KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka / Ohjelmistotekniikka

Niina Puhakka

TIETOKANTASOVELLUKSEN KEHITTÄMINEN ANDROIDILLE

Opinnäytetyö 2015

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka

PUHAKKA, NIINA

TIETOKANTASOVELLUKSEN KEHITTÄMINEN ANDROIDILLE

Opinnäytetyö 47 sivua
Työn ohjaaja Yliopettaja Paula Posio
Toimeksiantaja Kymenlaakson ammattikorkeakoulu
Maaliskuu 2015
Avainsanat Android, Eclipse, SDK, sovellus, tietokanta, SQLite

Android on tällä hetkellä suosituin käyttöjärjestelmä älypuhelimissa, ja sen suosio kasvaa jatkuvasti. Androidille kehittäminen on ilmaista, ja sille tehdään todella paljon erilaisia sovelluksia.

Työn tarkoituksena oli ohjelmoida yksinkertainen sovellusprototyyppi älypuhelimeen. Työssä tutustuttiin Android-ohjelmointiin ja erityisesti tiedon tallentamiseen Android-järjestelmällä. Työkaluina käytettiin Eclipseä ja siihen asennettua Android SDK:ta. Prototyyppi kirjoitettiin java-ohjelmointikielellä. Tiedon tallennus toteutettiin SQLite-tietokantajärjestelmällä. Raportissa käydään läpi prototyypin suunnittelu alusta alkaen sekä ohjelmoinnin vaiheet.

Työn tuloksena ei syntynyt valmista prototyyppiä, kuten alun perin oli tarkoitus, mutta tiedon tallennus saatiin kuitenkin toteutettua. Prototyypin luominen oli hyödyllinen kokemus, sillä se kehitti suunnittelu- ja ohjelmointitaitoja sekä projektinhallintaa.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology
PUHAKKA, NIINA

DEVELOPING DATABASE APPLICATION FOR ANDROID

Bachelor's Thesis 47 pages
Supervisor Paula Posio, Principal Lecturer
Commissioned by Kymenlaakso University of Applied Sciences
March 2015
Keywords Android, Eclipse, SDK, application, database, SQLite

At the moment, Android is the most popular operating system on smart phones, and its popularity increases all the time. It does not cost anything to develop for Android, and many applications are made for it.

The purpose was to program a simple application prototype for smart phones. The project focused on Android development and especially how to save data on Android system. Tools that were used for this project were Eclipse and Android SDK that was installed inside the Eclipse. The prototype was written in java programming language. Data was saved with SQLite database system. Prototype planning and programming phases are explored in this thesis.

The result of this project was not a complete prototype as originally planned but data saving was accomplished.

TERMIT JA LYHENTEET

ADT	Paketti, joka sisältää Eclipsen ja Android-kehityksen työkalut (Android Development Tools).
aktiviteetti	Android-sovelluksen komponentti, joista käyttöliittymä rakentuu.
Android Manifest	Tiedosto, joka kuvailee Android-sovelluksen komponentit.
Android SDK	Paketti, joka sisältää Android-kehittäjän tarvitsemat työkalut (Android Software Development Kit).
Android Studio	Työkalu Android-ohjelmointiin
Broadcast Receiver	Android-sovelluksen komponentti, joka kuuntelee intenttejä.
Content Provider	Android-sovelluksen komponentti, joka hallitsee sovellusdataa.
ContentValues	Luokka, jota käytetään arvojen varastointiin.
Cursor	Rajapinta, joka tarjoaa luku- ja kirjoitusoikeuden tietokantakyselyn palauttamaan kokoelmaan.
Dalvik	Virtuaalikone, jonka päällä Android-sovellukset toimivat.
DDMS	Debuggaustyökalu Androidille (Dalvik Debug Monitor Server).
Eclipse IDE	Ohjelmointiympäristö
EditText	Elementti, joka ottaa vastaan käyttäjän syötteitä.
emulaattori	Virtuaalikone, joka jäljittelee jonkin laitteen toimintaa
execSQL	SQLite-funktio, jolla voi suorittaa SQL-komennon.
getText	Androidin funktio, jolla tekstikenttien tekstit palautetaan xml-tiedostosta.
getWritableDatabase	SQLiteOpenHelper-luokan funktio tietokannan luomiseen ja avaamiseen.
Google Play	Sovelluskauppa
insertOrThrow	SQLite-funktio, joka lisää uuden rivin tietokantaan.
Intent	Android-sovelluksen komponentti, jolla lähetetään viestejä eri puolille järjestelmää.
JDK	Java-virtuaalikone (Java Development Kit)
Lollipop	Uusimman Android-version (5.0) nimi.
natiivisovellus	Käyttöjärjestelmän mukana tullut sovellus

Notifikaatio	Android-sovelluksen komponentti, joka tiedottaa käyttäjälle sovelluksessa tapahtuvista asioista.
onCreate	Androidin funktio, jossa aktiviteetti alustetaan.
OnCreateOptionsMenu	Androidin funktio, joka alustaa aktiviteetin käyttöliittymän valikon.
OnOptionsItemSelected	Androidin funktio, jota kutsutaan, kun aktiviteetin valikosta valitaan jokin kohde.
onUpgrade	SQLite-funktio tietokannan päivittämiseen.
query	SQLite-funktio, joka tekee kyselyn taulukkoon.
RelativeLayout	Asettelumalli, jossa elementit sijoittuvat toistensa sijainnin perusteella paikoilleen.
setContentView	Androidin funktio, joka määrittelee sovelluksen käyttöliittymän.
SharedPreferences	Rajapinta tiedon käyttämiseen ja muokkaamiseen.
SQLite Database Browser	Ohjelma SQLite-tietokantojen muokkaamiseen
SQLite	Tietokantajärjestelmä
SQLiteOpenHelper	Apuluokka tietokantojen luomiseen ja versionhallintaan.
startActivity	Androidin funktio, joka käynnistää aktiviteetin.
USB debugging	Älypuhelimen kehittäjätila
Widget	Android-sovelluksen näkyvä komponentti, joka sijoitetaan Android-laitteen kotinäytölle.
View	Androidin luokka, joka tarjoaa käyttöliittymäkomponentit.
ViewGroup	Androidin luokka, jonka objekti määrittelee käyttöliittymäelementtien asettelun näytölle.

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on kehittää sovellusprototyyppi Android-älypuhelimeen. Valitsin tämän aiheen, koska olen jo pitkään ollut kiinnostunut kännykkäsovelluksien ohjelmoinnista ja sovelluksen toteuttamisesta ensisijaisesti omiin tarpeisiin ja sitä kautta mahdollisesti myös julkaisukelpoiseksi sovellukseksi.

Tavoitteena on tehdä yksinkertainen prototyyppi, joka on mahdollisimman helppokäyttöinen. Työn tarkoituksena on tutustua Android-ohjelmointiin ja syventää ohjelmointitaitoja osittain tutussa, osittain uudessa ympäristössä, sekä antaa kokemusta sovelluksen tekemisestä. Työssä keskitytään tiedon tallentamiseen. Työkaluina käytetään Eclipseä ja Android SDK:ta. Prototyyppi kirjoitetaan java-ohjelmointikieltä käyttäen.

Raportissa kerrotaan, kuinka idea opinnäytetyöhön syntyi ja valotetaan työn suunnitteluvaihetta. Pääosassa on prototyypin toteutus. Ohjelmallisen toteutuksen lisäksi raportissa tarkastellaan tiedon tallentamista Android-järjestelmällä.

2 IDEAN SYNTY

Idea sovellukseen syntyi vähitellen. Olen jo pitkään ollut kiinnostunut tiedon tallentamisesta erilaisissa käyttöjärjestelmissä. Minulla on myös paljon harrastuksia, joissa tiedon tallentamista tarvitaan. Käsityöharrastus (neulonta, virkkaus ym.) on yksi näistä harrastuksista. Toiseksi tärkeäksi asiaksi nousi Android-älypuhelimen käyttö, sillä se on jokapäiväistä. Nämä kaksi asiaa yhdessä loivat pohjan Android-sovellukselle, jota voisi hyödyntää käsityöharrastuksen parissa.

Käsityöharrastukseen liittyy olennaisena osana tarvikkeiden hankinta. Koska aktiivisella harrastajalla on yleensä kohtalaisen suuri varasto erilaisia tarvikkeita, jonkinlainen kirjanpito on usein tarpeen. Ajatuksena on, että tämä kirjanpito voidaan hoitaa sovelluksen avulla. Sovelluksella voidaan siis tallentaa tietoa käsityötarvikkeista, tässä tapauksessa neulelangoista. Sovelluksesta olisi helppo tarkistaa, mitä lankoja varastossa on. Lankoja voisi myös poistaa sitä mukaa, kun ne on käytetty johonkin projektiin. Näin tietoja pidettäisiin ajan tasalla.

3 ANDROID

3.1 Yleistä

Android on alun perin Android Inc.:n kehittämä järjestelmä, jonka Google osti vuonna 2005 (1). Järjestelmä on tarkoitettu älypuhelimille ja muille mobiililaitteille. Se sisältää käyttöjärjestelmän, väliohjelmistoja ja käyttäjän perusohjelmia. Androidin lähdekoodi on avointa, joten sille kehittäminen on ilmaista (2). Android on suosituin käyttöjärjestelmä älypuhelimissa (3). Kuukausittaisia käyttäjiä on yli miljardi (4). Ensimmäinen kaupallinen Android-versio julkaistiin vuonna 2008 (5) ja tähän mennessä versioita on julkaistu 20 (6). Uusin versio Lollipop (versionumero 5.0) julkaistiin vuoden 2014 lopulla (7).

3.2 Teknistä tietoa

Teknisesti Android on ohjelmistopino, joka koostuu Linux-ytimestä ja kokoelmasta C/C++-kirjastoja. Linux-ydin toimii käyttöjärjestelmän pohjana, sillä se huolehtii ydinpalveluista, kuten ajureista, muistin hallinnasta ja turvallisuudesta. Android sisältää paljon kirjastoja, kuten erilaiset C/C++-ydinkirjastot, mediakirjaston, tietokantatuen (SQLite) ja grafiikkakirjastot. (8: 15)

Android-sovellukset toimivat Dalvik-virtuaalikoneen päällä (2). Dalvik perustuu javaan, mutta se ei ole java-virtuaalikone. Dalvik on mukautettu Androidille ja se varmistaa, että useat tapahtumat suoritetaan tehokkaasti yhdellä laitteella. (8: 16)

Ohjelmistopinolla on korkeakielinen rajapinta, joka on toteutettu java-kielellä. C/C++-kirjastojen ohjelmallisia kutsuja on siis abstraktoitu javaksi, jolloin ohjelmointi on huomattavasti helpompaa. Tämän vuoksi ohjelmointi toteutetaan javalla.

3.3 Sovellukset

Androidille tehdään paljon sovelluksia. Niitä voi ladata esimerkiksi Google Play -sovelluskaupasta. Lataamista varten tarvitaan Google-tili. Sovelluksia on sekä maksullisia että ilmaisia. Android-puhelin on erittäin muokattava sovellusten kannalta, sillä käyttöjärjestelmän natiivisovellukset ovat korvattavissa sovelluskaupoista

saatavilla sovelluksilla, tai natiivisovelluksia ja ladattuja sovelluksia voi vaikka käyttää rinnakkain. (2.)

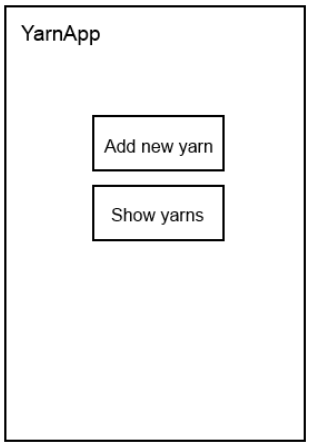
4 TYÖN SUUNNITTELU

Jotta kehitysprosessi etenisi sujuvasti, prototyyppi kannattaa ensin suunnitella pääpiirteisesti. Suunnittelu auttaa valitsemaan oikeanlaiset työkalut. Työympäristön pystyttämisen jälkeen voidaan aloittaa itse toteutus.

4.1 Prototyypin suunnittelu

Prototyypistä on tarkoitus tehdä yksinkertainen. Tärkeimmäksi ominaisuudeksi nousee tiedon tallennus. On siis tarkoitus keskittyä toimintojen toteuttamiseen. Graafisuus jää taka-alalle. Ensimmäiseksi toteutettava toiminto on uuden langan lisääminen. Langan lisäämiseen kuuluu olennaisena osana tietojen tallentaminen. Toinen toteutettava toiminto on lankalistauksen näyttäminen.

