    id: root
    anchors { left: page.left; right: page.right; bottom: page.bottom }
    height: page.height*.15
    color: "#9ACD32"
    property string okButtonText: ""
    property string okButtonIconSource: ""
    property string cancelButtonText: ""
    property string cancelButtonIconSource: ""
    property bool okButtonTextCenter: false
    property bool cancelButtonTextCenter: false
    property bool hideOkButton: false
    property bool hideCancelButton: false
    property bool invertButtons: false

    signal cancel()
    signal ok()

    BottomButton {
        anchors {
            left: parent.left
            leftMargin: page.width*.028;
            verticalCenter: parent.verticalCenter
        }
        width: parent.width*.44
        height: parent.height*.5
        buttonText: invertButtons ? okButtonText : cancelButtonText
        buttonImage: invertButtons ? okButtonIconSource : cancelButtonIconSource
        onClicked: invertButtons ? root.ok() : root.cancel()
        buttonTextCenter: invertButtons ? okButtonTextCenter : cancelButtonTextCenter
        visible: invertButtons ? !hideOkButton : !hideCancelButton
    }
    BottomButton {
        anchors {
            right: parent.right;
            rightMargin: page.width*.03
            verticalCenter: parent.verticalCenter
        }
        width: parent.width*.44
        height: parent.height*.5
        buttonText: invertButtons ? cancelButtonText : okButtonText
        buttonTextCenter: invertButtons ? cancelButtonTextCenter : okButtonTextCenter
        buttonImage: invertButtons ? cancelButtonIconSource : okButtonIconSource
        onClicked: invertButtons ? root.cancel() : root.ok()
        visible: invertButtons ? !hideCancelButton : !hideOkButton
    }
```

```
package org.kde.necessitas.origo;

import android.content.Context;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import org.qtproject.qt5.android.bindings.QtActivity;

public class QVibrator {

        Vibrator vibrator;
        Context mContext;

        public QVibrator()
        {
                mContext = QtActivity.getQtActivityInstance();
                vibrator = (Vibrator) mContext.getSystemService(mContext.VIBRATOR_SERVICE);
        }

        public void vibrate()
        {
                vibrator.vibrate(500);
        }

}
```