Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Tietojärjestelmät

2016

Marko Lehtinen

FRISBEEGOLF-TULOSSOVELLUKSEN TIETOKANNAN LUONTI ANDROIDILLA

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelym koulutusohjelma | Yrityksen tietojärjestelmät

2016 | 29

Anne Jumppanen

Marko Lehtinen

FRISBEEGOLF-TULOSSOVELLUKSEN TIETOKANNAN LUONTI ANDROIDILLA

Työn tarkoituksena oli ohjelmoida yksinkertainen älypuhelimeen sovellusprototyyppi, jolla frisbeegolfin pelaajat voivat tallentaa tuloksia. Kohdealustana toimii Android. Sovellus määriteltiin yhteensopivaksi Android-version 4.1 tai sitä uudemman kanssa.

Opinnäytetyössä tarkastellaan mobiiliohjelmointia Androidilla sekä tietokantaohjelmointia SQLite-tietokantajärjestelmän avulla. Opinnäytetyöllä ei ole toimeksiantajaa, vaan aihe valittiin omasta mielenkiinnosta.

Työn tuloksena syntyi valmis prototyyppi, kuten alun perin oli tarkoitus. Työssä käydään läpi Androidin historiaa ja esitellään tietokannan toteutus selkein kuvin. Tietokannan luonnin ohella luodaan sovellukseen käyttöliittymä.

Mikäli sovellusta vielä kehitetään, lisättäisiin tiedontallennus funktiot, jotta tietokantaan voidaan lisätä tietoja. Samalla kehitettäisiin käyttöliittymää, jotta tämä olisi mahdollista. Sovelluksen ulkonäköön tultaisiin myös kiinnittämään huomiota.

ASIASANAT:

Mobiili, Android, SQLite, tietokanta, Android Studio, mobiiliohjelmointi

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | Information Systems

2016 | 29

Anne Jumppanen

Marko Lehtinen

DISC GOLF-RESULT APPLICATION DATABASE CREATION FOR ANDROID

The purpose of this thesis was to program a simple prototype application on which disc golfers can save their results on their smartphone. The target platform was Android. The application was required to be compatible with Android-version 4.1 or above.

The thesis examines Android mobile programming and database programming with the SQLite database system. The thesis also introduces how Android has developed and how the data storage is implemented. The result of this project was a complete prototype as was originally intended.

In the future, if the application continues to be developed, there would be functions for saving data. At the same time, the user interface would also need to be developed. Attention to the appearance of the application would also be beneficial for the future development of the application.

KEYWORDS:

Mobile, Android, SQLite, database, Android Studio, mobile programming

SISÄLTÖ

KUVAT

KÄYTETYT LYHENTEET JA SANASTO

Android Manifest	Tiedosto, jossa kuvataan Android-sovelluksen komponentit.
Android Runtime (ART)	Sovelluksen ajonaikainen ympäristö.
Android Studio	Työkalu Android-ohjelmointiin.
Bug	Bug tai Bugi on virhe koodissa, joka saa sovelluksen käyttäytymään virheellisesti tai ei-toivotulla tavalla.
Dalvik-Virtuaalikone	Virtuaalikone, jonka päällä Android-sovellukset toimivat.
Emulaattori	Virtuaalikone, joka jäljittelee jonkin laitteen toimintaa.
HTML	Hypertext Markup Language. Merkintäkieli www-dokumenttien ohjelmointiin.
Instanssi	Tarkoittaa olio-ohjelmoinnissa luokan edustajaa.
Java-kieli	Laitteistoriippumaton oliopohjainen ohjelmointikieli.
Just-in-Time (JIT)	Ajonaikainen kääntäminen.
WiFi hotspot	Fyysinen sijainti, josta voi saada Internet-yhteyden.

1 JOHDANTO

Frisbeegolf on vielä varsin uusi laji suurelle yleisölle. Siitä huolimatta sen suosio kasvaa huimaa vauhtia. Frisbeegolfin pääperiaate on sama kuin perinteisessä golfissa. Kiekko pitää saada mahdollisimman vähin heitoin aloituspaikalta koriin. Pienimmän tuloksen saanut luonnollisesti voittaa.

Frisbeegolfissa käytetään kiekkoja, jotka voidaan jakaa karkeasti kolmeen eri ryhmään: draiverit, eli long range-kiekot, on tarkoitettu pitkän matkan heittoihin, midarit, eli mid range-kiekot, lähestymisheittoihin ja putteri, eli putter, lähipeliin, käytännössä puttaamiseen.