Kun käyttäjä käynnistää sovelluksen, näyttöön avautuu sovelluksen päänäkymä (kuva 1). Päänäkymässä ovat painikkeet Add new yarn (lisää uusi lanka) ja Show yarns (näytä langat). Add new yarn -painikkeesta aukeaa toinen näkymä, jossa on langan ominaisuuksille tekstikentät, jotka käyttäjä täyttää. Näkymässä alimpana näkyy painike Save (tallenna). Käyttäjä voi tallentaa syöttämänsä tiedot painamalla Save-painiketta. Tallennuksen jälkeen käyttäjä palaa automaattisesti päänäkymään. Show yarns -painike vie käyttäjän uuteen näkymään, jossa listataan kaikki tietokantaan tallennetut langat.



Kuva 1. Suunnitelma päänäkymälle.

Tallennettava data koostuu merkkijonoista. Kun uusi lanka lisätään, sille määritellään nimi (name), tuottaja (manufacturer), väri (color) ja materiaali (material).

4.2 Työvälineiden valinta

Projekti toteutetaan normaalissa PC-laiteympäristössä. Käyttöjärjestelmänä on Windows. Työvälineiden valinta perustui välineiden ilmaisuuteen ja helppokäyttöisyyteen sekä olemassaolevaan dokumentaatioon.

4.2.1 Eclipse IDE

Eclipse on ilmainen alustariippumaton ohjelmointiympäristö (9). Se tukee muun muassa javaa, C:tä ja C++:aa (10: 1). Eclipse valittiin, koska se tukee java-ohjelmointia. Java on työn tekijälle valmiiksi tuttu kieli, jolloin uutta ohjelmointikieltä ei tarvitse opetella. Toinen syy Eclipsen valinnalle oli se, että sille löytyy paljon ohjeita. Ongelmatilanteissa apua voi helposti etsiä monilta sivustoilta, jotka ovat erikoistuneet Eclipsen ohjeistukseen. Myös Eclipse itsessään on jokseenkin tuttu tekijälle, jolloin projektin tarkoituksena on myös syventää aiemmin opittuja asioita.

Suurin syy Eclipsen valinnalle oli se, että sen kanssa pystyy käyttämään Android SDK:ta. Verkosta löytyy oppaita Eclipsen käyttöön Android-ohjelmoinnissa, ja aiheesta on kirjoitettu kirjojakin. Tässä projektissa käytetään Eclipseä, joka kuuluu Androidin sivuilta saatavaan ADT-pakettiin.

4.2.2 Android SDK

Android SDK on paketti, joka sisältää Android-kehittäjän tarvitsemat työkalut. Pakettiin kuuluu Androidin API-kirjastot, kehittäjän työkalut, Androidin virtuaalilaitteiden hallinta, täysi dokumentaatio ja esimerkkikoodit. Androidilla on myös laaja kehittäjäkunta, jonka ansiosta verkosta löytyy paljon apua aloitteleville Android-kehittäjille. (8: 14)

Tässä työssä käytetään ADT-pakettia (ADT Bundle), jonka saa ladattua Androidin kotisivuilta kehittäjäosiosta. ADT-paketti sisältää Eclipsen version, jossa on sisäänrakennettu ADT (ADT tulee sanoista Android Development Tools eli Android-kehityksen työkalut) sekä olennaiset Android SDK -komponentit. (11.)

5 ALKUVALMISTELUT

5.1 JDK:n ja Eclipsen asennus

Koska Eclipse on javapohjainen ohjelma, se vaatii toimiakseen jonkin java-virtuaalikoneen. Koska Eclipseä käytetään tässä projektissa javakehitykseen, tietokoneeseen asennetaan virtuaalikone JDK (Java Development Kit). JDK sisältää muun muassa lähdekoodin javan standardikirjastoille (12).

Tätä projektia varten asennetaan uusin JDK, joka on opinnäytetyön tekohetkellä JDK 8. JDK:n saa ladattua Oraclen sivuilta (13). Asennus aloitetaan kaksoisklikkaamalla ladattua exe-tiedostoa. JDK on helppo asentaa ohjatun asennusohjelman avulla.

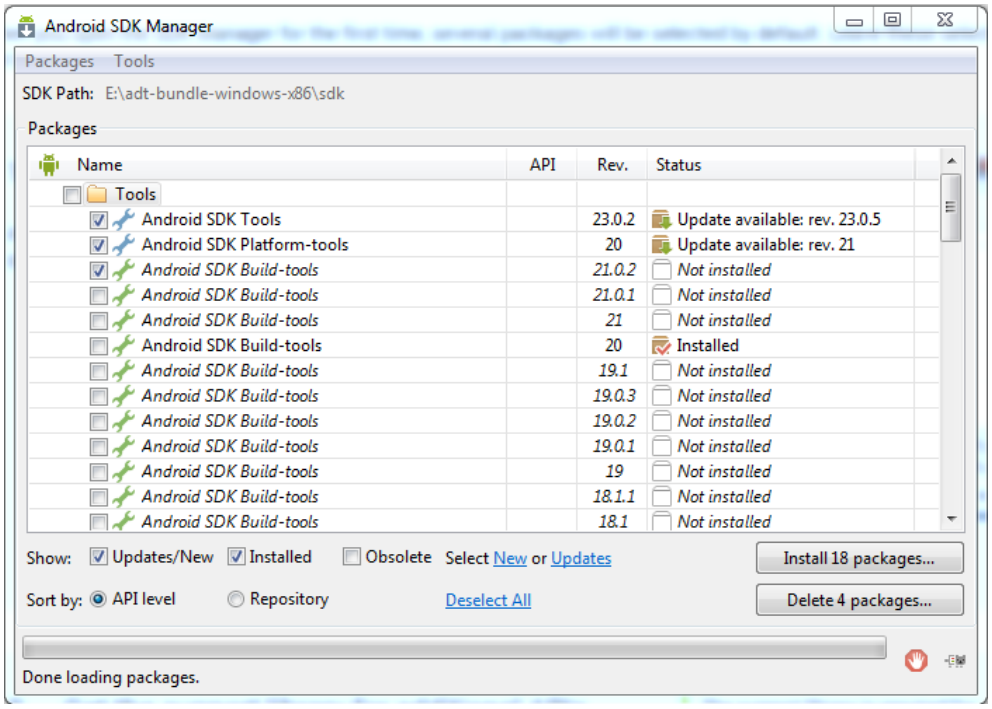
JDK:n jälkeen asennetaan Eclipse. Tässä projektissa käytetään Androidin sivuilta ladattua Eclipse-pakettia. Eclipsen asennus on yksinkertainen. Ladattu zip-paketti puretaan haluttuun kansioon ja näin Eclipse on asennettu.

5.2 SDK-pakettien lisääminen Eclipsessä

Android SDK ei sisällä valmiiksi kaikkea mitä sovelluskehittämiseen tarvitaan. Loput paketit ladataan ja asennetaan Eclipsen kautta Android SDK Managerin avulla. Ensin käynnistetään Eclipse ja sieltä avataan Android SDK Manager.

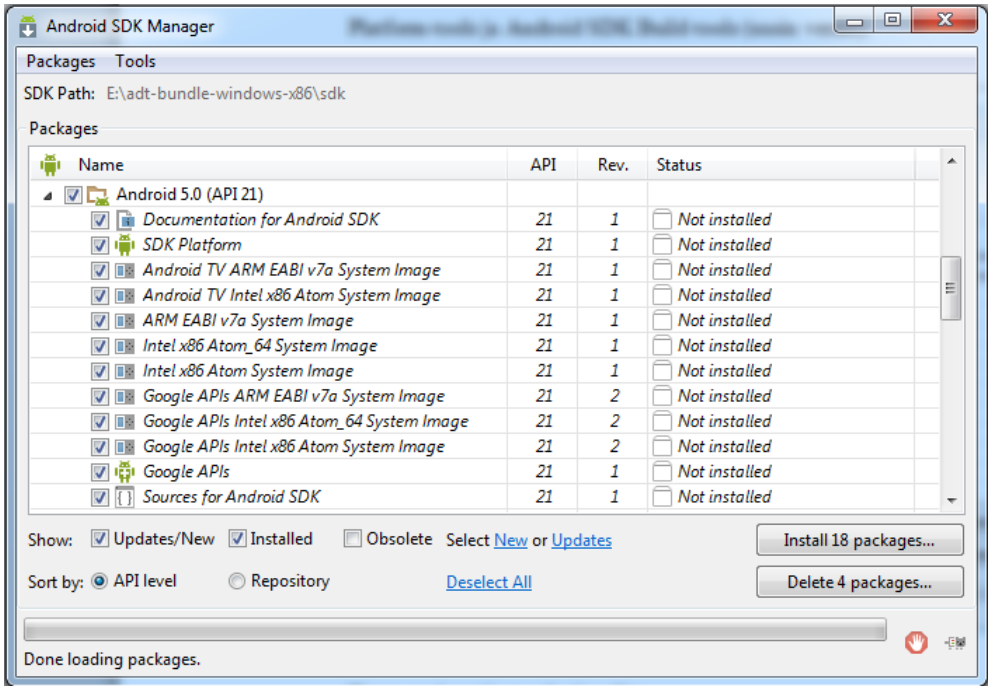
5.2.1 Uusimpien SDK-työkalujen hakeminen

Tools-kansiosta valitaan seuraavat työkalut: Android SDK Tools, Android SDK Platform-tools ja Android SDK Build-tools (uusin versio eli 21.0.2) (kuva 2).



Kuva 2. Ensimmäiset paketit valittuna.

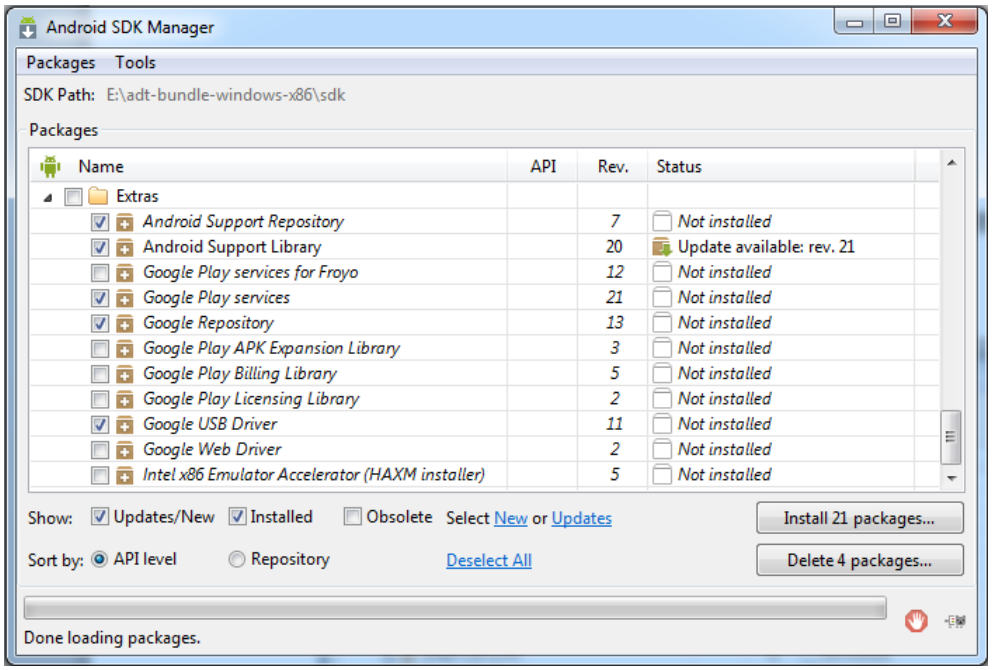
Seuraavaksi avataan Android X.X-kansio (tässä työssä Android 5.0, joka on uusin versio) ja valitaan SDK Platform ja järjestelmä-image emulaattoria varten. Ne ovat yleensä jo valmiiksi valittuna, ja loput valinnat voi jättää paikoilleen (kuva 3).



Kuva 3. Uusimman API-tason paketit.

