Pipsa Korkiakoski

MINIGOLF-TULOSSOVELLUKSEN OHJELMOINTI ANDROIDILLE

MINIGOLF-TULOSSOVELLUKSEN OHJELMOINTI ANDROIDILLE

Pipsa Korkiakoski

Opinnäytetyö

Syksy 2013

Tietojenkäsittelyn koulutusohjelma

Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä: Pipsa Korkiakoski
Opinnäytetyön nimi: Minigolf-tulossovelluksen ohjelmointi Androidille
Työn ohjaaja: Jouni Juntunen
Työn valmistumislukukausi ja –vuosi: Syksy 2013 Sivunmäärä: 34 + 1

Opinnäytetyön aikana tavoitteena oli kehittää prototyyppinen mobiilisovellus Minigolf-pallopelin pelaajille tulosten tallentamista varten. Kohdealustaksi valittiin Android-käyttöjärjestelmä. Sovelluksen avulla pelaaja voi tallentaa pelin aikana saamansa pisteet ja tarkastella niitä myöhemmin. Sovellus määriteltiin olevan yhteensopiva Android-version 3.0 tai sitä uudemman version kanssa. Testilaitteena projektin aikana käytettiin Samsung Galaxy S III-matkapuhelinta, jossa käyttöjärjestelmän versio oli 4.1.2.

Opinnäytetyön yhtenä osana oli tarkastella mobiiliohjelmointia Android-laitteille sekä erityisesti tietokantaohjelmointia Android-mobiililaitteille ja SQLite-relaatiotietokantajärjestelmän käyttöä Android-sovelluksissa. Opinnäytetyöllä ei ole toimeksiantajaa, vaan halusin itse tutustua aiheeseen tarkemmin. Mobiilisovellusohjelmointi Androidille oli jo aiemmin hieman tuttu itselleni ja halusin tehdä opinnäytetyönä jotain aiheeseen liittyvää, koska mielestäni ohjelmointi mobiililaitteille on alalla hyödyllinen taito osata ja Android-laitteille se tapahtuu kaikenlisäksi hyvin yleisellä Java-kielellä. Päätarkastelun kohteeksi valitsin siis tietokannan käsittelyn Android-mobiililaitteella ja jätin vähemmälle huomiolle muun muassa mobiilisovelluksen ulkoasun suunnittelun ja sen toteutuksen. Sovelluskehitys tapahtuu Eclipse-ohjelmointiympäristössä yhdessä Googlen tarjoamien Android ohjelmointityökalujen avulla.

Opinnäytetyön aikana valmiiksi saatu versio tulossovelluksesta ei sellaisenaan ole vielä julkaisukelpoinen, vaan sen ulkoasuun täytyisi vielä kiinnittää huomiota. Lisäksi sovelluksen toiminnallisuuteen olisi hyvä lisätä muun muassa syöttötietojen tarkistus ja tulostensyötön voisi toteuttaa hieman toisella tapaa, jotta pelaajan syöttämät tulokset tallentuisivat ratakohtaisesti tietokantaan. Tällä hetkellä sovellus tallentaa kaikkien ratojen tulokset vasta lopuksi ja tämä voi olla huono esimerkiksi silloin, jos pisteiden syöttö on pelaajalla kesken ja sovellus esimerkiksi sammutetaan jostain syystä kesken kaiken väärin. Pienten lisäysten ja korjausten jälkeen sovellus olisi vakaampi ja luotettavampi käyttää.

Avainsanat: Mobiili, sovellus, kehitys, mobiiliohjelmointi, Android, SQLite, tietokanta, Eclipse, Android SDK

ABSTRACT

Oulu University of Applied Sciences
Degree programme of Business Information Systems

Author: Pipsa Korkiakoski
Title of thesis: Minigolf-result application programming for Android
Supervisor: Jouni Juntunen
Term and year: Autumn 2013 Number of pages: 34 + 1

During the thesis the aim was to develop a prototype of a mobile application for players of mini golf ballgame. The application helps players to save their results during the game and players can view the results later if needed. Target platform was chosen to be Android operating system. The application was determined to be compatible with Android version 3.0 or later. During the project, the test device was Samsung Galaxy S III mobile phone with the operating system version 4.1.2.

One part of the thesis was to study the programming for Android mobile device, in particular database programming for Android mobile devices and using of SQLite relational database system in Android applications. The thesis did not have any contractor, but I just wanted to explore the topic more myself. Programming a mobile application for Android was already a little familiar to me and I wanted to use that subject in my thesis, because I think the programming for mobile devices is a useful skill to have in the field and Android applications are written in very common Java-language.

The main point in thesis was to handle the database on Android devices. For application's appearance's design and its implementation I gave less attention. Application development was done by the Eclipse-programming environment along with a Google's provided Android programming tools.

The version of the application which was finished during the thesis is not ready for publication because the appearance still needs to pay attention to. In addition to improve application's functionality it would be useful to add input validation and the result input could be carried out in slightly different way so that the results could save safely to the database. At present, the application stores all the results only in the end of the whole game. This may be bad if for example the application is shut down incorrectly while player is still inputting points. After the small additions and corrections the applications will be more stable and reliable to use.

Keywords: Mobile, application, development, mobile programming, Android, SQLite, database, Eclipse, Android SDK

SISÄLLYS

1 JOHDANTO

Minigolf on kansanomaisempi nimitys mailapelistä nimeltä ratagolf ja siinä on golfin tapaan 18-reikäinen rata. (Wikipedia 2013, hakupäivä 11.10.2013). Peli kerää kaikenlaisia ja kaiken ikäisiä pelaajia radoille, mutta osa harrastaa lajia tosissaan kilpailumielessä ja tätä varten on Suomeenkin perustettu Suomen Ratagolfliitto ry. Lajin historia suomessa alkaa vuodesta 1952 lähtien, kun teknillinen ammattikorkeakoulu muutti pian sodan jälkeen Espoon Otaniemeen. Teknillisen ammattikorkeakoulun opiskelijoiden eli teekkareiden kristillinen yhdistyksen Ristin Kiltan oli tarkoituksenaan perustaa kylään oma kappeli, mutta varoja rakentamiseen ei ollut. Eräs opiskelija ristinkiltalaisista, Eric Schalin, oli Englannissa tutustunut ratagolfin ja sai idean perustaa minigolfradan teekkarikylään. Radan toiminnalla voitiin kerätä varoja kappelin rakentamista varten. Ratagolfista tuli heti niin suosittu, että kiltalaiset perustivat pian jo samana vuonna toisen radan Sibeliuksen puistoon Helsinkiin. (Sibeliuspuiston minigolf 2013, hakupäivä 11.10.2013.)

Pelin tavoitteena on golfin tapaan saada pallo reikään mahdollisimman pienellä lyöntimäärällä. Mikäli pelaaja ei saa palloa kuudennella lyönnillä reikään, keskeytyy kyseisen radan pelaaminen ja radan tulokseksi merkitään seitsemän lyöntiä. (Hutunki 2013, hakupäivä 11.10.2013). Jos radan punainen jatkorajaviiva ylitetään, täytyy pelaamista jatkaa lyömällä palloa sen pysähtymiskohdassa. Pallo on mahdollista siirtää helpottaakseen lyömistä reunoista korkeintaan 20 cm ja esteestä 50 cm verran. Suomen yleisin ratamateriaali on tehty huovasta. Muita alustoja ovat eterniitti, betoni ja MOS (minigolf open stantard). Pääsääntöisesti yhden pelin aikana käytetään yhtä mailaa ja se on golfista tutun putterin tapainen. Palloja on tuhansia erilaisia ja ominaisuudet vaihtelevat sen kovuuden, pinnan ja pompun mukaan. (Valjärvi, J. & Metsäranta, T. 2007, hakupäivä 11.10.2013.)

Esimerkiksi Oulussa Heinäpään minigolf-radalla tulokset merkitään paperille. Opinnäytetyön tavoitteena on kehittää helppokäyttöinen tulossovellus, johon pelaaja voi syöttää omat pisteensä ja tallentaa ne myöhempiä tarkasteluja varten. Sovelluksen ulkoasun suunnitteluun ja toteutukseen ei opinnäytetyöprosessin aikana keskitytä juurikaan. Tavoitteena on kehittää sovelluksesta aluksi sellainen versio, että se vastaa toimintojensa puolesta käyttäjien tarpeisiin. Ensimmäisessä versiossa sovelluksella voi tallentaa samanaikaisesti vain yhden pelaajan pelitiedot, mutta jatkossa sovellusta kehittäessä, olisi hyvä ottaa huomioon myös pelaajan seurassa pelaavat henkilöt ja

heidän mahdollisuutensa hyödyntää yhdessä laitteessa asennettua sovellusta. Sovellus kehitetään toimimaan mobiililaitteissa, joissa on Android-käyttöjärjestelmä ja järjestelmältä vaaditaan, että se on vähintään Android-versio 3.0 tai uudempi, mutta kuitenkin korkeintaan 4.1., sillä se on määritelty sovelluksen kohdealustaksi.

Ohjelmointi Android-laitteille on hyödyllinen taito osata, koska siinä sovelletaan Java-kieltä ja Gartnerin helmikuussa 2013 tekemän tutkimuksen mukaan käyttöjärjestelmien suosiossa loppukäyttäjien kesken Android oli listan sijalla yksi (Gartner Inc. 2013, hakupäivä 17.10.2013). Sovelluskehityksessä Androidille vahvuutena on myös lähdekoodin avoimuus, työkalujen maksuttomuus ja jakelukanavien runsaus. Java-kieli on hyvä osata, koska sitä käytetään paljon. TIOBE ohjelmointiyhteisön määrittelemä sijoitus Java-kielelle on tällä hetkellä sija kaksi. Sija määritellään sen mukaan muun muassa, kuinka paljon maailmassa on kieltä osaavia tekijöitä ja kuinka paljon hakuja kieleen liittyen on tehty suosituimmilla hakukoneilla. TIOBE päivittää listaa kuukauden välein. Sijoitus ei ole lainkaan huono, koska eroa sijaan yksi, jolla on tällä hetkellä C-kieli, on noin 1-prosenttiykkö ja toiseen suuntaan sijaan kolme eroa on jopa noin 7-prosenttiyksikköä. (TIOBE Software 2013, hakupäivä 1.11.2013.)