Tämän opinnäytteen tarkoituksena on kehittää tietokanta frisbeegolf-aiheiselle sovellusprototyypille Androidilla. Samalla sovellukselle luodaan myös käyttöliittymä jatkokehitystä ajatellen. Valitsin aiheen omasta mielenkiinnosta mobiiliohjelmointia kohtaan, ja tavoitteena on kehittää itseäni ohjelmoijana.

Teoreettisessa osassa kerrotaan pääpiirteittäin Androidin historiasta ja sen kehityksestä vuosien varrella. Työssä esitellään käytettävät työkalut ja valittu työympäristö. Työympäristöksi valikoitui Android Studio, joka on tällä hetkellä Googlen tarjoama virallinen kehitysympäristö. Android tarjoaa muutenkin laajan valikoiman kehitystyökaluja ja on panostanut niiden helppokäyttöisyyteen.

Empiirisessä osassa keskitytään tietokannan luontiin SQLiten avulla. Raportissa käydään läpi, mikä SQLite on ja mitä sillä voidaan tehdä. Tietokannan luontia on pyritty selkeyttämään kuvilla. Tiedontallennusta ei tässä opinnäytteessä käsitellä.

2 ANDROID

Android on Googlen julkaisema mobiililaitteille suunnattu, avoimeen lähdekoodiin perustuva ohjelmistopino, joka perustuu Linuxiin. Käyttöjärjestelmän lisäksi Android sisältää perusohjelmia ja väliohjelmistoja. Suurimpia Android-puhelin valmistajia ovat tällä hetkellä HTC, LG, Motorola, Samsung ja Sony. Kaikki Android-puhelimet ovat pohjimmiltaan älypuhelimia. (Android Suomi 2013.)

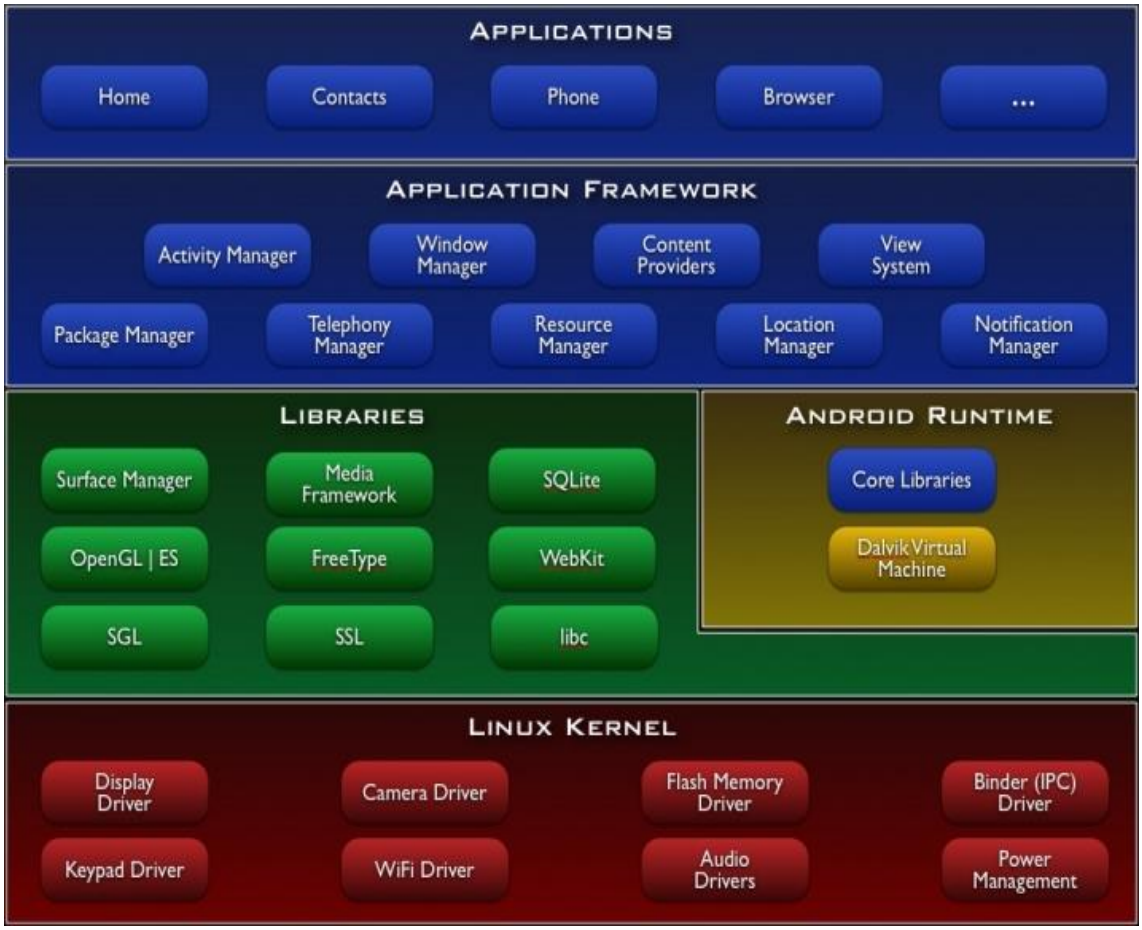
Android- puhelimille on saatavilla runsaasti sovelluksia Google Play Storen kautta. Lataamista varten tarvitaan Google-tili. Sovelluksia on sekä ilmaisia että maksullisia. Kaikki ladatut sovellukset liitetään Google-tiliin, joten ne ovat aina uudelleenladattavissa, vaikka puhelin vaihtuisikin. (Android Suomi 2013.)