5.2.2 Android Support Library ja Google Play services

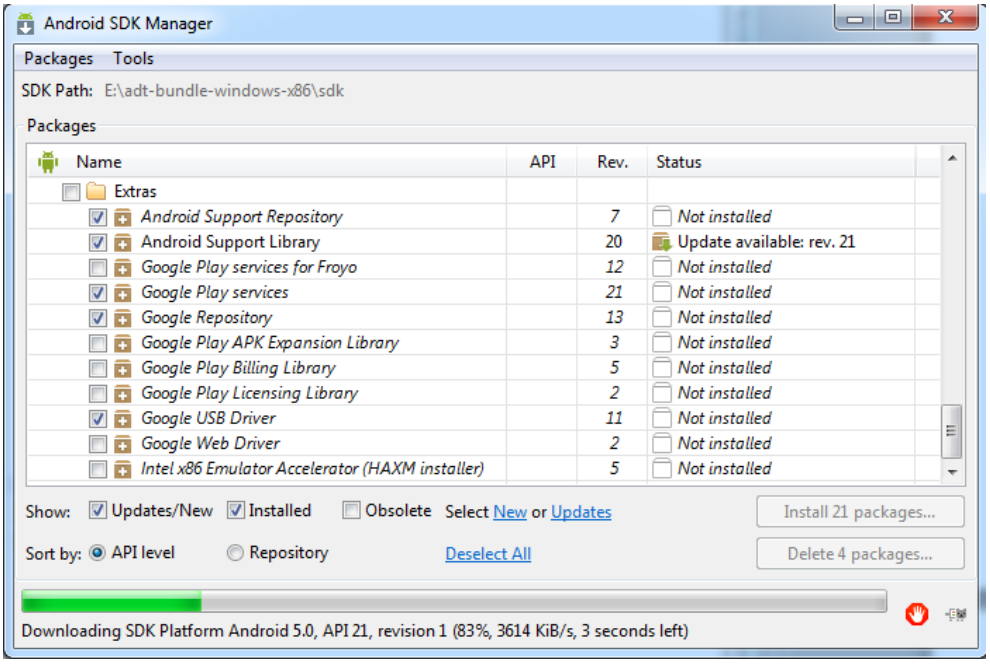
Android Support Library tarjoaa laajan kokoelman API-tasoja, jotka sopivat yhteen usean Android-version kanssa. SDK Managerista avataan Extras-kansio ja sieltä valitaan Android Support Repository ja Android Support Library (kuva 4). Jotta kehityksessä voitaisiin käyttää Googlen API-tasoja, täytyy asentaa myös Google Play Services -paketti. Valitaan samasta kansiosta Google Repository ja Google Play services.



Kuva 4. Extras-kansion valinnat.

5.2.3 Pakettien asentaminen

Kun kaikki tarvittavat paketit on valittu, niiden asennus voidaan aloittaa valitsemalla Install (asenna). Android SDK Manager lataa ja asentaa paketit (kuva 5).



Kuva 5. Pakettien lataus ja asennus.

Kehitykseen tarvittavat ohjelmistot on nyt asennettu. Jotta kehityksessä päästään heti alkuun, on selvitettävä kuinka Android-laitteen saa kytkettyä Eclipseen.

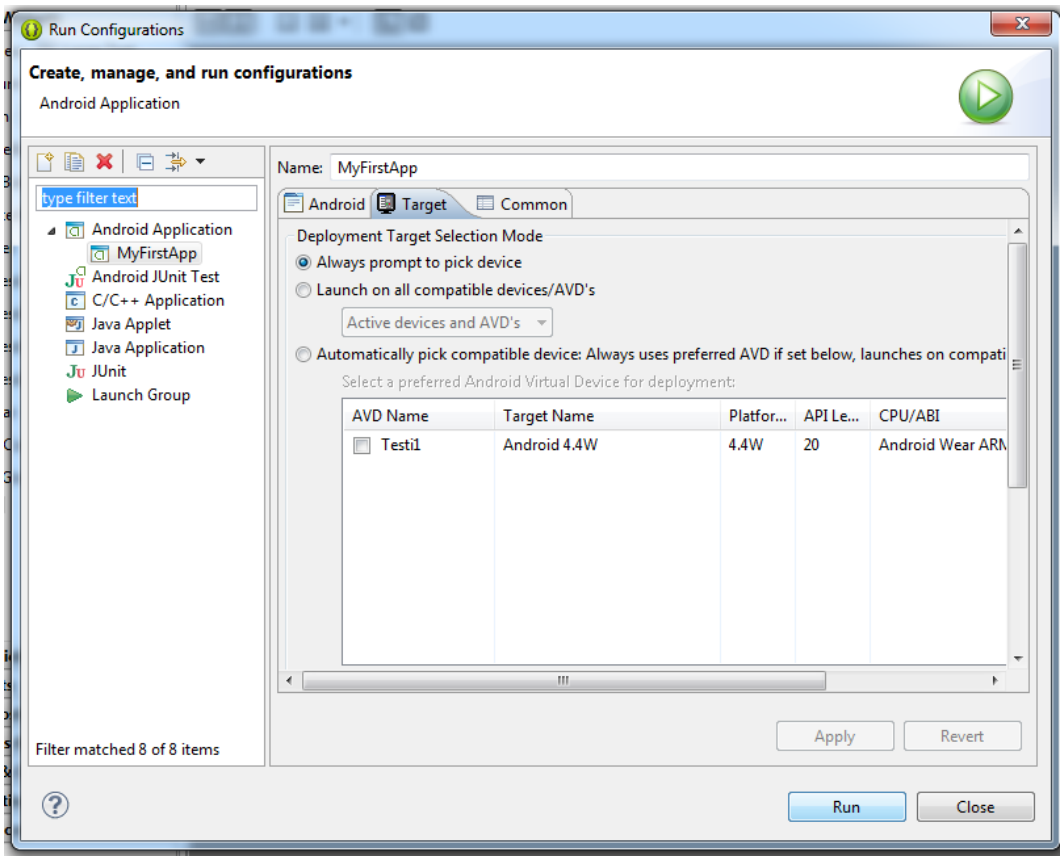
5.3 Android-laitteen yhdistäminen Eclipseen

Android-laite on kytkettävä Eclipseen, jotta kehityksessä oleva sovellus voidaan aina suorittaa realistisessa ympäristössä. Työhön valittiin Android-puhelin, jossa on käyttöjärjestelmänä Androidin versio 2.3. Kyseinen laite valittiin, koska työn tekijällä on laite käytettävissään ja koska sovellus tulee ensisijaisesti puhelimelle. Käyttämällä versiota 2.3 voidaan taata, että sovellus toimii kaikissa puhelimissa, joissa on versio 2.3 tai sitä uudempi versio. Näin sovellus toimii mahdollisimman monessa Android-puhelimessa.

Windows-käyttöjärjestelmällä työskenneltäessä on asennettava erilliset ajurit, jotta Android-laitteen saa kytkettyä Eclipseen (14). Ajureita tarvitaan, kun laite kytketään tietokoneeseen ja tietokoneesta siirretään tietoa laitteeseen (15).

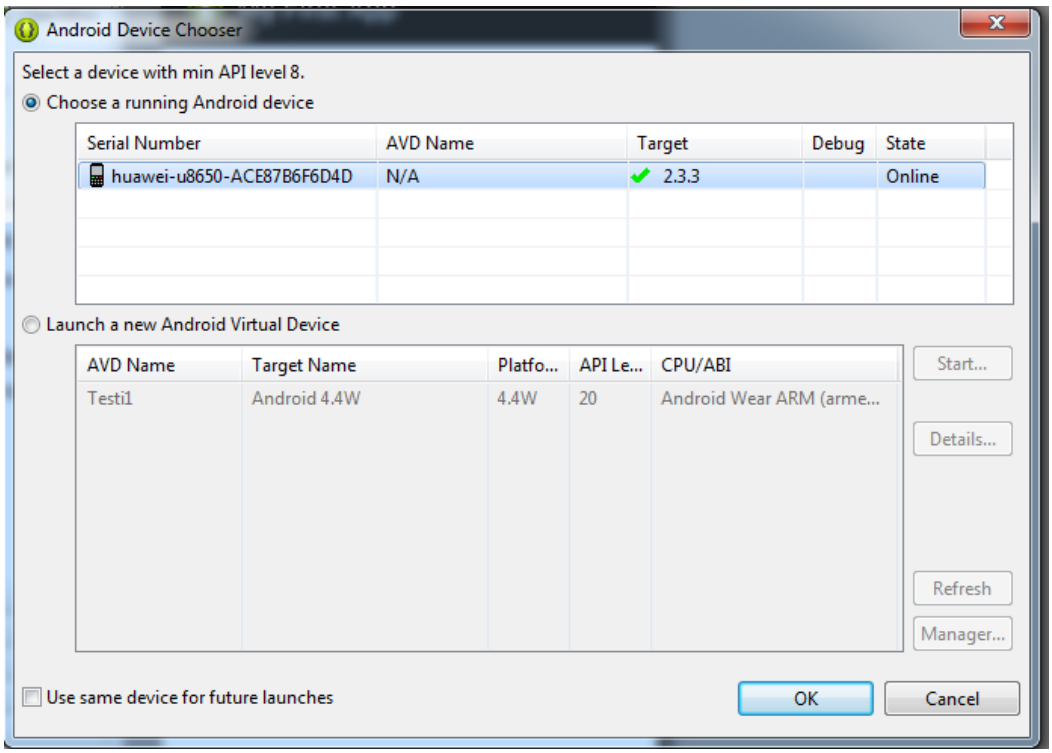
Laitteen yhdistämisessä käytetään yksinkertaista hello world -testiprojektia, joka tuli Android SDK:n mukana. Ajurien etsiminen vanhalle puhelimelle oli haastavaa, mutta ne löytyivät lopulta Team Androidin sivuilta (15). Latauksen jälkeen ajurit asennetaan niin, että asennuspaketti puretaan ja sieltä suoritetaan DriverSetup.exe.

Android-puhelin kytketään tietokoneeseen USB-kaapelin avulla. Puhelimen asetuksista kytketään USB debugging päälle, jotta tietokoneeseen yhdistäminen on mahdollista. Tämän jälkeen palataan Eclipseen, jossa hello world -projekti on avattuna. Eclipsestä valitaan Run Configurations Run-valikosta. Aukeavasta ikkunasta tarkistetaan, että valittuna on kyseinen projekti. Seuraavaksi avataan Target-välilehti ja varmistetaan, että Always prompt to pick a device on valittuna, jolloin Eclipse kysyy jokaisella ajokerralla, mikä laite suoritukseen valitaan (kuva 6). Tämä kannattaa olla valittuna, jos kehityksessä on tarkoitus käyttää useita puhelimia tai emulaattoria puhelimen lisäksi. Tämän jälkeen valitaan Run.



Kuva 6. Run configurations.

Seuraavassa ikkunassa valitaan laite näkyvästä listasta ja hyväksytään se valitsemalla ok. Tässä työssä valitaan siis Android-puhelin Huawei U8650, jolle ajurit asennettiin (kuva 7).

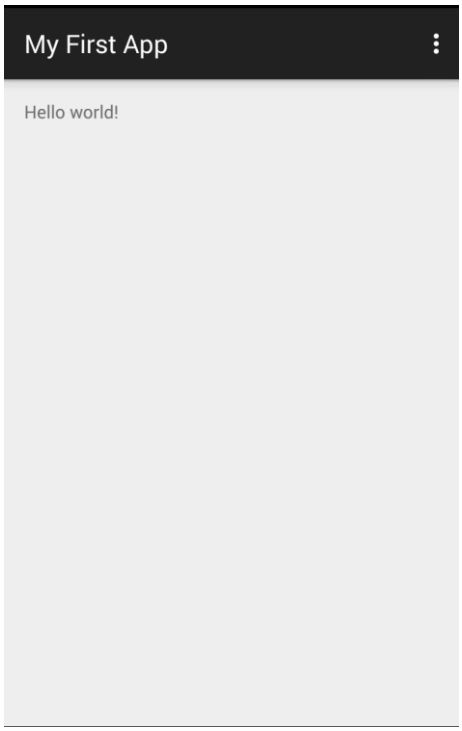


Kuva 7. Laitteen valinta suoritusta varten.

Suoritus alkaa. Eclipse siirtää projektin apk-tiedoston puhelimeen ja asentaa sen siihen (kuva 8). Kun asennus on valmis, sovellus käynnistyy puhelimessa (kuva 9).

```
[2014-08-20 20:34:45 - MyFirstApp] Android Launch!
[2014-08-20 20:34:45 - MyFirstApp] adb is running normally.
[2014-08-20 20:34:45 - MyFirstApp] Performing com.example.myfirstapp.MainActivity activity launch
[2014-08-20 20:35:14 - MyFirstApp] Uploading MyFirstApp.apk onto device 'ACE87B6F6D4D'
[2014-08-20 20:35:16 - MyFirstApp] Installing MyFirstApp.apk...
[2014-08-20 20:35:27 - MyFirstApp] Success!
[2014-08-20 20:35:27 - MyFirstApp] Starting activity com.example.myfirstapp.MainActivity on device ACE87B6F6D4D
```

Kuva 8. Eclipsen konsoli suorituksen aikana.