Aloitteleville mobiiliohjelmoijille on nykyään tarjolla ilmainen ladattava paketti internetissä, joka sisältää kaiken tarvittavan Android-mobiiliohjelmoinnissa. ADT (Android Developer Tools) Bundle-paketti sisältää Eclipse Foundationin tarjoaman Eclipse-ohjelmointiympäristön, Googlen tarjoaman AndroidSDK:n sekä ajoympäristön ja erilaiset emulaattorit. (Android Developer, hakupäivä 8.11.2013).

Opinnäytetyön eräänä mielenkiintoisimpana tarkastelukohteena on Android-laitteissa käytettävä SQLite-relaatiotietokantajärjestelmä, jota käytetään tietojen tallennukseen. SQLite on kevyt tietokantamoottori, joka vie pienen tilan laitteen kiintolevyltä ja sitä käytetään monissa mobiilikäyttöjärjestelmissä (Code Project 2013, hakupäivä 17.10.2013).

2 ANDROID

Android on internetpalveluiden tarjoajan Googlen julkaisema käyttöjärjestelmä älypuhelimille ja mobiililaitteille. Se on avoimen lähdekoodin alusta ja sen käyttäminen on ilmaista. Android-puhelimia kehittävät valmistajat kuten Samsung, HTC ja Sony. (Android Suomi 2013, hakupäivä 11.10.2013.)

Android Inc. perustettiin Palo Altossa Kaliforniassa lokakuussa 2003. Yhtiön perustajina olivat Andy Rubin, Rich Miner, Nick Sears sekä Chris White. Alun perin yrityksen oli tarkoitus kehittää käyttöjärjestelmiä digikameroille, mutta silloisten alan heikkojen markkinoiden vuoksi yritys siirtyi kehittämään käyttöjärjestelmää älypuhelimille. Jo kehityksen alussa rahat loppuivat ja kerrotaan, että Andy Rubyn läheinen ystävä, Applen entinen insinööri, toi hänelle 10 000 dollaria rahaa kirjekuoressa, mutta kieltäytyi yrityksen osakkuudesta. (Deleon, W. 2013, hakupäivä 7.11.2013.)

Järjestelmä koostuu karkeasti kahdesta osiosta, jossa käyttöjärjestelmän pohjana on Googlen muokkaama Linux mobiilikäyttöön ja Android-sovellukset toimivat Java-kieleen perustuvan Dalvik-virtuaalikoneen päällä. Sovelluskehitys tehdään Java-kielellä, mutta ei tarkalleen virallisella Java-kielellä vaan Googlen käyttää Apache Harmony-luokkakirjastoja ja niistä luotu Java-tavukoodi käännetään erikseen Dalvik-virtuaalikoneen käyttämään muotoon. (Android Suomi 2013, hakupäivä 11.10.2013.)

Androidin lähdekoodi on pituudeltaan noin 12 miljoonaa koodiriviä, joista 3 miljoonaa riviä on XML-koodia, 2,8 miljoonaa riviä on C-koodia ja 2,1 miljoonaa riviä on Java-koodia (Wikipedia 2013, hakupäivä 11.10.2013).

2.1 Ohjelmistokehitys

Androidista on julkaistu paljon uusia versioita ja päivityksiä aina julkistuksen jälkeen. Uusi versio tai päivitys korjaa yleensä aiempia virheitä ja tuo lisää uusia ominaisuuksia. Esimerkiksi versiosta 1.0 puuttunut virtuaalinen näppäimistö saatiin seuraavaan versioon 1.5 lisättyä. (Wikipedia 2013, hakupäivä 11.10.2013.) Useiden eri versioiden oleminen yhtä aikaa markkinoilla tuo myös haasteita ja haittoja sovelluskehittäjille, koska eri versioiden julkaisu sirpaloittaa alustaa ja tiettyä so-

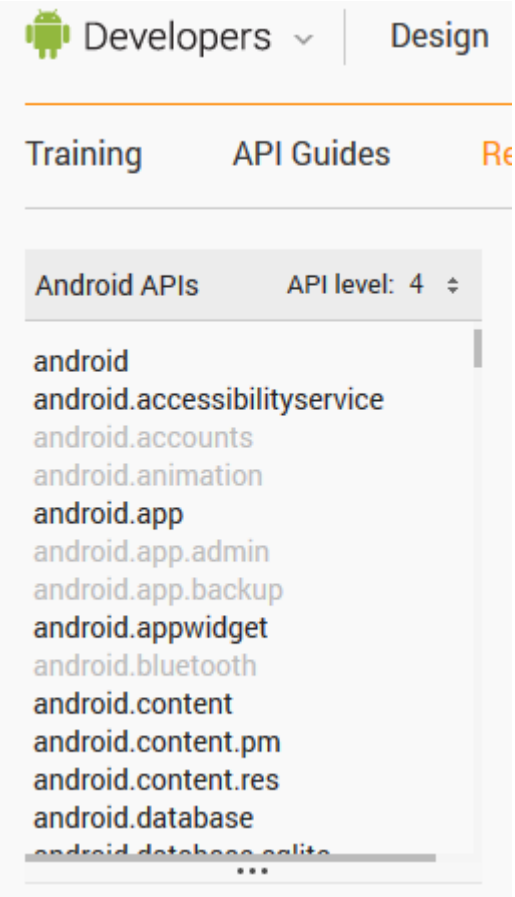
vellusta ei saa välttämättä toimimaan kaikilla Android-puhelimilla vaan pahimmissa tapauksissa kehittäjät joutuvat ylläpitämään useampaa koodipohjaa eri käyttöjärjestelmän versioille (Hynninen, T. 2009, hakupäivä 7.11.2013).

Ensimmäinen versio Androidista, Android 1.0 Astro julkaistiin virallisesti yhdessä HTC Dreamin T-Mobile G1 puhelimen käyttöjärjestelmänä syksyllä 2008. Suurin heikkous julkaisussa oli virtuaalinäppäimistön puuttuminen, jolloin laitteelta vaadittiin aina fyysinen näppäimistö. Tämä puutos korjaantui versiossa 1.5 Cupcake vuonna 2009 keväällä. Versiosta 1.5 lähtien Google on nimennyt käyttöjärjestelmäversiot jälkiruokiin liittyvillä nimillä. Android 3.0 Honeycomb-versio on blogin mukaan Androidin historian kummajainen, joka toi käyttöliittymään futuristisia elementtejä, joita sitten jalostettiin seuraavissa julkaisuissa. Versiota pidetään välijulkaisuna, joka loi pohjan seuraaville versioille. Versio esiteltiin alkukeväästä vuonna 2011 käytettäväksi ainoastaan tablet-laitteissa. Tärkeimpiä ominaisuuksia versiossa oli mm. tuli moniydinsuorittimelle. (Hynninen, T. 2013, hakupäivä 11.10.2013.)

Honycomb -versiosta jalostui Android 4.0 Ice Cream Sandwich, joka oli yhteensopiva niin tablet-laitteille kun älypuhelimillekin. Androidin suurin heikkous oli yhä suorituskyky, johon paneuduttiin antaumuksella projektilla nimeltä Project Butter. Kesällä 2012 julkaistiin versio 4.1 Jelly Bean, jossa pääasiassa muutoksena oli vain uusi kehittyneempi ja parempi suorituskyky. Jelly Beanista julkaistiin myöhemmin myös lisäversiot 4.2 ja 4.3. (Hynninen, T. 2013, hakupäivä 11.10.2013.) Uusin tähän asti julkaistuista versioista julkaistiin 31.10.2013 ja se on versio 4.4 koodinimellä Kit-Kat (Lehtiniitty, M. 2013, hakupäivä 7.11.2013).

2.2 API-taso

API-taso (API Level) on järjestysluku, joka yksilöi sovelluskehyksen, jonka Android-käyttöjärjestelmä tarjoaa. API-taso määrittelee muun muassa pakettien ja luokkien ydinjoukon sekä vaikuttaa esimerkiksi XML-elementtien määrään ja käytettävyyteen. Kuviossa 1 näkyy kaikki ominaisuudet, jotka on viimeisimpään API-tasoon päivitetty. API-tasolla 4 on käytettävissä vain tummennetulla näkyvät ominaisuudet (kuvio 1). (Android Developer, hakupäivä 17.10.2013).



KUVIO 1. API-tason vaikutus ominaisuuksiin (Android Developer)

Sovelluskehyspäivitys kasvattaa API-tason kokonaislukua. Päivitykset on suunniteltu niin, että päivitetty API-taso on yhteensopiva aiempien API versioiden kanssa. Päivitetty taso voi sisältää uusia ominaisuuksia tai muutoksia, mutta aiempia ominaisuuksia ei juurikaan koskaan poisteta kasvattaessa API-tasoa. Kuviossa 2 voi hahmottaa, miten API-tasoa kasvatetaan Android-käyttöjärjestelmän version julkaisun yhteydessä. (Android Developer, hakupäivä 17.10.2013).