Kuva 9. Hello world -projektin suoritus Android-puhelimessa.

Kaikki on nyt valmista ohjelmallisen toteutuksen aloittamiseksi. Ensiksi käydään läpi Android-ohjelmoinnin perusasioita.

6 PERUSTEITA ANDROID-OHJELMOINNISTA

Tässä osiossa käydään läpi sovelluksen rakenne sekä uuden Android-projektin luonti ja siihen liittyvät asiat.

6.1 Sovelluksen rakenne

Android-sovellukset koostuvat erilaisista komponenteista. Android Manifest kuvailee jokaisen komponentin ja sen, kuinka ne toimivat. Manifest myös määrittelee sovelluksen metadatan, laitteistovaatimukset, vaaditut luvat ja ulkoiset kirjastot. Seuraavaksi kerrotaan lyhyesti kyseisistä komponenteista. Android Manifestista kerrotaan tarkemmin luvussa 6.3.3. (8: 54)

6.1.1 Aktiviteetit

Sovelluksen käyttöliittymä rakennetaan aktiviteettien avulla. Tiedon asettelu ja näyttäminen sekä käyttäjän toimintoihin vastaaminen kuuluvat aktiviteettien tehtäviin. (8: 54)

6.1.2 Palvelut

Palvelukomponentit toimivat taustalla ilman käyttöliittymää. Niitä käytetään pitkissä tehtävissä sekä sellaisissa, joilla ei ole mitään tekemistä käyttäjän vuorovaikutuksen kanssa. Tällaisia ovat esimerkiksi ne tehtävät, jotka ovat suorituksessa vaikka sovelluksen käyttöliittymä ei ole näkyvissä. (8: 54)

6.1.3 Content Providerit

Content Providerit hallitsevat ja säilyttävät sovellusdataa ja ovat usein vuorovaikutuksessa SQL-tietokantojen kanssa. (8: 54)

6.1.4 Intentit

Intentejä käytetään Androidissa hyvin laajasti. Niillä voidaan esimerkiksi aloittaa ja lopettaa aktiviteetteja ja palveluita sekä lähettää viestejä eri puolille järjestelmää. (8: 54)

6.1.5 Broadcast Receiverit

Nämä komponentit kuuntelevat edellä mainittuja intenttejä. Sovellus kuuntelee niitä intenttejä, jotka kehittäjä on määritellyt kuunneltaviksi. (8: 55)

6.1.6 Widgetit

Widgetit ovat näkyviä sovelluksen komponentteja, joita yleensä lisätään Android-laitteen kotinäytölle. Niillä voidaan luoda entistä sujuvampi käyttökokemus. (8: 55)

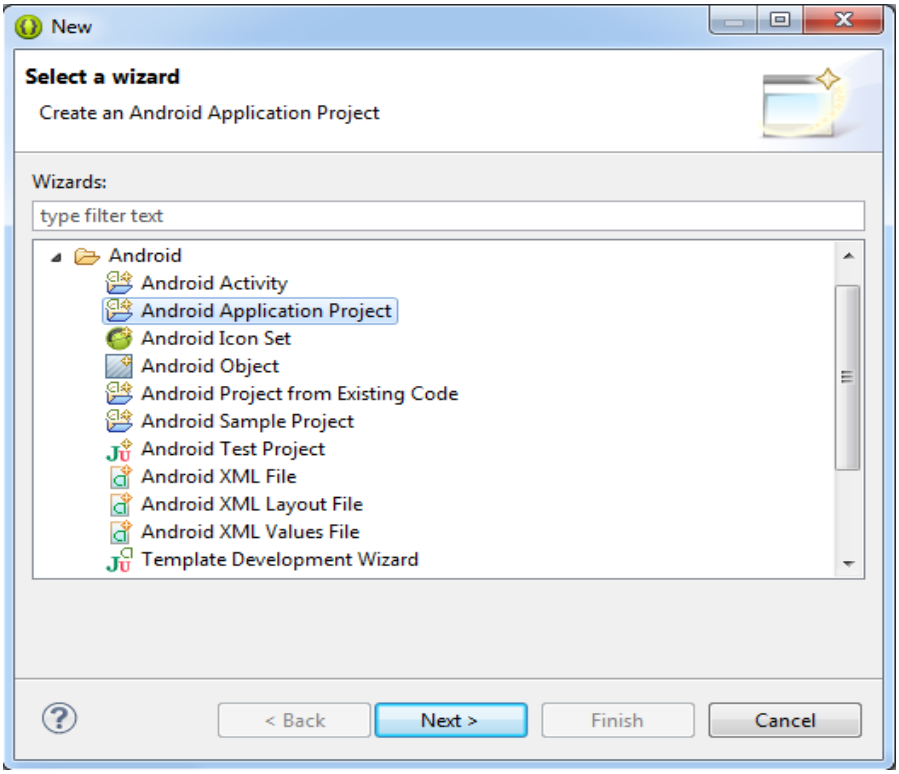
6.1.7 Notifikaatiot

Notifikaatioiden avulla käyttäjää voidaan hälyttää erilaisista sovelluksessa tapahtuvista asioista ilman, että sovellus on aktiivisena tai näkyvänä näytöllä. (8: 55)

6.2 Uuden Android-projektin luominen Eclipsessä

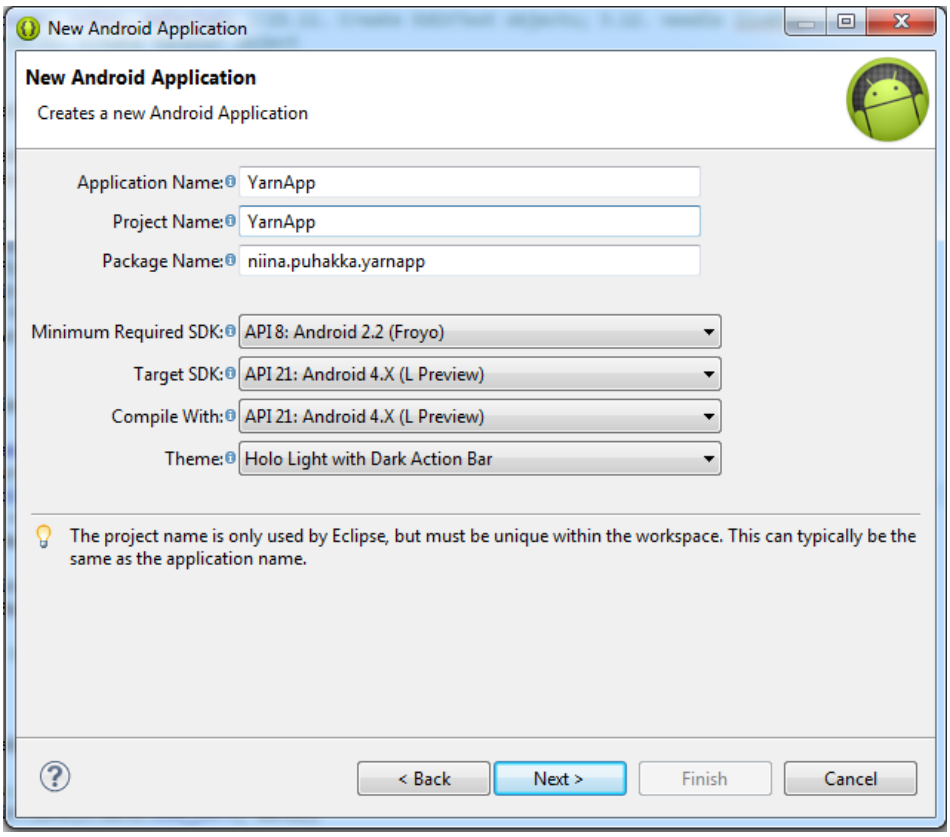
Eclipsessä Android-projektin luonti aloitetaan painamalla New-kuvaketta Eclipsen vasemmassa yläkulmassa. Siitä avautuu ikkuna, josta avataan Android-kansio.

Kansion alle avautuu eri projektivaihtoehtoja. Niistä valitaan Android Application Project (kuva 10) ja painetaan Next.



Kuva 10. Projektityypin valinta.

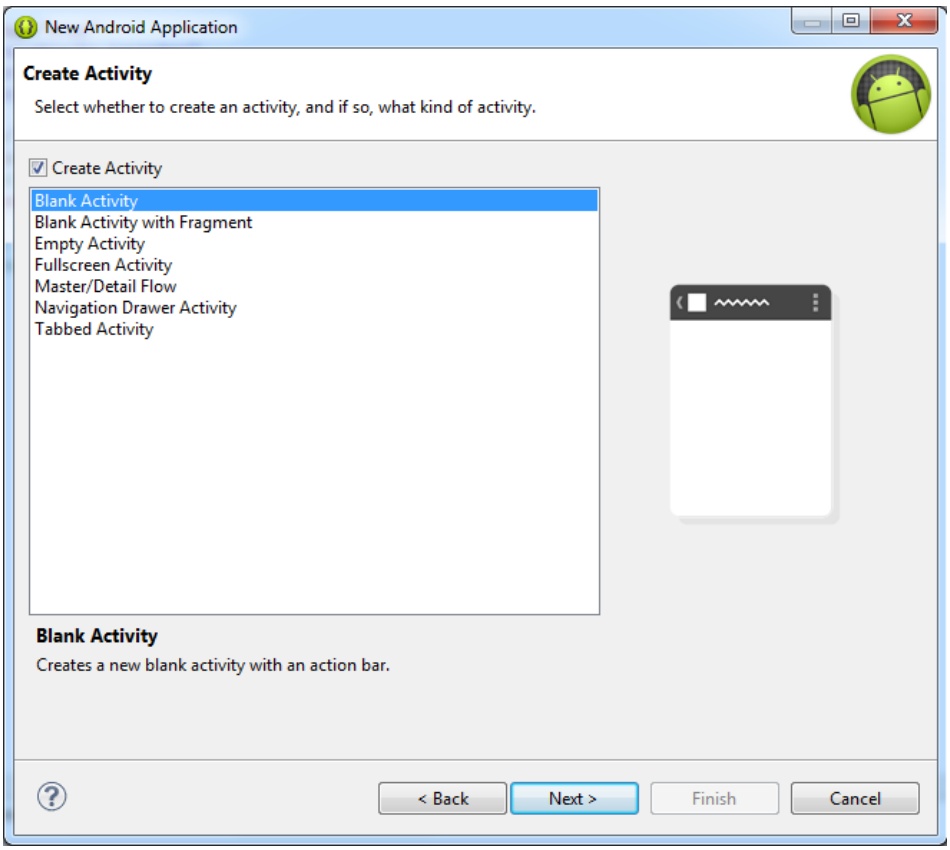
Seuraavassa ikkunassa täytetään tarvittavat tiedot. Sovellukselle, projektille ja paketille annetaan nimet. Sovelluksen nimi on se, joka näkyy Android-käyttäjälle. Projektin nimi on projektikansion nimi, joka näkyy Eclipsessä. Paketin nimen tulee olla uniikki kaikkien muiden pakettien keskuudessa, jotka on asennettu Android-järjestelmään. Tässä työssä sovelluksen nimi on YarnApp (suomeksi lankasovellus), projektin nimenä käytetään samaa nimeä ja paketin nimeksi annetaan niina.puhakka.yarnapp (kuva 11).



Kuva 11. Uuden Android-sovelluksen luonti.

Ikkunassa valitaan myös alkeellisin Android-versio, jota tuleva sovellus tukee (Minimum Required SDK). Tähän kannattaa valita vanhin Android-versio, joka on saatavilla, jotta sovellus toimii myös vanhemmissa Android-puhelimissa. Target SDK tarkoittaa uusinta Android-versiota, jota tuleva sovellus tukee. Tähän kannattaa valita uusin versio, joka on saatavilla, jotta kehityksessä on mahdollista hyödyntää uusimman Android-version ominaisuudet. Sitten jatketaan painamalla Next.