Platform Version	API Level	VERSION_CODE	Notes
Android 4.3	18	JELLY_BEAN_MR2	Platform Highlights
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1	Platform Highlights
Android 4.1, 4.1.1	16	JELLY_BEAN	Platform Highlights
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1	Platform Highlights
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH	
Android 3.2	13	HONEYCOMB_MR2	
Android 3.1.x	12	HONEYCOMB_MR1	Platform Highlights
Android 3.0.x	11	HONEYCOMB	Platform Highlights
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1	Platform Highlights
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD	
Android 2.2.x	8	FROYO	Platform Highlights
Android 2.1.x	7	ECLAIR_MR1	Platform Highlights
Android 2.0.1	6	ECLAIR_0_1	
Android 2.0	5	ECLAIR	
Android 1.6	4	DONUT	Platform Highlights
Android 1.5	3	CUPCAKE	Platform Highlights
Android 1.1	2	BASE_1_1	
Android 1.0	1	BASE	

KUVIO 2. API-tasot (Android Developer)

Sovelluksen ohjelmakoodiin API-taso määritellään AndroidManifest.xml-tiedostoon (kuvio 3). AndroidManifestissa määritellään tason versiolle minimi vaatimus, jolla sovellus tulee toimimaan sekä kohde versio, jolla sovellus on suunniteltu ajettavan. (Android Developer, hakupäivä 07.11.2013.)

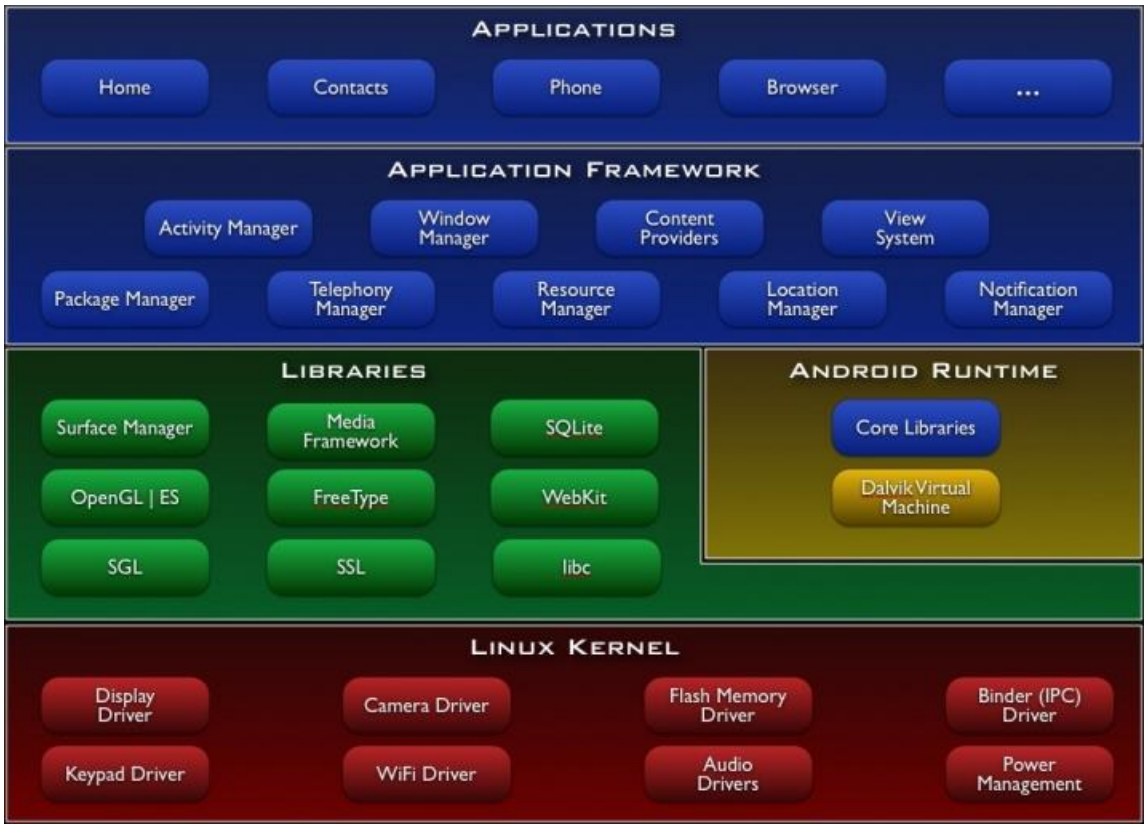
```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="18" />
```

KUVIO 3. API-tason määritys AndroidManifestissa

2.3 Arkkitehtuuri

Android on Linux-ytimen päälle rakennettu sovelluspino. Pino on jaettu viiteen sektoriin ja neljään pääkerrokseen (kuvio 4). Alimmalla tasolla on mobiililaitteille Googlen räätälöimä oma Linux-ydin.

Android käyttää Linuxin versiota 2.6. Ydin sisältää ajurit muun muassa näytölle, näppäimistölle ja äänille sekä verkkopinon, prosessien välisen viestinnän että muistinhallinnan. (Tutorialspoint 2012, hakupäivä 17.11.2013.)



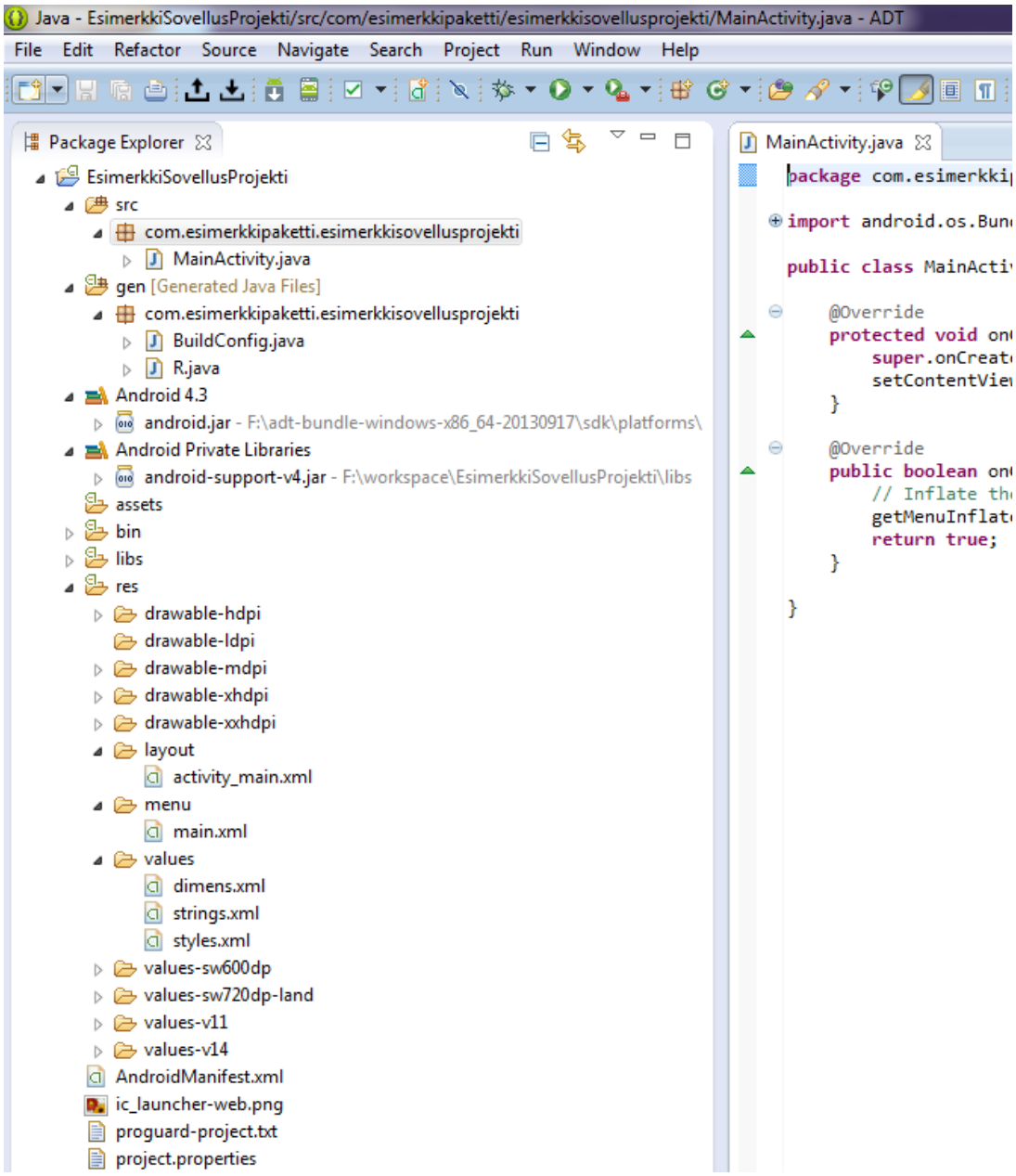
KUVIO 4. Android-käyttöjärjestelmän arkkitehtuuri (eLinuxWiki 2011)

Ydintä ylemmässä kerroksessa ovat natiivit kirjastot sekä Android Runtime. Natiiveissa kirjastoissa ovat muun muassa perus C-kielen ja C++-ohjelmointikielen kirjastot järjestelmälle ja sulautetuille järjestelmille, mediakirjasto videoille ja äänille, kompakti tietokantakirjasto SQLite ja WebKit-selainmoottori. Android Runtime:sta löytyy Javan ydinkirjastot sekä Dalvik-virtuaalikone. Sovelluskehys on sovelluskehittäjien rajapinta. Sovelluskehys sisältää muun muassa aktiviteettien ja ikkunan hallinnan, resurssinhallinnan ja pakkaustenhallinnan. Ylimpänä on sovellustaso, joka näkyy loppukäyttäjälle sovelluksena, jonka Dalvik-virtuaalikone on kääntänyt koodista. Sovellustaso on taso, johon sovellus asennetaan ja jolla sovellus ajetaan. (Tutorialspoint 2012, hakupäivä 17.11.2013.)

3 KEHITYSYMPÄRISTÖ

Yleisin ohjelmointikieli Android-sovellusten tekoon on Java-kieli. Android ei tue Java-kieltä suoraan, vaan luokat esikäännetään Androidin omalla Dalvik-virtuaalikoneella Dalvik executables-tiedostomuotoon (.dex). (Harju, J. 2013 hakupäivä 25.10.2013.) Sovellusten ohjelmointiin tarvitaan Android Software Development Kit (SDK), jonka asennus onnistuu lataamalla paketti Androidin kehityssivustolta developer.android.com ja purkamalla se haluttuun paikkaan tietokoneella. Sovellusten kehittäminen onnistuu pelkällä SDK:lla, mutta usein käytetään myös IDE:tä (Integrated Development Environment), kuten esimerkiksi Eclipse -sovelluskehitintä. Aloittaville mobiiliohjelmoijille on tarjolla nykyään myös ladattava paketti internetissä, joka sisältää kaiken tarvittavan Android-mobiiliohjelmoinnissa. ADT (Android Developer Tools) Bundle-paketti sisältää Eclipse Foundationin tarjoaman Eclipse-ohjelmointiympäristön, Googlen tarjoaman AndroidSDK:n sekä ajoympäristön ja erilaiset emulaattorit. (Android Developer, hakupäivä 17.10.2013.) Lisäksi koneelle täytyy asentaa vielä Javan JDK (Java Development Kit), joka sisältää Java-ohjelmointiin tarvittavat työkalut, kuten Java-kielen kääntäjän (Harju, J. 2013, hakupäivä 25.10.2013).

Projektin luominen Eclipsessä aloitetaan avaamalla käyttäjää opastava velhotoiminto. Android sovellus aloitetaan valitsemalla Android Application Project. Velho kehottaa antamaan aluksi sovellukselle, sovelluksen pakkaukselle sekä itse projektille nimet. Samalla valitaan sovelluksen kohdealusta, jolle sovellus tullaan kehittämään sekä määritellään, mikä on Androidin aikaisin versio, minkä kanssa sovelluksen tulisi olla myös yhteensopiva. Jälkimmäinen määritelmä vaikuttaa käytettävissä olevien kirjastojen määrään. Seuraavassa vaiheessa velhotoimintoa voidaan määritellä, luodaanko sovellukselle käynnistyskuvake ja ensimmäinen aktiviteetti, sekä merkitä sovellus kirjastotyypiksi, muuttaa työtilaa tai liittää sovellus osaksi työjoukkoa. Velhotoiminnon voi viedä loppuun jättämällä loput tiedot oletusarvoihin ja viimeistelemällä projektin luonnin. Eclipsen velho luo sovellukselle automaattisesti hakemistohierarkian (kuvio 5). (Harju, J. 2013, hakupäivä 25.10.2013.)



KUVIO 5. Android-sovelluksen projektin hakemistohierarkia

3.1 Hakemistohierarkia

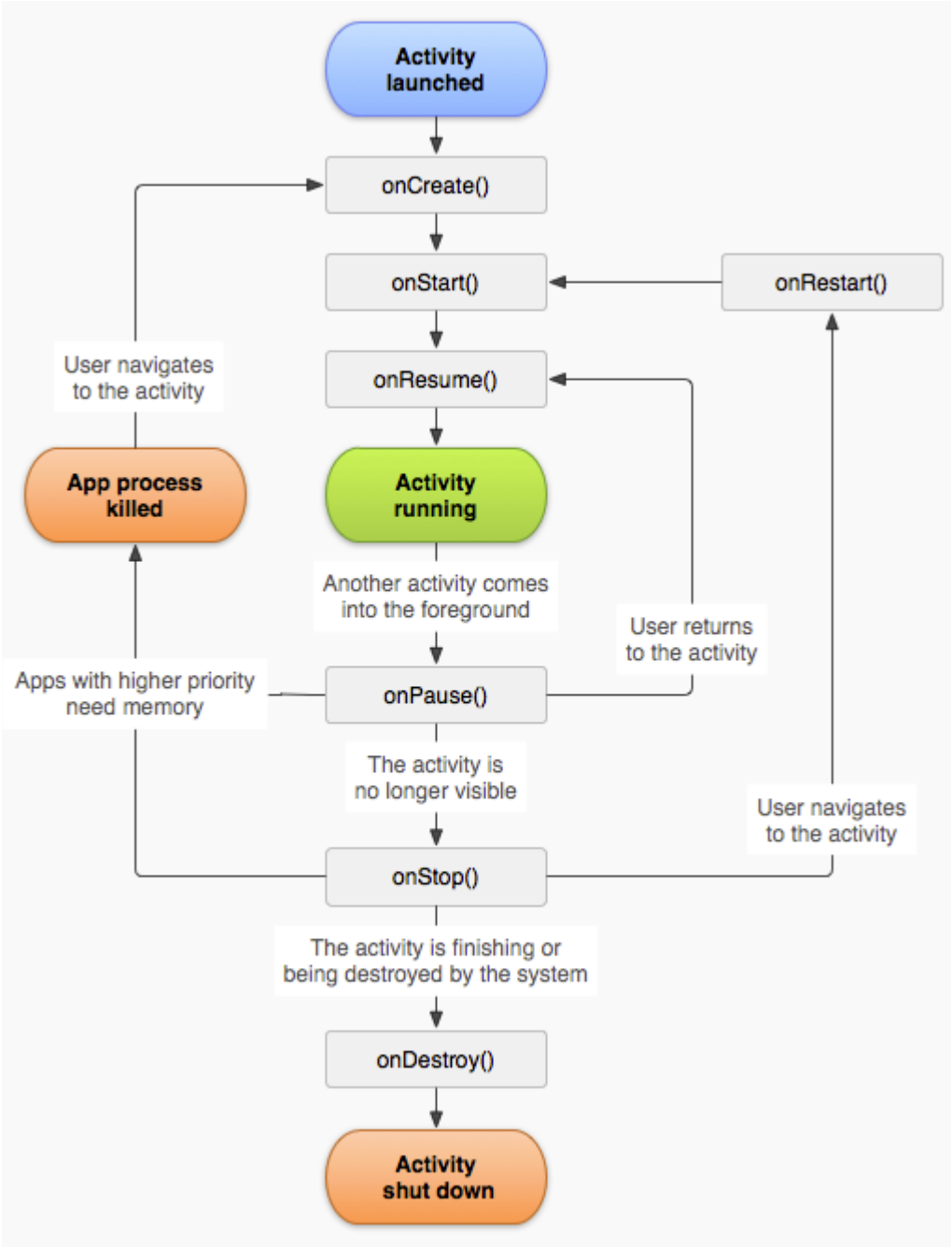
Projektin hakemistohierarkia kertoo Android-sovelluksen rakenteesta. AndroidManifest.xml-tiedostossa määritellään sovelluksen toteuttavat komponentit. Tiedostoon on määriteltävä muun muassa vähintään yksi komponentti, joka mahdollistaa sovelluksen käynnistyksen. Manifest-tiedostoon voidaan myös määritellä sovelluksen tukema minimi Android API-versio, sovelluksen tarvitsemat käyttöoikeudet sekä laitteisto ja ohjelmistovaatimukset, kuten näyttökoot. (Harju, J. 2013, hakupäivä 25.10.2013.)

Sovellushierarkian tärkeimpiä kansioita ovat muun muassa src-kansio, Android 4.3-kansio ja res-hakemisto. Src-kansio sisältää Java-luokkien lähdekoodimuotoiset tiedostot, Android 4.3-kansio sisältää valitun Android version mukaiset kirjastot ja res-hakemistoon sijoitetaan sovelluksen tiedostot, kuten kuvatiedostot ja sovelluksen näkymien asetteluun luodut XML-tiedostot (Harju, J. 2013, hakupäivä 25.10.2013).

3.2 Komponentit ja aktiviteetin elinkaari

Androidin sovelluslogiikka koostuu neljästä eri komponentista: aktiviteetti (Activity), palvelu (Service), sisällöntarjoaja (Content provider) ja vastaanottaja (Broadcast receiver). Aktiviteetti tuottaa sovelluksessa yhden näkymän ja voi sisältää käyttöliittymäelementtejä. Aktiviteetti periytetään Activity-luokasta. (Harju, J. 2013, hakupäivä 25.10.2013). Aktiviteetin kautta käyttäjä on vuorovaikutuksessa laitteiston kanssa ja se on tärkeä luokka osana ohjelman elinkaarta ajatellen (Vaara, S. & Vaara, V. 2011, hakupäivä 25.10.2013). Palvelu-komponentti ei ole vuorovaikutuksessa käyttäjän kanssa vaan se suoritetaan taustalla. Tällainen tapahtuma on esimerkiksi jokin synkronointi tietyin väliajoin. Sisällöntarjoajan avulla voidaan käsitellä esimerkiksi laitteelle tallennettuja ja jaettuja kuvia ja vastaanottaja-komponenttia käytetään järjestelmätasoisten viestien välittämiseen. (Harju, J. 2013, hakupäivä 25.10.2013.)

Aktiviteetin tila voi elinkaarensa aikana (kuvio 6) olla aktiivinen, keskeytetty, pysäytetty tai inaktiivinen. Siirtyminen eri vaiheesta toiseen tapahtuu kutsumalla tapahtumankäsittelijä luonteisia metodeja. Metodit ovat onCreate, onStart, onResume, onRestart, onPause, onStop ja onDestroy. Aktiviteetti käynnistetään luomalla Intent-luokasta olio. Luotu olio on viestinvälittäjä, jolla pyydetään komponentilta toimintaa. Kun aktiviteetti käynnistetään, se siirtyy aktiiviseen tilaan ja se tulee käyttäjälle näkyväksi ja se tulee ”aktiviteettipinon päällimmäiseksi”. Kun aktiviteetti lopetetaan, se poistetaan pinosta ja palautetaan pinosta päällimmäisin aktiviteetti näkyviin. (Harju, J. 2013, hakupäivä 25.10.2013.)