Seuraavassa ikkunassa voidaan valita, luodaanko sovellukselle mukautettu käynnistyskuvake ja luodaanko projektinluonnin yhteydessä ensimmäinen aktiviteetti. Projektin tallennussijainniksi on mahdollista valita joko valmiiksi määritelty Eclipsen workspace tai jokin muu sijainti. Seuraavaksi luodaan ensimmäinen aktiviteetti. Seuraavassa ikkunassa (kuva 12) määritellään, millainen aktiviteetti luodaan. Tähän projektiin valitaan Blank Activity, jossa valmiina on ainoastaan action bar.



Kuva 12. Aktiviteetti.

Seuraavaksi aktiviteetille sekä sen asettelutiedostolle annetaan nimet. Oletuksena nimiksi tulee MainActivity ja activity_main. Tässä työssä käytetään oletusnimiä. Lopuksi painetaan Finish (lopeta). Uusi Android-projekti sekä aktiviteetti on nyt luotu. Seuraavaksi tutkitaan, mitä tiedostoja kyseinen projekti sisältää.

6.3 Android-projektin tiedostot

6.3.1 Asettelutiedosto activity_main.xml

Tämä tiedosto sisältää xml-asettelun aktiviteettitiedostoa varten (16). Tässä vaiheessa tiedosto sisältää ainoastaan tekstin ”Hello world!”. Asettelusta kerrotaan tarkemmin luvussa 7.1.

6.3.2 Aktiviteettitiedosto MainActivity.java

MainActivity-tiedosto vastaa sovelluksen toiminnallisuudesta. Kun sovellus käynnistetään, Activity-luokka käynnistää tämän kyseisen aktiviteetin ja lataa edellä mainitun asettelutiedoston, jolloin näyttöön ilmestyy teksti ”Hello world!”. (16.)

6.3.3 Manifest-tiedosto AndroidManifest.xml

Jokaisessa Android-projektissa on AndroidManifest.xml-tiedosto. Manifest tiedottaa Android-järjestelmälle kaikki sovelluksen tärkeimmät tiedot. Nämä tiedot ovat mm. sellaisia, jotka järjestelmän on tiedettävä ennen kuin se voi alkaa suorittaa sovelluksen ohjelmakoodia. Manifest myös kuvailee sovelluksen komponentit, määrittelee mitkä prosessit isännöivät kyseisiä komponentteja, määrittelee sovelluksen tarvitsemat luvat erilaisia tehtäviä varten (esim. vuorovaikutus muiden sovellusten kanssa) sekä määrittelee alhaisimman API-tason, jonka sovellus vaatii toimiakseen oikein. (17.)

6.4 Valmiit funktiot

Android-projektissa on valmiina muutamat funktiot, jotka vastaavat sovelluksen perustoiminnoista. Ne ovat nimeltään onCreate, setContentView, onCreateOptionsMenu ja onOptionsItemSelected.

Aktiviteetti alustetaan onCreate-funktiossa. Funktion sisällä on setContentView, joka määrittelee sovelluksen käyttöliittymän. OnCreateOptionsMenu alustaa Activity-luokan standardi-valikon, joka näkyy näytöllä, kun aktiviteetti on käynnissä. OnOptionsItemSelected-funktiota kutsutaan aina, kun kyseisestä valikosta valitaan jokin kohde. (18.)

7 PROTOTYYPIN PÄÄPIIRTEET

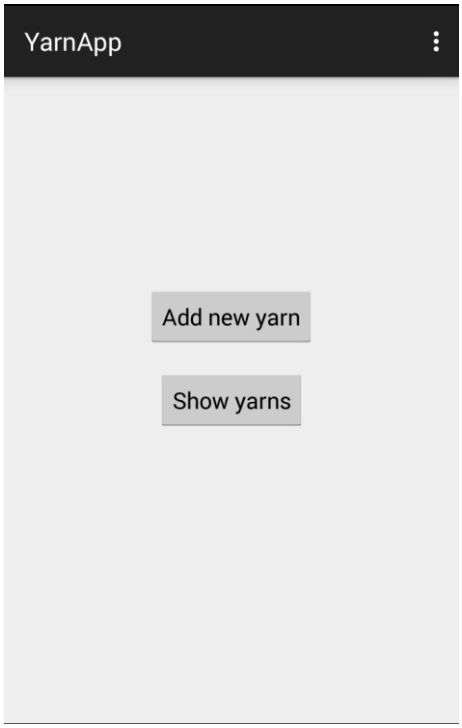
7.1 Käyttöliittymän asettelu

Sovelluksen toteutus aloitetaan käyttöliittymän rakentamisesta. Android-sovelluksen kaikki käyttöliittymäelementit rakennetaan käyttämällä View- ja ViewGroup-objekteja. View-objekti piirtää näytölle jotakin, joka vastaa käyttäjän toimintoihin (esim. painikkeen). ViewGroup-objekti määrittelee sen, miten käyttöliittymäelementit asettuvat näytölle. View- ja ViewGroup-luokilla on aliluokat, jotka tarjoavat tyypilliset käyttäjän syötteeseen tarkoitetut elementit (esim. painike ja tekstikenttä) sekä erilaisia asettelumalleja (esim. lineaarinen tai relatiivinen asettelumalli). (19.)

Sovelluksen käyttöliittymä on helpointa määritellä XML-tiedoston kautta. XML tarjoaa helposti luettavan rakenteen asettelua varten (19).

7.1.1 Päänäkymä

Ensin toteutetaan prototyypin päänäkymä eli asettelu aktiviteetille, joka luotiin projektin yhteydessä. Päänäkymään tulee kaksi painiketta (button): Add yarn ja Show yarns (kuva 13).



Kuva 13. Prototyypin päänäkymä.

Päänäkymän asettelussa käytetään relatiivista asettelumallia (RelativeLayout), koska sen avulla elementit on helppo sijoittaa haluttuihin paikkoihin (20). Add new yarn -painike aloittaa uuden aktiviteetin Add Yarn Activity, jolla lisätään uusi lanka. Kyseisen painikkeen attribuutteihin lisätään onClick-attribuutti. Tämän attribuutin avulla Add new yarn -painikkeelle määritellään metodi nimeltä addYarn (listaus 1) (21).

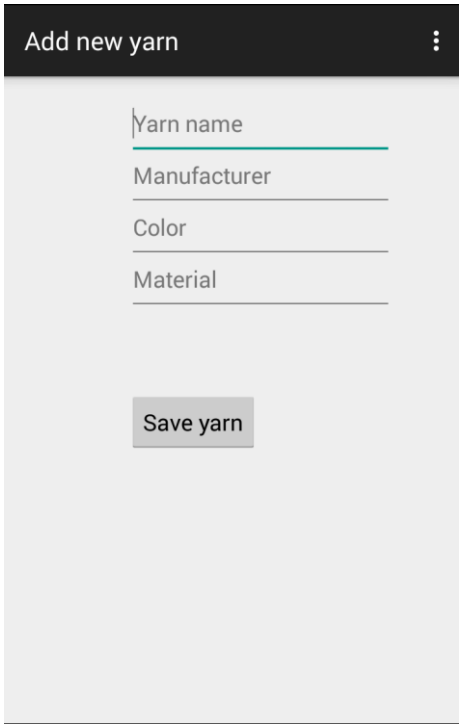
```
<Button
    android:id="@+id/add"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/show"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="18dp"
    android:onClick="addYarn"
    android:text="@string/button_add_yarn" />
```

Listaus 1. OnClick-attribuutti.

Kun käyttäjä painaa Add new yarn -painiketta, käyttäjä viedään uuteen näkymään, jossa uuden langan lisääminen tapahtuu.

7.1.2 Uuden langan lisääminen

Uuden langan lisäämistä varten tarvitaan uusi aktiviteetti Add Yarn Activity. Tälle aktiviteetille tehdään yksinkertainen asettelu. Asetteluun tulee täytettävät tekstikentät yarn name (langan nimi), manufacturer (langan tuottaja), color (langan väri) ja material (langan materiaali). Alimpana näkymässä on Save yarn -painike, jota painamalla käyttäjän kirjoittamat tiedot on mahdollista tallentaa. (kuva 14)



Kuva 14. Asettelu langan lisäämistä varten.

Tässäkin asettelussa käytetään relatiivista asettelumallia. Täytettävät tekstikentät ovat EditText-elementtejä. Ne ottavat vastaan käyttäjän syötteitä (22).

7.2 Aktiviteetit

7.2.1 Main activity

Pääaktiviteetin addYarn-metodi käynnistää uuden aktiviteetin nimeltä AddYarnActivity. Aktiviteetin käynnistämiseen käytetään Intent-luokkaa, joka toimii startActivity-metodin kanssa käynnistäen uuden aktiviteetin (listaus 2) (23).

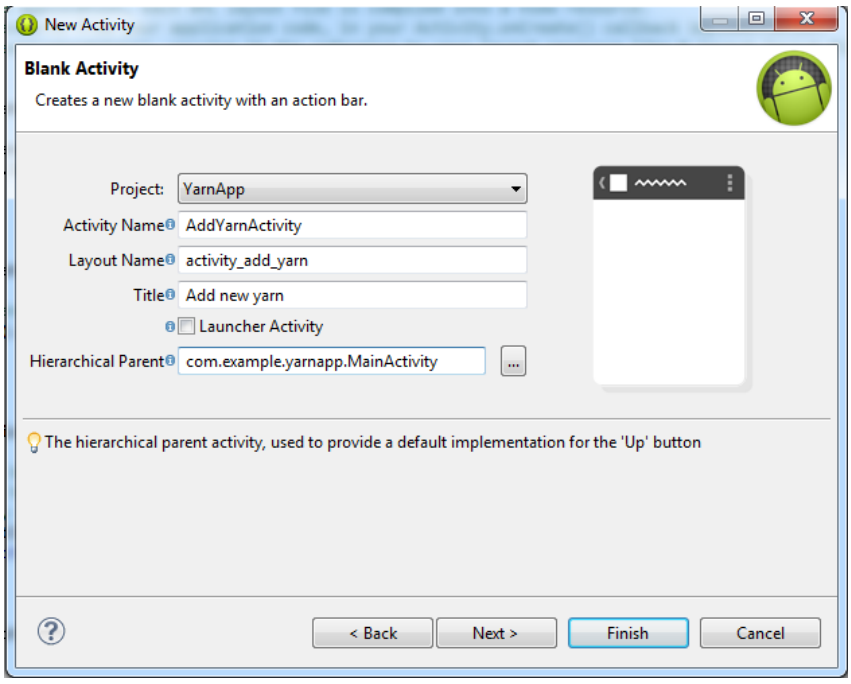
```
/** Called when the user clicks the Add yarn button */
   public void addYarn(View view)
   {
      // Create an Intent to start an activity called AddYarnActivity
                  Intent addyarn = new Intent(this, AddYarnActivity.class);
                  startActivity(addyarn);
   }
```

Listaus 2. Uuden aktiviteetin aloittaminen Intentillä.

Ohjelmakoodin toimiminen edellyttää, että AddYarnActivity-luokka on luotu projektiin.

7.2.2 Add Yarn Activity

Uusi luokka on mahdollista luoda normaalina luokkana tai aktiviteettina. Koska uuden langan lisäämiseen tarvitaan oma asettelu (ks. luku 7.1.2), tässä tapauksessa luodaan aktiviteetti. Eclipsen vasemmasta yläkulmasta valitaan New ja auenneesta ikkunasta Android-kansiosta Android Activity. Seuraavaksi valitaan aktiviteetin malli ja lopuksi aktiviteetille ja sen asettelutiedostolle annetaan nimet (kuva 15). Samalla on hyvä varmistaa, että aktiviteetti liitetään oikeaan projektiin.



Kuva 15. Uuden aktiviteetin luonti.

AddYarnActivity lisää uuden langan. Metodi, joka suorittaa lisäämisen on nimeltään saveYarn. SaveYarn voidaan toteuttaa vasta sitten, kun tiedon tallentamisesta tiedetään enemmän (luku 9.4.4.).

8 TIEDON TALLENTAMINEN ANDROIDILLA

Jotta tiedettäisiin varmasti, mikä menetelmä tietojen tallentamiseen kannattaa valita, vaihtoehtoja on käytävä läpi. Suurin osa Android-sovelluksista tallentaa tietoa jollakin tavalla. Kun käyttäjä käyttää jotakin sovellusta, hänen edistyksensä pitää pysyä tallessa käytön aikana. Monissa sovelluksissa käyttäjän asetukset on tallennettava, ja joissakin sovelluksissa täytyy tallentaa paljon tiedostoja ja käyttää isoja tietokantoja tiedon tallentamiseen. (24.)

Tiedon tallentamiseen on kolme erilaista menetelmää. Menetelmä valitaan tarpeiden mukaan. Valintaan vaikuttaa esimerkiksi se, pitääkö datan olla yksityistä vai pitääkö muiden sovelluksien tai itse käyttäjän päästä siihen käsiksi. Muita vaikuttavia tekijöitä ovat datan suuruus ja se, millaista tallennettava data on. (25.) Seuraavaksi käydään läpi eri menetelmät.

8.1 Tiedostojen tallentaminen

Android käyttää samankaltaista tiedostojärjestelmää kuin levypohjaiset tiedostojärjestelmät käyttävät. Android-laitteilla on kaksi muistialuetta: sisäinen ja ulkoinen muisti. Nimitykset tulevat Androidin alkuajoilta, sillä suurin osa laitteista tarjosi sisäänrakennetun muistin ja irrotettavan muistin (esim. muistikortti). Joissakin laitteissa muistitila on jaettu sisäiseen osioon ja ulkoiseen osioon. Tällöin laitteessa on silti kaksi muistia, vaikka siinä ei olisikaan irrotettavaa muistia. (26.) Seuraavissa kappaleissa kuvataan sisäisen ja ulkoisen muistin ominaisuudet tiivistetysti.

8.1.1 Sisäinen muisti

Sisäinen muisti on aina saatavilla. Siihen tallennetut tiedostot ovat yleensä saatavilla vain sen sovelluksen kautta, jossa ne on tallennettu. Kun käyttäjä poistaa sovelluksen, myös kaikki sovelluksen tiedostot sisäisestä muistista poistetaan. Sisäistä muistia kannattaa käyttää silloin, kun halutaan, että muut sovellukset tai käyttäjä eivät pääse muokkaamaan tiedostoja. (26.)

8.1.2 Ulkoinen muisti

Ulkoinen muisti ei välttämättä ole aina saatavilla, sillä käyttäjä voi halutessaan irrottaa muistikortin laitteesta. Ulkoisen muistin tiedostoja voidaan usein lukea ulkopuolelta (muut sovellukset, käyttäjä). Kun käyttäjä poistaa sovelluksen, ulkoisessa muistissa olevat sovelluksen tiedostot poistetaan vain, jos ne on tallennettu tiettyyn sijaintiin. Ulkoista muistia kannattaa käyttää silloin, kun tiedostoja pitää voida jakaa muiden sovellusten kanssa tai käyttäjän pitää voida esim. kopioida tiedostoja toiseen sijaintiin. (26.)

8.2 Avain-arvoparien tallentaminen (SharedPreferences)

Jos tallennettavana on suhteellisen pieni joukko avain-arvopareja, menetelmäksi kannattaa valita SharedPreferences (27). Se on ohjelmointirajapinta (API), jonka avulla voidaan tallentaa ja hakea perustietotyypeistä koostuvia avain-arvopareja: boolean, float, int, long ja string (25). SharedPreferences-API:a kannattaa käyttää esimerkiksi käyttäjän asetusten tallentamiseen.

8.3 SQL-tietokannat

Jos tallennettava data on toistuvaa ja strukturoitua, tietokantaan tallentaminen on paras vaihtoehto (28). Androidilla on täysi tuki SQLite-tietokantoihin (25). Esimerkiksi jos sovellukseen on tarkoitus tallentaa monen henkilön yhteystiedot (esim. nimi, puhelinnumero, sähköposti), tällöin kannattaa käyttää SQLite-tietokantaa.

8.3.1 SQLite

SQLite on avoimen lähdekoodin tietokantajärjestelmä. Sen käyttäminen on ilmaista. SQLiteä kehittää kansainvälinen tiimi, joka työskentelee täysipäiväisesti SQLiten parissa. SQLite on suosittu tietokantamoottori esimerkiksi kännyköissä ja MP3-soittimissa, koska se ei vie paljoa muistia. (29.)

SQLitellä on kompakti kirjasto. Kirjaston koko voi olla pienempi kuin 500 kilotavua, vaikka kaikki ominaisuudet olisivat käytössä. (29.) SQLite tarvitsee vain vähän ulkoisten kirjastojen tai käyttöjärjestelmän tukea (30). SQLitellä luotu tietokanta sisältyy yhteen ainoaan tiedostoon, jonka pääte on db. SQLitellä ei ole erillistä serveriprosessia, toisin kuin suurimmalla osalla SQL-tietokannoista on. Tämä tarkoittaa sitä, että SQLite lukee ja kirjoittaa suoraan alkuperäisiin levyn tiedostoihin. (29.) Kun erillistä serveriprosessia ei ole, sitä ei tarvitse asentaa ja käynnistää erikseen, jolloin säästyy aikaa. Tietokannan käytössä ei myöskään tarvita mitään järjestelmänvalvojan lupia, jolloin käyttö on yksinkertaisempaa. Huono puoli on kuitenkin se, että bugeja saattaa esiintyä enemmän. Tämä siksi, että serveriprosessi osaltaan suojaa joiltakin bugeilta. (31.)

SQLiten käyttöönotto on helppoa, sillä sitä ei tarvitse asentaa erikseen mitenkään (32). Käyttö on myös luotettavaa, sillä SQLite testataan tarkasti ennen uuden version julkaisemista (29).

8.4 SQLite vai SharedPreferences?

Tätä työtä suunnitellessa tiedostojen tallennus karsiutuu toistaiseksi pois ja jäljelle jäävät SQLite ja SharedPreferences. Kumpi näistä sitten kannattaa valita?

SharedPreferences sopii pienelle datamäärälle. Sen avulla datan tallentaminen ja sen lukeminen on helppoa. Tässä työssä kerralla tallennettavaa dataa ei ole paljon (nimi, tuottaja, väri ja materiaali), joten avain-arvo -parien tallentaminen voisi hyvinkin riittää. Avain-arvo -parien tallentaminen voi olla nopeampaa kuin tallentaminen SQLite-tietokantaan.

Sovelluksessa on kuitenkin tarkoitus tallentaa useita lankoja, joilla kaikilla on oma nimi, tuottaja, väri ja materiaali. Tallennettava data on siis jäsennettyä ja toistuvaa, ja sen määrä kasvaa sitä mukaa kun lankoja lisätään. SQLite sopii tällaiseen dataan paremmin kuin SharedPreferences.

Käytettäväksi valitaan siis SQLite-tietokanta, koska se on näistä kolmesta vaihtoehdosta sopivin.

9 TIETOKANNAN TOTEUTUS

9.1 Tietokannan suunnittelu

Aluksi suunnitellaan tietokannan rakenne. Langan ominaisuudet tallennetaan taulukkoon, jossa ominaisuudet ovat luonnollisesti sarakkeissa (taulukko 1).

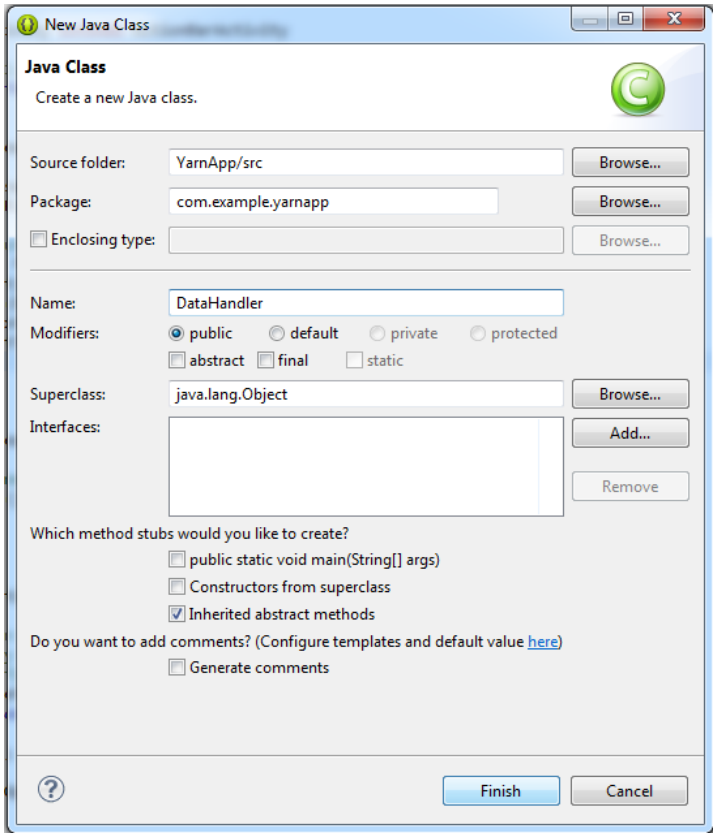
Name	Manufacturer	Color	Material

Taulukko 1. Tietokannan taulukon rakenne.

9.2 Tietokantaluokan ja aliluokan luominen

Suositeltu tapa SQLite-tietokannan luomiseen on luoda SQLiteOpenHelper-luokan aliluokka (25). SQLiteOpenHelper on apuluokka tietokannan luomiseen ja versionhallintaan (33). Ensin luodaan julkinen luokka DataHandler tietokannan ohjelmointia varten. DataHandler-luokkaan luodaan yksityinen SQLiteOpenHelperin aliluokka DBHelper.

Tietokantaluokka luodaan normaalina luokkana (ei aktiviteettina). Eclipsen hierarkkisesta näkymästä valitaan projektin pakkauskansio hiiren oikealla näppäimellä, ja avautuvasta valikosta valitaan New ja sen valikosta Class. Uuden luokan luonti-ikkuna avautuu. Nimeksi annetaan DataHandler ja muut asetukset jätetään oletuksiksi (kuva 16).



Kuva 16. Tietokantaluokan luonti.

Aliluokka DBHelper luodaan DataHandler-luokan sisään (listaus 3). DBHelper perii SQLiteOpenHelper-luokan ominaisuudet. DBHelper vastaa tietokannan taulukon luomisesta ja sen päivittämisestä (33).

```
private static class DBHelper extends SQLiteOpenHelper
{


}
```

Listaus 3. Aliluokka DBHelper.

9.3 Tietokannan määrittely ohjelmakoodissa

Aikaisemmin määriteltiin, että tallennettavaa dataa ovat langan nimi, tuottaja, väri ja materiaali.

Taulukkoa varten luodaan string-muuttujat name, manufacturer, color ja material, ja niille asetetaan arvot. Arvot otetaan Add Yarn Activity -aktiviteetin asettelutiedoston EditText-elementeistä. (Listaus 4.)

```
public class DataHandler
{
        public static final String NAME = "name";
        public static final String MANUFACTURER = "manufacturer";
        public static final String COLOR = "color";
        public static final String MATERIAL = "material";
}
```

Listaus 4. String-muuttujat taulukolle.

Taulukolle ja tietokannalle tarvitaan nimet. Tietokannan nimi tulee Android-laitteeseen tallennettavan tietokantatiedoston nimeksi (listaus 5). Tietokannalle tarvitaan myös int-tyyppinen versionumero (listaus 5).

```
        public static final String TABLE_NAME = "yarntable"; //table name
        public static final String DATA_BASE_NAME = "yarndatabase.db"; //Data base name
        public static final int DATABASE_VERSION = 1;
```

Listaus 5. Muuttujat taulukon nimelle, tietokannan nimelle ja tietokannan versiolle.

Lopuksi tehdään muuttuja, joka määrittelee taulukon (listaus 6).

```
public static final String TABLE_CREATE = "create table yarntable " +
                                          "(name text not null, " +
                                          "manufacturer text not null, " +
                                          "color text not null, " +
                                          "material text not null);";
```

Listaus 6. Taulukon määrittävä muuttuja.

9.4 Tietokantafunktiot

9.4.1 Tietokannan luominen ja päivittäminen

DBHelper-luokkaan tulevat SQLite-metodit onCreate ja onUpgrade.

OnCreate-metodi kutsutaan, kun tietokanta luodaan ensimmäisen kerran (33). Taulukon luominen tapahtuu tässä metodissa execSQL-metodin avulla (listaus 7). Tässä luodaan siis taulukko nimeltä yarntable.

```
public void onCreate(SQLiteDatabase db)
                    {
                                db.execSQL(TABLE_CREATE);
                    }
```

Listaus 7. Taulukon luominen onCreate-metodissa.

OnUpgrade-metodi kutsutaan, kun tietokanta pitää päivittää jollakin tavalla (33). ExecSQL-metodia käytetään vanhojen tietojen poistamiseen. DROP TABLE IF EXISTS on SQLite-komento (listaus 8), joka poistaa vanhat tiedot taulukosta ennen uusien tietojen asettamista. Vanhat tiedot ovat tällöin jo tallessa tietokannassa. Sitten voidaan luoda uusi tyhjä taulukko kutsumalla metodia onCreate.