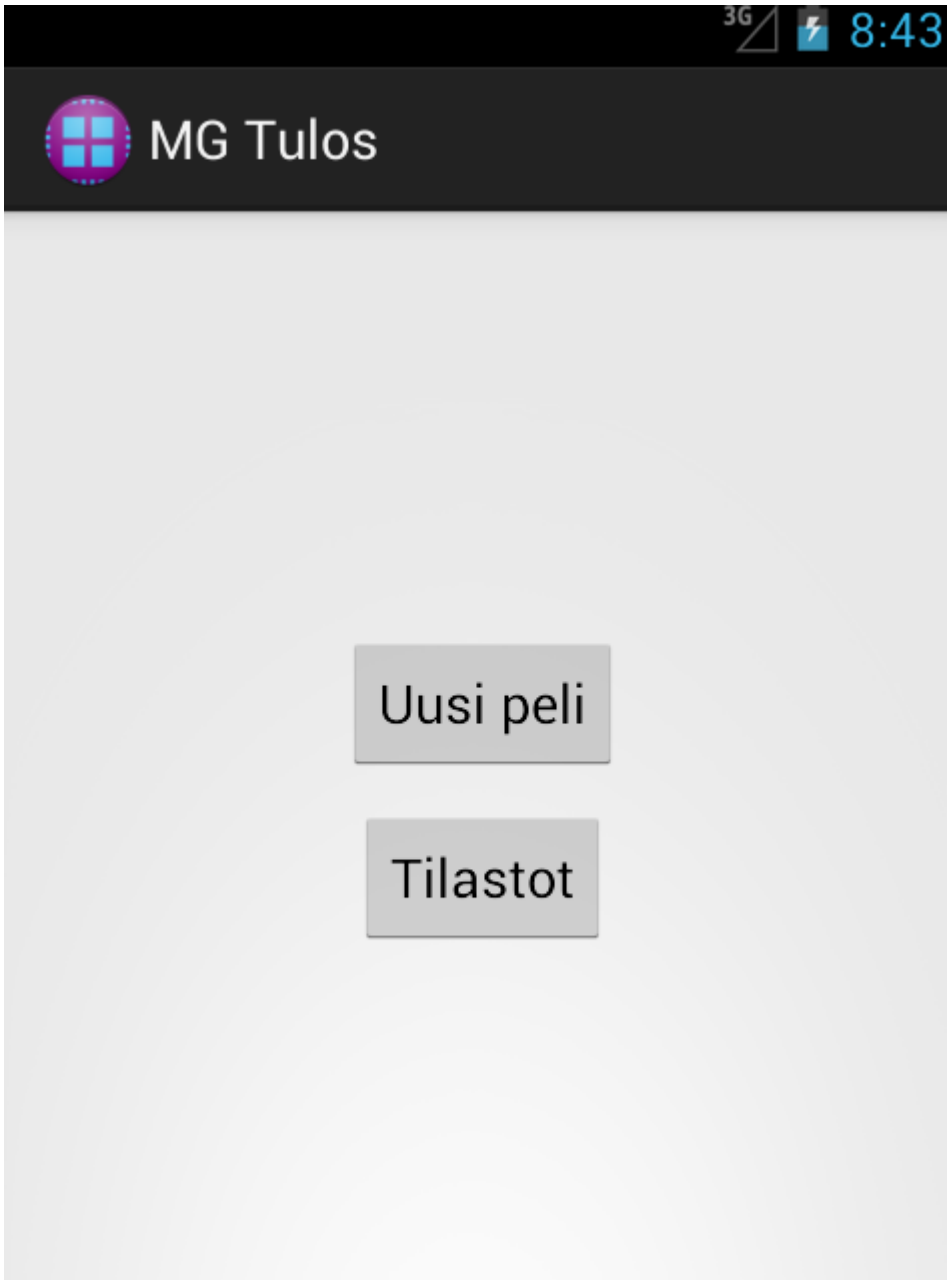
KUVIO 6. Aktiviteetin elinkaari (Android Developer 2013)

4 SOVELLUKSEN MÄÄRITTELY

Opinnäytetyön toiminnallinen osa on kehittää sovellus minigolfin pelaajille. Sovellus on tavallaan tietokantaohjelma, jolla voi tallentaa minigolfissa saadut pisteet puhelimen muistiin. Käyttäjät voivat olla eri-ikäisiä ja erilaisia. Sovellus pystyy tallentamaan samanaikaisesti vain yhden pelaajan tulokset pelin aikana. Käyttäjä määrittelee pelin alussa pelaavan henkilön ja pelattavan paikan. Ensimmäisiä kertoja sovellusta käyttävän käyttäjän täytyy tallentaa uusi pelaaja ja uusi paikka, mutta myöhemmin käyttäjä voi käyttää aiemmin tallennettua pelaaja-profiilia ja paikka-tietoa. Sovellus luo automaattisesti pelatulle pelille pelikerran, joka sisältää tiedon pelaajasta, pelipaikasta ja pelatuista radoista ja tallentaa myös pelatun päivän päivämäärän. Käyttäjä voi halutessaan tarkastella aiempien tallennettujen pelaajien ja pelikertojen tilastohistoriaa.

Sovelluksen käyttäjällä tulee olla Android-mobiililaite. Projektin luonnin alussa Minigolf-tulossovellukselle on määritelty alhaisimmaksi Android-versioksi 3.0, minkä kanssa sovelluksen tulee vähintään olla yhteensopiva. Testilaitteena sovelluskehityksessä on ollut mukana Samsung Galaxy S III-mobiililaite, missä versio on 4.1.2 ja tämä on määritelty myös projektin alussa kohdealustaksi. Api-taso on tällöin vähintään 11 ja korkeintaan 16. Jos halutaan, että sovellus tukee mahdollisimman montaa versiota, määritellään mahdollisimman alhainen versio.

Käyttöliittymässä tulee näkymään tarvittavat komponentit karkeasti aseteltuna. Seuraavassa kuviossa (kuvio 7) on näkymä sovelluksen päävalikosta. Päävalikko avautuu ensimmäisenä sovelluksen käynnistyttyä.



KUVIO 7. Päävalikko

Jokaiselle toiminnolle ohjelmoidaan oma aktiviteetti ja se tarvitsee näytölle tuotettavan näkymän. Näkymän asettelun määrittely on suositeltavaa tehdä XML-tiedostona, joka sijaitsee res-hakemistossa, mutta sen voi osalta määritellä myös aktiviteetissa Java-kielellä. XML-tiedostoon määritellylle elementille, johon tarvitsee viitata muista elementeistä tai Java-koodista, määritetään id-arvo android:id-attribuutilla. Asettelu otetaan Java-koodissa käyttöön kutsumalla aktiviteetin onCreate-metodissa setContentView(int layoutResId)-metodia ja elementtiin viitataan luomalla elementistä olio ja viittaamalla siihen findViewById-metodilla. (Harju, J. 2013, hakupäivä 25.10.2013.)

Yleisimmät asettelumallit ovat LinearLayout ja RelativeLayout.Lineaarisessa asettelussa elementit sijoitetaan peräkkäin joko vaaka- tai pystysuunnassa. Relatiivisessa asettelussa elementin sijainti voidaan määritellä suhteessa muihin elementteihin ja asettelun reunoihin. (Harju, J. 2013, hakupäivä 25.10.2013.) Minigolf-tulossovelluksessa tilastot listaava näkymä on toteutettu taulukkona määrittelemällä XML-tiedostoon taulukkonäkymä (TableLayout) ja määrittelemällä toiseen XML-tiedostoon sarake (TableRow) ja sarakkeelle kaksi tekstinäkymää lapsielementteinä (kuvio 8).

```
<TableRow xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/attrib_name"
        android:textStyle="bold"/>
    <TextView
        android:id="@+id/attrib_value"
        android:gravity="right"
        android:textStyle="normal"/>

</TableRow>
```

KUVIO 8. Sarakkeen määrittely XML-tiedostossa

Aktiviteetissä sarakkeesta voi luoda silmukassa (kuvio 9) olion tarpeiden mukaan.

```
for(int i = 0; i<c.getCount(); i++){
    TableRow row =  (TableRow)LayoutInflater.from(TilastotActivity.this).inflate(R.layout.attrib_row, null);
    ((TextView)row.findViewById(R.id.attrib_name)).setText("tekstiä");
    ((TextView)row.findViewById(R.id.attrib_value)).setText("teeeekstiä");
    table.addView(row);
}
table.requestLayout();
```

KUVIO 9. Sarake-olioiden luonti silmukassa

Yleisimmät elementit, joita minigolf-sovelluksessakin tullaan tarvitsemaan, ovat: painike (Button), syöttökenttä (EditText), tekstin esittämiseen tarvittava tekstikenttä-elementti (TextView) ja pudotusvalikko (Spinner). Muitakin on ja omien tekeminen ja valmiiden yhdistelykin on mahdollista. Eclipse tarjoaa graafisen käyttöliittymäeditorin, jolla elementtejä voi raahata työkalupaletista näkymään haluttuun kohtaan ja se generoi automaattisesti XML-tiedoston koodin. Koodia voi viimeistellä ja editoida myös manuaalisesti ei-graafisella puolella. (Harju, J. 2013, hakupäivä 25.10.213.)