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

Listaus 8. Taulukon poistaminen SQLite-komennolla.

9.4.2 Tietokannan avaaminen ja sulkeminen

Tietokannan avaamiseen ja sulkemiseen tarkoitetut metodit tulevat DataHandler-luokkaan. Metodi, joka vastaa tietokannan avaamisesta on nimeltään Open. Open-metodissa DBHelper-luokan objekti dbhelper kutsuu metodia getWritableDatabase (listaus 9). GetWritableDatabase avaa tietokannan, jota tullaan käyttämään tiedon lukemiseen ja kirjoittamiseen (33).

```
public DataHandler open()
		{
			db = dbhelper.getWritableDatabase(); //open the database
			return this;
		}
```

Listaus 9. Open-metodi.

Sulkemisesta vastaava metodi on nimeltään close. Close-metodissa kutsutaan SQLite-funktiota nimeltä close (listaus 10), joka sulkee minkä tahansa avoinna olevan tietokantaobjektin (33).

```
public void close() //26.11.
		{
			dbhelper.close(); //close the database
		}
```

Listaus 10. Close-metodi.

9.4.3 Tiedon lisääminen ja palautus

Tiedon lisäämiseen ja palauttamiseen tarkoitetut metodit tulevat myös DataHandler-luokkaan. InsertData-metodissa tiedot lisätään tietokantaan. Kyseisen metodin sisällä käytetään Androidin ContentValues-luokkaa, joka varastoi kokoelman arvoja (34). Luokkametodi nimeltä Put lisää arvot kokoelmaan (listaus 11). Kun kaikki arvot on lisätty, kutsutaan SQLite-funktio nimeltä insertOrThrow (listaus 11). Tämä funktio lisää uuden rivin tietokantaan.

```
public long insertData(String name, String manufacturer, String color, String material) //insert data to db
		{
			ContentValues content = new ContentValues();
			content.put(NAME, name);
			content.put(MANUFACTURER, manufacturer);
			content.put(COLOR, color);
			content.put(MATERIAL, material);
			return db.insertOrThrow(TABLE_NAME, null, content);
		}
```

Listaus 11. insertData-metodi.

ReturnData on Cursor-tyyppinen metodi. Cursor on rajapinta, joka tarjoaa luku- ja kirjoitusoikeuden siihen kokoelmaan, jonka tietokantakysely on palauttanut (35).

Niinpä metodin sisällä kutsutaan SQLite-metodia nimeltä query, joka tekee kyselyn taulukkoon (listaus 12). Ensimmäinen parametri on taulukon nimi, johon kysely tehdään. Toisena parametrina on lista niistä sarakkeista, jotka halutaan palauttaa, eli tässä tapauksessa kaikki neljä.

```
public Cursor returnData()
		{
			return db.query(TABLE_NAME,
			new String[] {NAME, MANUFACTURER, COLOR, MATERIAL},
			null, null, null, null, null);
		}
```

Listaus 12. ReturnData-metodi.

9.4.4 Tallennusfunktio

Lopuksi tehdään vielä tallennusmetodi nimeltä saveYarn AddYarnActivity-luokkaan. Tämä julkinen metodi käynnistyy, kun käyttäjä painaa Save yarn -painiketta sen jälkeen, kun on täyttänyt tekstikentät. Tekstikenttien tekstit palautetaan xml-tiedostosta getText-metodin avulla ja muutetaan string-tyyppisiksi (listaus 13) (36).

```
public void saveYarn(View view)
		{
			String getName = name.getText().toString();
		}
```

Listaus 13. Ensimmäisen tekstikentän teksti palautetaan.

Seuraavaksi luodaan DataHandler-luokan objekti handler. Handler-objektia ja DataHandler-luokan metodeja käytetään tietokannan avaamiseen ja tiedon lisäämiseen tietokantaan (listaus 14). Lopuksi tietokanta suljetaan ja AddYarnActivity lopetetaan.

```
public void saveYarn(View view)
        {
                …
                handler = new DataHandler(getBaseContext());
                handler.open();
                handler.insertData(getName, getManufacturer, getColor, getMaterial);
                handler.close();
                finish(); //finish the AddYarnActivity
        }
```

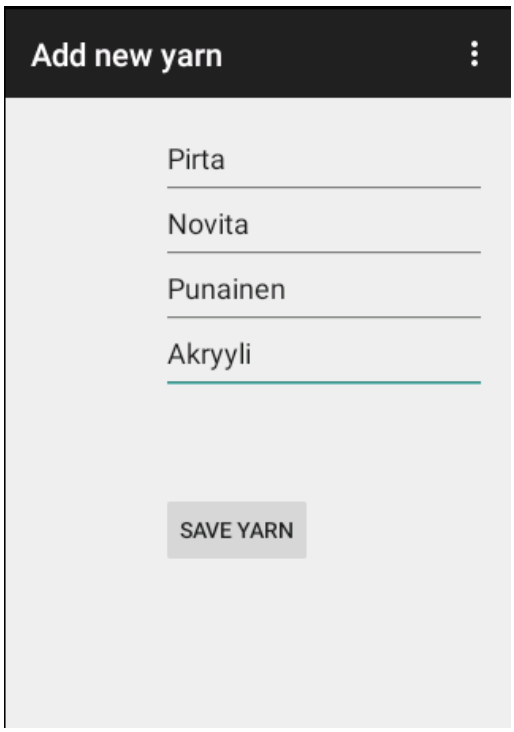
Listaus 14. Tietokannan käsittely.

Käyttäjän kirjoittamat tiedot on nyt lisätty tietokantaan. Jotta asia voidaan varmistaa, tiedot on tarkistettava.

9.5 Tallennetun tiedon tarkastelu

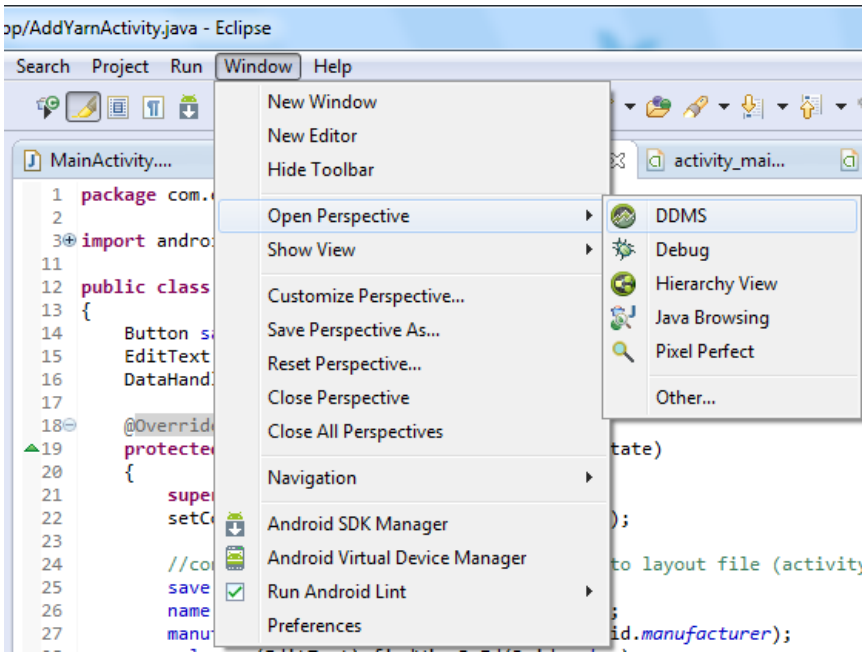
Helpoin tapa tarkastella tallennettuja SQLite-taulukoita on tarkastelu Android-emulaattorin kautta. Emulaattori on eräänlainen virtuaalinen laite, joka toimii mahdollisimman samalla tavalla kuin alkuperäinen laite. Emulaattori on hyvä vaihtoehto Android-kehityksessä, mikäli alkuperäistä laitetta (esim. puhelinta) ei ole saatavilla. Emulaattori täytyy asentaa Eclipsessä erikseen, mutta välineet asennukseen tulevat SDK-paketin mukana.

Eclipsessä sovellus käynnistetään emulaattorissa, ja sovellusta käytetään sen verran, että joitakin tietoja saadaan lisättyä tietokantaan (kuva 17).



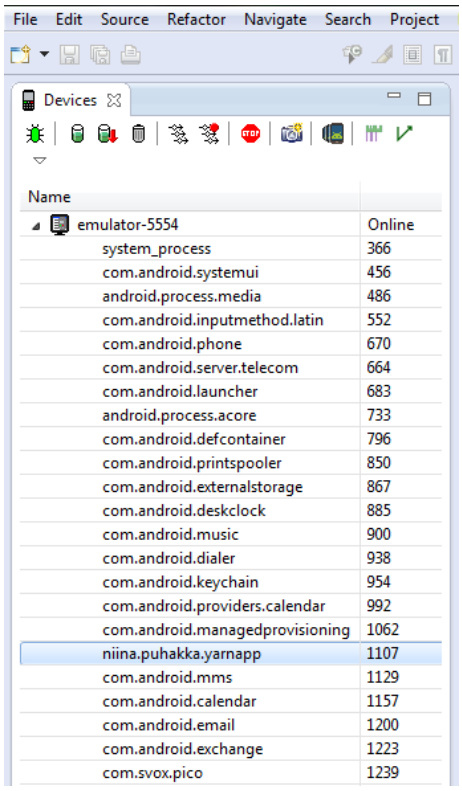
Kuva 17. Tietojen lisääminen sovelluksessa.

Tämän jälkeen jätetään sovellus käyntiin ja siirrytään takaisin Eclipseen. Ylävalikosta valitaan Window ja sieltä Open Perspective. Auenneesta valikosta valitaan DDMS (kuva 18).



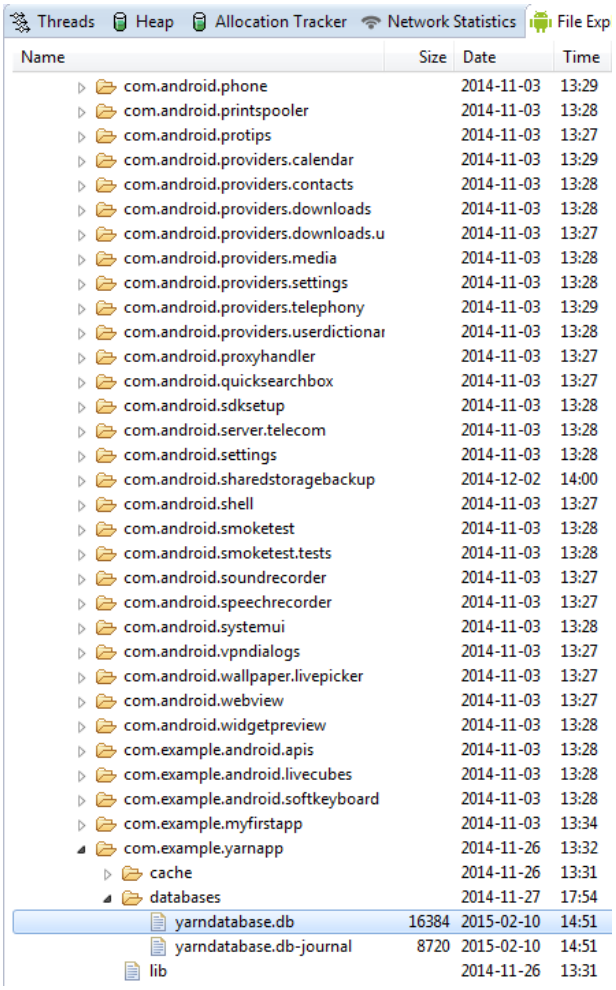
Kuva 18. DDMS-perspektiivin valinta.

DDMS-perspektiivissä vasemmalla näkyvät käytössä olevat laitteet. Laitteista valitaan emulaattori, ja alla olevista pakettinimistä valitaan sovelluksen pakettinimi niina.puhakka.yarnapp (kuva 19).



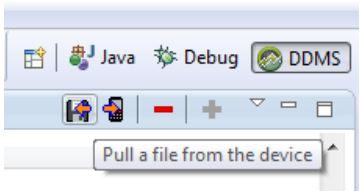
Kuva 19. Pakettinimen valinta.