Esimerkiksi painikkeelle voidaan asettaa tapahtumankäsittelijä onClick() joka on rajapinnan View.OnClickListener-metodi ja jota kutsutaan, kun käyttöliittymäelementtiä klikataan. OnClick-metodiin ohjelmoidaan toimenpiteet, jotka halutaan suorittaa, kun elementtiä klikataan. Minigolf-sovelluksessa on esimerkiksi painike ”uusi peli”, jonka halutaan avaavan näkymän uuden pelin luontia varten. Kohdesovelluksessa painike alustetaan aluksi aktiviteetti-luokassa, jossa painiketta käsitellään, Button-tyyppiseksi tietotyypiksi. onCreate-metodissa luodaan Button-elementistä, joka xml-tiedostoon on määritelty, olio id:n perusteella. Sen jälkeen painikkeelle on tehty kuuntelija ja tapahtumankäsittelijä (kuvio 10). Tapahtuman käsittelyyn on ohjelmoitu aktiviteetti avaamaan toisen aktiviteetin intentin avulla ja päättämään aktiviteetin, josta poistutaan.

```
btn_uusi_peli.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this, UusiPeliActivity.class);
        startActivity(intent);
        finish();

    }

});
```

KUVIO 10. Painikkeen kuuntelija ja tapahtuman käsittelijä

5 TIEDON TALLENTAMINEN JA KÄSITTELY

Androidille on tarjolla useita eri vaihtoehtoja sovelluksessa käsiteltävien tietojen tallennukseen. Tavan voi valita tarpeiden mukaan ja valintaan vaikuttaa muun muassa se, tarvitseeko tietoja käyttää myös muissa sovelluksissa ja kuinka paljon tilaa tietojen tallennus vaatii. Eräs tapa on kirjoittaa suoraan tiedostoon tai käyttää paikallista varastointimahdollisuutta tai siirrettävää muistia. Mahdollisuus on käyttää myös internetyhteyttä ja tallentaa tiedot omalle nettipalvelimelle. (Android Developer, hakupäivä 09.11.2013.) Minigolf-sovellus käyttää SQLite-tietokantaa ja senkin tekemiseen on monia eri tapoja.

Androidissa on täysi tuki SQLite-tietokannalle. Sovellukselle luotu tietokanta on sovellukselle yksityinen, eikä siihen pääse käsiksi muista sovelluksista. (Android Developer, hakupäivä 09.11.2013.) SQLite on avoimen lähdekoodin relaatiotietokanta ja se tukee standardeja relaatiotietokantojen ominaisuuksia. SQLite tietokanta vaatii vain vähän muistia suorittamiseen. Tuetut tietotyypit ovat TEXT, INTEGER ja REAL.

Kehitettävän sovelluksen tietokannan nimi on mg ja se on lyhenne sanasta minigolf. Ensimmäisessä versiossa sovellusta tullaan tarvitsemaan neljää taulua sovelluksen tietojen tallentamista varten. Taulut ovat pelaajien henkilötietojen tallentamiseen, paikkojen tietojen tallentamiseen, ratojen tulosten tallentamiseen sekä pelikertojen tallentamiseen. Tietokanta on kuvattu taulukkoina ja se on katseltavana raportin liitetiedostoissa ensimmäisenä liitteenä (liite 1).

5.1 Kustomoitu tietokanta-adapteri

Suositeltu tapa luoda tietokanta on tehdä alaluokka SQLiteOpenHelper-luokasta (Android Developer, hakupäivä 09.11.2013). Minigolf-sovelluksessa on kuitenkin tehty kustomoitu tietokanta-adapteri, joka on DBAdapter-niminen luokka. DBAdapter-luokkaan on määritelty alaluokka SQLiteOpenHelper-luokasta sisäisenä luokkana (kuvio 11). Sisäluokan konstruktorissa kutsutaan super()-metodia SQLiteOpenHelper-luokasta ja parametreina viedään tietokannan nimi ja versio. Taulujen luonti tapahtuu sisäluokan onCreate()-metodissa. OnUpgrade-metodia kutsutaan vain jos tietokannan rakenteeseen on tehty muutoksia ja tietokannan versiota kasvatetaan.

```
private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_PELAAJA);
        db.execSQL(CREATE_PAIKKA);
        db.execSQL(CREATE_VAYLA);
        db.execSQL(CREATE_PELIKERTA);


    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS pelaaja");
        db.execSQL("DROP TABLE IF EXISTS paikka");
        db.execSQL("DROP TABLE IF EXISTS vayla");
        db.execSQL("DROP TABLE IF EXISTS pelikerta");
        onCreate(db);

    }


}
```

KUVIO 11. DBAdapteri-luokan sisäluokka DatabaseHelper

Luokan staattiset jäsenmuuttujat eli tietokannan taulun nimet ja sarakkeiden nimet voi määritellä tietokantaa määriteltävän luokan alussa tai erilisissä sisäluokissa finaaleina muuttujina (Tommy 2011, hakupäivä 09.11.2013), mutta Minigolfin kustomoidussa adapterissa nämä on määritelty yhdessä taulun luontilauseen kanssa (kuvio 12).

```
public class DBAdapter{

    private static final String TAG = "DBAdapter";
    public static final String DATABASE_NAME = "mg.db";
    public static final int DATABASE_VERSION = 8;


    public static final String CREATE_PELAAJA =
            "create table pelaaja (pelaaja_id integer primary key autoincrement, " +
                    "pelaaja_nimi text not null);";
    public static final String CREATE_PAIKKA =
            "create table paikka (paikka_id integer primary key autoincrement, " +
                    "paikka_nimi text not null);";
    public static final String CREATE_VAYLA =
            "create table vayla (vayla_id integer primary key autoincrement, " +
                    "vayla_tulos_1 text, " +
                    "vayla_tulos_2 text, " +
                    "vayla_tulos_3 text);";
    public static final String CREATE_PELIKERTA =
            "create table pelikerta (pelikerta_id integer primary key autoincrement, " +
                    "pelikerta_paikka_id integer not null, " +
                    "pelikerta_pelaaja_id integer not null, " +
                    "pelikerta_vayla_id integer not null, " +
                    "pelikerta_pvm text);";

    private final Context context;
    private DatabaseHelper DBHelper;
    private SQLiteDatabase db;

    public DBAdapter(Context ctx) {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }
```

KUVIO 12. Taulujen ja sarakkeiden määrittely ja luontilauseet

DBAdapter-luokan lopussa on luodut metodit tietokannan avaamiselle ja sulkemiselle, tiedon lisäämiselle ja poistamiselle, päivittämiselle sekä korvaamiselle. Tietokannan avaamiseen tehdyssä metodissa luodaan olio DatabaseHelper:istä ja tehdään siitä kirjoitusoikeuksinen (kuvio 13).

```
//---opens the database---
  public DBAdapter open() throws SQLException {
      db = DBHelper.getWritableDatabase();
      return this;
  }
```

Kuvio 13. Open()-metodi

Suositeltu tapa tehdä kyselyt tietokantaan on määritellä query()- ja rawQuery()-metodit. Jälkimmäiseen voi kyselyn antaa argumenttina suoraan String-muodossa. Paluuarvona saadaan Kursori-luokan (Cursor) ilmentymä. (Vogel, L. 2013, hakupäivä 09.11.2013.) Tulossovelluksessa metodit ovat get() ja rawQuery() (kuvio 14).

```
public Cursor rawQuery(String text) {
    Cursor mCursor = db.rawQuery(text, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

public Cursor get(String tablenames, String[] wantedRowNames, String whereClause,
        String orderBy, String limit) throws SQLException {
    Cursor mCursor = db.query(
                    tablenames,
                    wantedRowNames,
                    whereClause,
                    null,
                    null,
                    null,
                    orderBy,
                    limit);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}
```

Kuvio 14. Kysely-metodit

Kursori on kyselyn palauttama objekti joka esittää kyselyn tuloksen ja osoittaa periaatteessa yhteen riviin kyselyn tuloksesta. Useita elementtejä voi saada käyttämällä getCount()-menetelmää. Yksittäisellä rivillä siirtyminen voidaan tehdä moveToFirst() ja moveToNext() metodeilla. (Vogel, L. 2013, hakupäivä 09.11.2013.)

5.2 Adapterin käyttäminen

Minigolf-sovelluksen aktiviteetissa, jossa käsitellään tietokantaa, luokan alussa luodaan DBAdapter-luokasta olio (kuvio 15).

```
DBAdapter db = new DBAdapter(this);
```

KUVIO 15. DBAdapter-luokan olion luonti

Tietokanta avataan aktiviteetin onCreate()-metodissa kutsumalla DBAdapter-luokan metodia open(). Esimerkiksi kaikki pelikerrat voidaan hakea tietokantayhteyden avaamisen jälkeen luomalla kursoriluokasta olio ja kutsumalla metodia get(). Get()-metodille viedään argumentteina taulun nimi, halututu sarakkeet sekä määritellään halutessaan ehtolause, järjestys ehto ja raja (kuvio 16).

```
private void hae_tilastot() {
    try{
        String[] select = new String[] {"pelikerta_id, pelikerta_paikka_id, pelikerta_pelaaja_id"};
        String from = "pelikerta";
        Cursor c = db.get(from, select, null, null, null);
```

KUVIO 16. Kysely pelikerta-tauluun

Kuviossa 16 (kuvio 16) määritellään halutut sarakkeet String-tyyppiseen taulukkoon select ja taulun nimi String-tyyppiseen muuttujaan from. Samalla lailla myös esimerkiksi ehtolause määriteltäisiin esimerkiksi String-tyyppiseen muuttujaan where ja se vietäisiin kolmantena argumenttina get()-metodissa.

Minigolf-sovelluksessa kyselyn palauttamat tulokset on käsitelty esimerkiksi do-while-toistorakenteessa (kuvio 17). Silmukassa loppuehdoksi on määritelty kursorin metodi moveToNext() ja toisto päättyy kun kaikki osoittimen palauttamat rivit on käyty yksitellen läpi. Osoittimen palauttama rivi on tallennettu Vector-tyyppiseen dynaamiseen taulukkoon.

```
do{
    vector_paikka_id.add(c.getInt(0));
    vector_paikka_nimi.add(c.getString(1));
}while(c.moveToNext());
```

KUVIO 17. Do-while-toistorakenne osoittimen palauttamien rivien läpikäyntiin

Esimerkiksi uuden pelaajan lisääminen on sovelluksessa viety tietokantaan insert()-metodilla dynaamista vector-taulukkoa apuna käyttäen (kuvio 18). Insert()-metodissa argumentteina viedään String-tyyppisenä tietona taulun nimi, taulukkona sarakkeet ja data. Pelaaja-tauluun pelaaja_id on määritelty automaattisesti kasvavana järjestyslukuna (autoincrement), joten SQLite antaa automaattisesti seuraavan mahdollisen järjestysluvun pelaajan id:ksi. Tämä uusi id on haettu int-tyyppiseen muuttujaan kuviossa näkyen get()-metodilla ja metodille argumenttiin sarakkeelle on käytetty MAX-koostefunktiota.

```
Vector<String> row = new Vector<String>();
Vector<String> data = new Vector<String>();

row.add("pelaaja_nimi");
data.add("" + edit_nimi.getText());

db.insert("pelaaja", row, data);

int new_pelaaja_id = db.get("pelaaja", new String[] {"MAX(pelaaja_id)"},
        null, null, null).getInt(0);
```

KUVIO 18. Insert()-metodin käyttäminen

Taulun päivittäminen tapahtuu miltei samantapaisesti kuin tiedon lisääminen tauluun. Päivittäminen tapahtuu update()-metodilla ja argumentteina viedään taulun nimi, sarakkeiden nimet ja datat ja lisäksi vaaditaan ehtolause. Esimerkiksi pelikertatauluun lisätään pelikerta_vayla_id:ksi uusi väylä_id (new_vayla_id) ja ehtona on, että pelikerta_id on sama kuin tässä tapauksessa haluttu uusimman lisätyn pelikarran id (new_pelikerta_id) (kuvio 19).

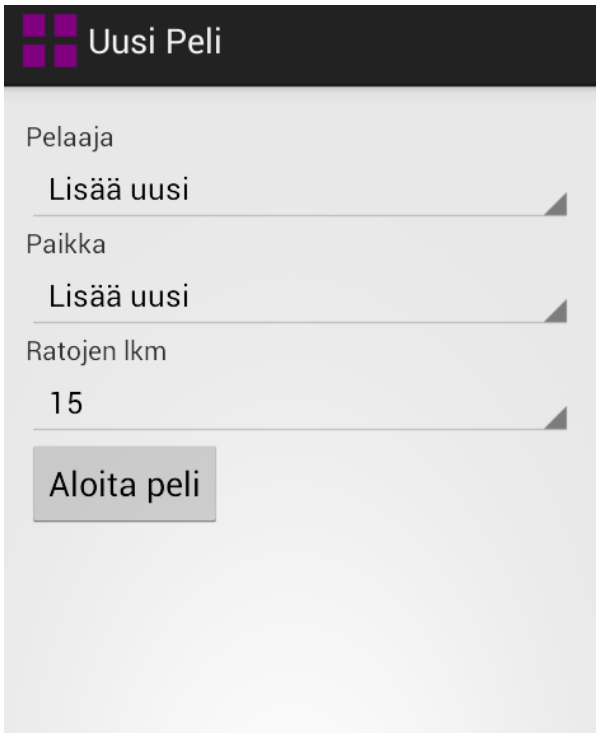
```
Vector<String> uprow = new Vector<String>();
Vector<String> updata = new Vector<String>();

uprow.add("pelikerta_vayla_id");
updata.add("" + new_vayla_id);

db.update("pelikerta", uprow, updata, "pelikerta_id = '" + new_pelikerta_id + "'");
```

KUVIO 19. Update()-metodin käyttäminen

Mikäli jonkin taulun jokin tietty rivi haluttaisiin poistaa, voitaisiin poistaminen tehdä miltei saman tapaisesti kuin edellä päivittäminen. Poistamista varten kutsuttaisiin metodia delete() ja argumentteina vietäisiin taulun nimi ja ehto.

6 VALMIS SOVELLUS

Sovellus toimii kaikin puolin tavoitteiden mukaan ja on nyt esiteltävässä muodossaan oikeastaan jonkilainen prototyyppi/testiversio. Kun sovellus käynnistyy, ensimmäisenä näkymänä avautuu päävalikko. Päävalikosta voi valita kahdesta painikkeesta joko uuden pelin lisäämisen tai tilastojen selauksen. Uuden pelin lisäysnäkymässä on pudotusvalikot pelaajille, paikoille ja myös ratojen lukumäärille (kuvio 20).



KUVIO 20. Uuden pelin lisäys -näkymä

Ensimmäisen kerran sovellusta avattaessa tietokannassa ei ole aiemmin tallennettuja pelaajia eikä paikkoja. Myöhemmin, kun aiemmin tallennettuja pelaajia ja paikkoja löytyy tietokannasta, ne näkyvät pudotusvalikoissa. Seuraavassa kuviossa (kuvio 21) on pelaaja- ja paikka-taulun sisältö SQLite Data Browser-ohjelman avulla tarkasteltavana.

Table: pelaaja

	pelaaja_id	pelaaja_nimi
1	1	Erkki
2	2	Krista
3	3	Matti

Table: paikka

	paikka_id	paikka_nimi
1	1	Oulu Hukka
2	2	Oulu Toppila

KUVIO 21. Pelaaja- ja paikka-tauluun tallennetut rivit

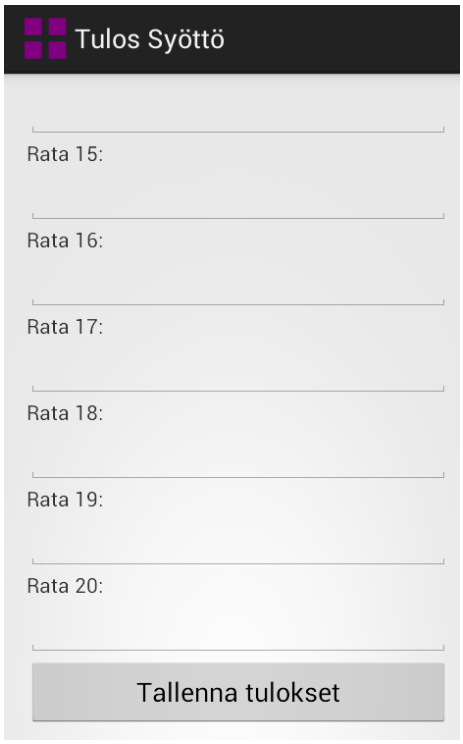
Käyttäjän painettua ”Aloita peli”-painiketta sovellus tallentaa kantaan uuden rivin Pelikerta-tauluun. Pelikerta tauluun viedään pelaaja_id ja paikka_id sekä ratojen lukumäärä (kuivo 22). Käytännössä tässä vaiheessa, jos kuvion neljäs rivi olisi juuri edellä mainittu uusi rivi, olisi pelikerta_valya_id-sarake tyhjä.

Table: pelikerta New Record Delete Record

	pelikerta_id	pelikerta_paikk	pelikerta_pelaaj	pelikerta_vayla	pelikerta_rata_l	pelikerta_pvm
1	1	1	1	1	3	31.10.2011
2	2	1	1	2	8	16.07.2013
3	3	1	2	3	18	23.09.2013
4	4	2	3	4	18	20131110

KUVIO 22. Pelikerta-tauluun tallennetut rivit

Samalla avautuu myös uusi näkymä ”Tulos Syöttö” pisteiden syöttöä varten (kuvio 23).



KUVIO 23. Pisteiden syöttö –näkymä

Tulos Syöttö-näkymän avautuessa sovellus luo tietokantaan jo uuden rivin ratojen tuloksille. Uusi rivi luodaan vayla-tauluun ja uusi automaattisesti luotu vayla_id päivitetään samalla pelikerta-tauluun. Tulos Syöttö-näkymää on mahdollista rullatta ylös tai alas ja edellisessä näkymässä määritelty ratojen lukumäärä vaikuttaa syötettävien tekstikenttien määrään. Tekstikenttiin voi syöttää ainoastaan numeerista tietoa ja oikella laitteella testattuna napsauttamalla jotain editoitavaa tekstikenttää avautuu näkyville vain numeronäppäimistö. Kentässä ei ole syötteen tarkistusta, joten käyttäjällä on mahdollisuus kirjata yhdeksi ratatulokseksi vaikkapa 9, mikä ei oikeasti olisi mahdollista minigolf-pelissä. Tulokset tallennetaan tietokantaan painamalla painiketta ”Tallenna tulokset”. Tulokset tallentuvat juuri aiemmin luodun vayla_id:n perusteella vayla-tauluun. Samalla sovellus laskee saadut pisteet yhteen ja tallentaa myös sen vayla-tauluun vayla_tulos_yht-sarakkeeseen. Seuraavassa kuviossa (kuvio 24) näkyy osa valya-taulun sarakkeista ja tauluun tallennetut rivit.

Table: vayla

New Record | Delete Record

	vayla_id	vayla_tulos_1	vayla_tulos_2	vayla_tulos_3	vayla_tulos_4	vayla_tulos_5	vayla
1	1	4	7	2	2	2	
2	2	4	7	2	2	2	
3	3	4	7	2	2	2	
4	4	5	3	3	6	5	

KUVIO 24. Vayla-tauluun tallennetut rivit

Kun sovelluksen on tallentanut tiedot kantaan, avautuu automaattisesti tilastojen selaus näkymä (kuvio 25). Tilasto-näkymän rivit ovat käytännössä pelikertoja. Riviin on liitetty kuuntelija, joten haluttua pelikertaa klikkaamalla avautuu näkymä, jossa kaikki sen pelikerran pisteet on listattuna alekain.

Tilastot

Pelaaja	Paikka	Aika	Pisteet
Erkki	Oulu Hukka	31.10.2011	89
Erkki	Oulu Hukka	16.07.2013	102
Krista	Oulu Hukka	23.09.2013	231
Matti	Oulu Toppila	20131110	68

KUVIO 25. Tilasto -näkymä

7 POHDINTA

Opinnäytetyön tekeminen oli erittäin raskasta ja haastavaa aikaa. Erittäin vaikeaa oli täyspäiväisen työn lisäksi löytää aikaa opinnäytetyön tekemiselle. Prosessia ryhditti se, että ohjaava opettaja jaksoi aktiivisesti pitää säännöllisin ja lyhin väliajoin ylimääräisen katselmoinnin, minkä ansiosta projekti eteni varmasti. Mobiiliohjelmointi Androidille oli sinänsä erittäin mielenkiintoinen aiheena ja sovelluskehityksen elinkaari suunnittelusta toteutukseen oli helppo rajata opinnäytetyöksi. Tietoperustaa oli mukava kerätä ja sovelluksen ohjelmointi oli haastavaa mutta hauskaa. Hankalinta oli sovelluksen toteutuksen yhdistäminen tietoperustaan ja omien ratkaisuiden mahdollinen perustelu raportin muodossa.