Kun pakettinimi on valittu, oikealta valitaan File Explorer, joka näyttää emulaattorin tiedostot hierarkisessa järjestyksessä. Tiedostoista etsitään kansio, joka on nimetty pakettinimen mukaan. Tässä kansiossa on alikansio databases, jonka sisältä löytyy tietokantatiedosto yarndatabase.db (kuva 20).



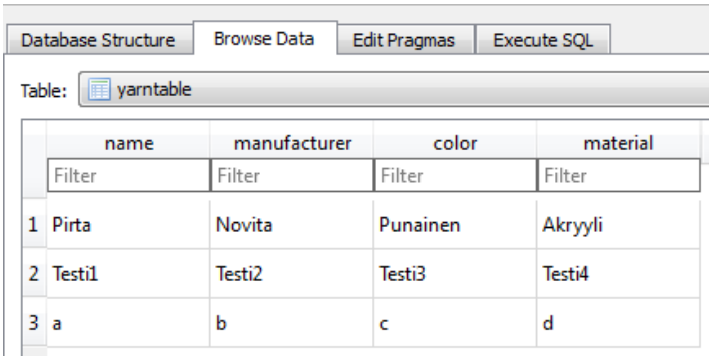
Kuva 20. Tietokantatiedoston sijainti.

Tiedosto on tallennettava johonkin muuhun sijaintiin, jotta se voidaan lukea. DDMS-näkymän oikeasta yläkulmasta valitaan kuvake ”pull a file from the device” ja näin tiedosto voidaan tallentaa johonkin muuhun sijaintiin tietokoneella (kuva 21).



Kuva 21. Tiedoston tallennus emulaattorin ulkopuolelle.

Tietokantatiedoston avaamiseen tarvitaan erillinen ohjelma. Tässä työssä käytetään ohjelmaa nimeltä SQLite Database Browser (kuva 22).



Kuva 22. Tietokannan yarntable-taulukko SQLite Database Browser -ohjelmassa.

Ohjelmalla nähdään, että tiedot ovat tallentuneet oikein. Tästä tilanteesta on hyvä jatkaa prototyypin kehittämistä.

10 TULOSTEN TARKASTELU JA PÄÄTELMÄT

10.1 Suunnittelun tärkeys

Opinnäytetyön tekovaiheessa tuli usein huomattua, kuinka tärkeää projektin suunnittelu on. Onnistuneen projektin takana on yleensä hyvä suunnitelma. Tämän työn aikana huomattiin esimerkiksi se, kuinka tärkeää aikataulun laatiminen on. Kun kunnollista aikataulua ei laadittu suunnitteluvaiheessa, projekti venyi liian pitkäksi ja lopulta tekemisellä tuli kiire. Huono aikataulutus heijastui myös siihen, millainen kehityksessä olleesta prototyypistä lopulta saatiin tehtyä.

10.2 Prototyypin lopputulos

Prototyyppi jäi tämän opinnäytetyön aikana hieman keskeneräiseksi. Tiedon tulostus ja poisto jäivät puuttumaan toiminnoista. Alun perin tarkoituksena oli tehdä sovellusprototyyppi, jota voisi hyödyntää käsityöharrastuksen parissa. Tuloksena on kuitenkin lähinnä pohja mahdolliselle tulevalle sovellukselle.

10.3 Työkalujen valinta ja dokumentointi

Työkalujen valinnan tärkeys korostui etenkin dokumentointivaiheessa. Projektin alussa piti käytännössä valita kahden eri työkalun väliltä: Eclipse vai Android Studio. Android Studio on Android-kehityksen uusi työkalu. Android Studio oli kuitenkin vielä beta-vaiheessa, kun projekti aloitettiin. Beta-vaiheessa saattaa ilmetä outoja

bugeja, joihin ei välttämättä löydy ratkaisua, joten Eclipse oli siinä mielessä parempi vaihtoehto. Androidin ohjeetkin oli kirjoitettu Eclipselle. Kun opinnäytetyö oli edennyt dokumentointivaiheeseen, Android Studiosta tuli virallinen kehitystyökalu, jolloin Eclipseä ei enää kehitetty. Ohjeet päivitettiin Android Studiolle, joten Eclipsestä oli hankala löytää ajantasaisia ohjeita. Niinpä dokumentointi vaikeutui jonkin verran.

Dokumentoinnissa oli muitakin haasteita. Kehityksen aikana ilmeni monesti kysymyksiä, joihin ei löytänyt vastausta Androidin virallisesta dokumentaatiosta. Tällöin oli turvauduttava foorumeihin ja blogeihin, jolloin tiedon paikkansapitävyyttä oli hankalampi arvioida. Tämä hidasti opinnäytetyön dokumentointiprosessia.

10.4 Tekniset ongelmat ja ohjelmointi

Tekniset ongelmat osaltaan hidastivat ohjelmointiprosessia. Eclipsessä on omat buginsa, joiden kanssa on vain opittava toimimaan oikein. Käytössä ollut Android-puhelin ei myöskään aina toiminut odotetulla tavalla. Joinakin päivinä puhelin ei vain yhdistänyt tietokoneeseen, vaikka kaikki oli asennettu oikein. Tällöin oli käytettävä emulaattoria, joka käynnisti sovelluksen huomattavan hitaasti verrattuna puhelimeen. Ohjelmoinnin alkuvaiheessa oli myös paljon asioita, joita ei vain tajunnut ottaa huomioon, koska tietämys Android-kehityksestä oli vielä niin vähäinen. Niinpä jotkut asiat oli vain tehtävä uudestaan, mikä hidasti jonkin verran projektin etenemistä. Ohjelmointi vaati paljon Androidin API-tasojen tutkimista, mikä tuntui varsinkin projektin alkuvaiheessa työläältä.

10.5 Yhteenveto

Opinnäytetyö oli kokonaisuudessaan erittäin opettavainen prosessi. En tiennyt Android-ohjelmoinnista juuri mitään ennestään, joten siltä alalta olen oppinut paljon. Opin myös erityisesti sen, että suunnitteluun kannattaa varata riittävästi aikaa ennen kuin asioita alkaa toteuttaa. Tällöin vältytään parhaimmassa tapauksessa turhalta työltä, kun samoja asioita ei tarvitse tehdä useampaan kertaan. Opin myös priorisoimaan asioita, sillä opinnäytetyö itsessään vaati luopumaan joistakin asioista, jotta työtä sai työstettyä eteenpäin. Myös opinnäytetyön tekoprosessissa oli mietittävä, mikä on tärkeää, koska tiesi, ettei kaikkea ehdi kuitenkaan toteuttamaan. Opinnäytetyö

on myös hyödyllinen tulevaisuuden kannalta, sillä opin sen aikana projektinhallintaa ja itsenäistä työskentelyä, ja koin kehittyväni ohjelmoijana.

LÄHTEET

1. Elgin, B. 2005. Google Buys Android for Its Mobile Arsenal. Saatavissa: http://www.webcitation.org/5wk7sIvVb/ [viitattu 5.2.2015].

2. Androidsuomi.fi. Mikä on Android?. Saatavissa: http://blog.androidsuomi.fi/mika-on-android/ [viitattu 13.1.2015].

3. Mahapatra, L. 2013. Android Vs. iOS: What's The Most Popular Mobile Operating System In Your Country?. Saatavissa: http://www.ibtimes.com/android-vs-ios-whats-most-popular-mobile-operating-system-your-country-1464892 [viitattu 13.1.2015].

4. Kahn, J. 2014. Google shows off new version of Android, announces 1 billion active monthly users. Saatavissa: http://www.techspot.com/news/57228-google-shows-off-new-version-of-android-announces-1-billion-active-monthly-users.html [viitattu 13.1.2015].

5. Morrill, D. 2008. Announcing the Android 1.0 SDK, release 1. Saatavissa: http://android-developers.blogspot.fi/2008/09/announcing-android-10-sdk-release-1.html [viitattu 14.1.2015].

6. Android. 2015. Codenames, Tags, and Build Numbers. Saatavissa: https://source.android.com/source/build-numbers.html [viitattu 15.1.2015].

7. Chacos, B. 2014. From Android L to Google Fit: All the new products and features from Google I/O. Saatavissa: http://www.pcworld.com/article/2367746/from-android-l-to-google-fit-all-the-announcements-from-google-i/from-android-l-to-google-fit-all-t.html [viitattu 15.1.2015].

8. Meier, R. 2012. Professional Android 4 Application Development. Indiana: John Wiley & Sons, Inc.

9. The Eclipse Foundation. 2015. About the Eclipse Foundation. Saatavissa: http://www.eclipse.org/org/#about [viitattu 10.2.2015].

10. Holzner, S. 2004. Eclipse. Sebastopol: O'Reilly Media, Inc.

11. Android Developers. 2014. Get the Android SDK. Saatavissa: https://developer.android.com/sdk/index.html [viitattu 28.10.2014].

12. Eclipse. 2014. Eclipse/Installation. Saatavissa: https://wiki.eclipse.org/Eclipse/Installation [viitattu 28.10.2014].

13. Oracle Corporation. 2014. Java SE Development Kit 8 Downloads. Saatavissa: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html [viitattu 28.10.2014].

14. Thornsby, J. 2013. Connecting Your Android Device to Eclipse. Saatavissa: http://www.developer.com/ws/android/connecting-your-android-device-to-eclipse.html [viitattu 11.11.2014].

15. Team Android. 2014. Download Android USB Drivers for Windows and Mac. Saatavissa: http://www.teamandroid.com/download-android-usb-drivers/ [viitattu 11.11.2014].

16. Android Developers. 2014. Creating an Android Project. Saatavissa: https://developer.android.com/training/basics/firstapp/creating-project.html [viitattu 18.8.2014].

17. Android Developers. 2015. App Manifest. Saatavissa: https://developer.android.com/guide/topics/manifest/manifest-intro.html [viitattu 20.1.2015].

18. Android Developers. 2015. Activity. Saatavissa: http://developer.android.com/reference/android/app/Activity.html [viitattu 21.1.2015].

19. Android Developers. 2015. UI Overview. Saatavissa: http://developer.android.com/guide/topics/ui/overview.html [viitattu 21.1.2015].

20. Android Developers. 2014. Relative Layout. Saatavissa: http://developer.android.com/guide/topics/ui/layout/relative.html [viitattu 29.12.2014].

21. Android Developers. 2015. Button. Saatavissa: http://developer.android.com/reference/android/widget/Button.html [viitattu 21.1.2015].

22. Android Developers. 2015. Text Fields. Saatavissa: http://developer.android.com/guide/topics/ui/controls/text.html [viitattu 22.1.2015].

23. Android Developers. 2015. Intent. Saatavissa: http://developer.android.com/reference/android/content/Intent.html [viitattu 22.1.2015].

24. Android Developers. 2014. Saving Data. Saatavissa: http://developer.android.com/training/basics/data-storage/index.html [viitattu 11.12.2014].

25. Android Developers. 2014. Storage Options. Saatavissa: http://developer.android.com/guide/topics/data/data-storage.html [viitattu 11.12.2014].

26. Android Developers. 2014. Saving Files. Saatavissa: http://developer.android.com/training/basics/data-storage/files.html [viitattu 15.12.2014].

27. Android Developers. 2014. Saving Key-Value Sets. Saatavissa: http://developer.android.com/training/basics/data-storage/shared-preferences.html [viitattu 15.12.2014].

28. Android Developers. 2014. Saving Data in SQL Databases. Saatavissa: http://developer.android.com/training/basics/data-storage/databases.html [viitattu 15.12.2014].

29. SQLite. 2014. About SQLite. Saatavissa: http://www.sqlite.org/about.html [viitattu 16.12.2014].

30. SQLite. 2014. SQLite Is Self-Contained. Saatavissa: http://www.sqlite.org/selfcontained.html [viitattu 16.12.2014].

31. Sqlite. 2014. SQLite Is Serverless. Saatavissa: http://www.sqlite.org/serverless.html [viitattu 16.12.2014].

32. SQLite. 2014. SQLite Is A Zero-Configuration Database. Saatavissa: http://www.sqlite.org/zeroconf.html [viitattu 16.12.2014].

33. Android Developers. 2014. SQLiteOpenHelper. Saatavissa: http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html [viitattu 29.12.2014].

34. Android Developers. 2015. ContentValues. Saatavissa: http://developer.android.com/reference/android/content/ContentValues.html [viitattu 9.2.2015].

35. Android Developers. 2015. Cursor. Saatavissa: http://developer.android.com/reference/android/database/Cursor.html [viitattu 9.2.2015].

36. Android Developers. 2015. EditText. Saatavissa: http://developer.android.com/reference/android/widget/EditText.html [viitattu 11.2.2015].