Tietokantaohjelmoinnista Androidille oli mielestäni yllättävän vähän materiaalia enkä varsinkaan löytänyt sellaista materiaalia, joka olisi tukenut minun tapaani toteuttaa Androidille sql-tietokanta ja sen käsittelymetodit. Opin tyylin tietokantatoteutukselle työharjoittelussa ja rajallisen aikataulun vuoksi käytin sitä myös opinnäytetyössäni. Suosituin ja suositelluin tapa näytti olevan toteuttaa suoraan tietokanta-apuluokka, joka on laajennos SQLiteOpenHelper-luokasta. Tämän lisäksi suositeltiin tehtäväksi jonkinlainen datan käsittely luokka ja ymmärtääkseni myös erilliset luokat taulukohtaisesti. Olisi ollut toki mielenkiintoinen perehtyä tähän tapaan ja kokeilla tehdä sillä omankin sovelluksen tietokanta, mutta päätin heti alkuun pitäytyä tutussa opitussa tavassa.

Jatkossa, mikäli sovelluksen kehittämistä vielä opinnäytetyöprosessin jälkeen jatkettaisiin, olisi ehdottoman tärkeää kiinnittää huomio ennen kaikkea aluksi pieniin yksityiskohtiin, jotka jäivät harmillisesti prototyyppisestä sovellusversiosta pois ja jotka ovat mielestäni erittäin tärkeitä sovelluksen vakaan ja luotettavan toiminnan kannalta. Esimerkiksi syöttötietojen tarkastus tulisi liittää kenttiin, joissa otetaan vastaan käyttäjän syöttämää tietoa. Myös tietokantatapahtumiin (transactiot) tulisi kiinnittää huomiota, jotta tietokanta pysyisi luotettavana ja eheänä tietojen lisäilyjen ja päivitysten yhteydessä. Kun nämä pienet asiat on korjattu, olisi tietenkin ehdottoman tärkeää kiinnittää huomio sovelluksen ulkonäköön ja tehdä siitä miellyttävän ja mielenkiintoisen näköinen visuaalisesti. Suurimpana kehityskohteena olisi hyvä ottaa huomioon myös mobiililaitteen käyttäjän seurassa pelaavat henkilöt ja heidän mahdollisuutensa hyödyntää samaa ohjelmaa pelin aikana joko niin, että yhdellä mobiililaitteella voisi tallentaa useamman pelaajan tuloksia yhtä aikaa.

Lisäksi sovelluksen tietokannan synkronointi jollekin nettipalvelimelle lisäisi uusi mahdollisuuksia jakaa ja käyttää tietokantaan tallennettuja tietoja.

LÄHTEET

Android Developer. 2013a. Activity. Hakupäivä 17.10.2013, http://developer.android.com/reference/android/app/Activity.html.

Android Developer. 2013b. Reference. Hakupäivä 17.10.2013, http://developer.android.com/reference/packages.html.

Android Developer. Storage Options. Hakupäivä 09.11.2013, http://developer.android.com/guide/topics/data/data-storage.html. (Ei julkaisuvuotta)

Android Developer. What is API Level? Hakupäivä 17.10.2013, http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels. (Ei julkaisuvuotta).

Android Suomi. Mikä on Android? Hakupäivä 11.10.2013, http://blog.androidsuomi.fi/mika-on-android/. (Ei julkaisuvuotta).

Deleon, Walter. 2013. A Brief History Of Android. Hakupäivä 07.11.2013, http://technoblimp.com/2013/09/14/a-brief-history-of-android/.

eLinux Wiki . 2011. Android Architecture. Hakupäivä 7.10.2013, http://elinux.org/Android_Architecture.

Gartner, Inc. 2013. Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012. Hakupäivä 17.10.2013, http://www.gartner.com/newsroom/id/2335616.

Hutunki. Lajitiedot. Minigolf. Hakupäivä 11.10.2013, http://www.hutunki.fi/lajitiedot/minigolf. (Ei julkaisuvuotta).

Hynninen, Teemu. 2009. Androidin eri versiot vaikeuttavat sovellusten kehittämistä? Hakupäivä 07.11.2013, http://www.mobiiliblogi.com/2009/12/15/androidin-eri-versiot-vaikeuttavat-sovellusten-kehittamista/.

Hynninen, Teemu. 2013. Androidin historia: Astrosta Jelly Beaniin. Hakupäivä 11.10.2013, http://www.mobiiliblogi.com/2013/07/20/androidin-historia-astrosta-jelly-beaniin/.

Jyväskylän Yliopisto. 2011. Androidin ohjelmointia. Hakupäivä 11.10.2013, https://trac.cc.jyu.fi/projects/ohj2/wiki/Android.

Sibeliuspuiston minigolf. Historia. Hakupäivä 11.10.2013, http://www.sibeliuspuistonminigolf.fi/historia. (Ei julkaisuvuotta).

TIOBE. 2013. TIOBE Programming Community Index for October 2013. Hakupäivä 1.11.2013, http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

Tommy. QVIK. 2011. Android SQL-tietokannan käyttäminen. Hakupäivä 09.11.2013, http://mobiilikehitys.fi/android-sql-tietokannan-kayttaminen/.

Tutorialspoint. 2012. Android Architecture. Hakupäivä 17.11.2013, http://www.tutorialspoint.com/android/android_architecture.htm.

Valkjärvi, Jouni & Metsäranta, Timo. 2007. Suomen Ratagolfliitto. Ratagolf ja kilpaileminen. Hakupäivä 11.10.2013, http://www.kissanpaivia.com/minigolf/RATAGOLF.pdf.

Vaara, Santeri & Vaara Visa, Jyväskylän Yliopisto. 2011. Google Android -ohjelmistokehitys. Hakupäivä 25.10.2013, http://www.mit.jyu.fi/opiskelu/seminaarit/tiesem2011/Andoidsovelluskehitys.html.

Wikipedia. 2013a. Android (operating system). Hakupäivä 5.5.2013, http://en.wikipedia.org/wiki/Android_%28operating_system%29

Wikipedia. 2013b. Minigolf. Hakupäivä 11.10.2013, http://fi.wikipedia.org/wiki/Minigolf.

LIITTEET

SOVELLUKSEN TIETOKANTAKUVAUS LIITE 1

TAULUKKO 1. Pelaaja-taulu sarakkeiden määrittely

Taulu: **PELAAJA** Kuvaus: Sisältää tallennettujen pelaajien henkilötiedot			
Sarake	Kuvaus	Tietotyyppi	Pakollisuus
pelaaja_id	Yksilöi pelaajan, pelaajan id (perusavain)	Numeerinen	Kyllä
pelaaja_nimi	Pelaajan koko nimi	Merkkijono	Kyllä

TAULUKKO 2: Paikka-talun sarakkeiden määrittely

Taulu: **PAIKKA** Kuvaus: Sisältää tallennettujen pelipakkojen tiedot			
Sarake	Kuvaus	Tietotyyppi	Pakollisuus
paikka_id	Yksilöi paikan, paikan id (perusavain)	Numeerinen	Kyllä
paikka_nimi	Paikan nimi, vapaa tekstimuoto	Merkkijono	Kyllä

TAULUKKO 3: Väylä-taulun sarakkeiden määrittely

Taulu: **VÄYLÄ** Kuvaus: Sisältää väylältä käyttäjän tallentamat pisteet. Yksi tallennettu väylätulos luo väylälle yksilöllisen id:n ja se yksilöi samalla kaikkien reikien pisteet.			
Sarake	Kuvaus	Tietotyyppi	Pakollisuus
vayla_id	Yksilö väylät, väylien yhteinen id (perusavain)	Numeerinen	Kyllä
vayla_tulos_1	Pelipaikan reiältä 1 saatu pistemäärä	Numeerinen	Ei
vayla_tulos_2	Pelipaikan reiältä 2 saatu pistemäärä	Numeerinen	Ei
vayla_tulos_3	Pelipaikan reiältä 3	Numeerinen	Ei

	saatu piste-määrä		
vayla_tulos_4	Pelipaikan reiältä 4 saatu pistemäärä	Numeerinen	Ei
..tulos_5….tulos_18..:	Pelipaikan reiältä 5 … 18 saatu piste-määrä	Numeerinen	Ei
vayla_tulos_19	Pelipaikan reiältä 19 saatu pistemäärä	Numeerinen	Ei
vayla_tulos_20	Pelipaikan reiältä 20 saatu pistemäärä	Numeerinen	Ei

TAULUKKO 4: Pelikerta-taulun sarakkeiden määrittely

Taulu: **PELIKERRAT** Kuvaus: Sisältää käyttäjän tallentaman pelikerran. Pelikerta –tieto sisältää pelin pelaajan id:n, pelatun paikan id:n, pelatun väylän id:n			
Sarake	Kuvaus	Tietotyyppi	Pakollisuus
pelikerta_id	Pelikerran yksilöivä id, pelikerta id (pe-rusavain)	Numeerinen	Kyllä
pelikerta_paikka_id	Pelatun paikan id, viittaa tauluun Paik-ka	Numeerinen	Kyllä
pelikerta_pelaaja_id	Pelatun pelikerran pelaajan id, viittaa tauluun Pelaaja	Numeerinen	Kyllä
pelikerta_vayla_id	Pelattujen reikien vayla_id, viittaa tauluun Vayla	Numeerinen	Kyllä
pelikerta_rata_lkm	Pelipaikan ratojen lukumäärä	Numeerinen	Ei
pelikerta_pvm	Pelatun pelikerran päivämäärä	Merkkijono	Ei