2.1 Historia

Alun perin Androidin kehityksestä vastasi Android Inc., mutta Google osti sen vuonna 2005 ja siirsi samalla kehitysvastuun Open Handset Alliancelle. Nykyisin Open Handset Alliance koostuu 84 eri yrityksen yhteenliittymästä, joiden tavoitteena on kehittää kuluttajille parempi kokonaisuus mobiilikäytöstä. (Open Handset Alliance 2007.)

2.2 Arkkitehtuuri

Kuvasta 1 voidaan huomata, mistä pääelementeistä Android-käyttöjärjestelmän arkkitehtuuri muodostuu.



Kuva 1. Android-järjestelmän arkkitehtuuri (eLinux 2011).

Applikaatiokerros (Applications) on Android-järjestelmän ylin kerros. Tämä kerros muodostuu sekä Androidin alkuperäisistä ohjelmista, kuten selaimesta ja sähköpostisovelluksesta, että kolmannen osapuolen sovelluksista, joita käyttäjä on itse asentanut puhelimeen. (Techotopia 2014.)

Applikaation ohjelmistokehys (Application Framework) on useimmiten ohjelmistokehittäjien käytössä. Kaikki sovellukset ovat jatkuvassa vuorovaikutuksessa tämän tason kanssa, mikä mahdollistaa puhelimen perustoiminnot soittamisesta kuvien ottamiseen. (Sharma 2013.)

Kirjasto (Libraries) käsittää Androidin valmiit kirjastot. Kirjastot sisältävät joukon käskyjä, jotka määrittävät, miten laite käsittelee minkäkin tyyppistä dataa. Esimerkiksi audio ja videoformaattien toisto tapahtuu media viitekehyksen kautta. (Sharma 2013.)

Ajonaikaiset komponentit (Android runtime) ovat kirjaston kanssa samalla tasolla. Tässä osiossa keskeinen tekijä on Dalvik-Virtuaalikone. Dalvik-Virtuaalikone on eräänlainen Java-virtuaalikone, joka on suunniteltu ja optimoitu erityisesti Androidia varten. (Techotopia 2014.)

Dalvik-Virtuaalikone käyttää Linuxin ydinominaisuuksia, kuten muistinhallintaa ja monisäietekniikkaa, joka on luontainen asia Java-kielellä. Dalvik-Virtuaalikone mahdollistaa jokaisen Android applikaation ajon omassa prosessissaan, jolla on oma instanssi Dalvik-Virtuaalikoneeseen. (Techotopia 2014.)

Linux-ydin (Linux Kernel) sijaitsee koko joukon alimmaisena kerroksena. Se ei juurikaan kommunikoi käyttäjän tai kehittäjien kanssa, mutta on koko systeemin ydin. (Techotopia 2014.)

2.3 Versiot

Android 1.0 (Astro) lanseerattiin syyskuussa 2008. Sen ominaisuudet oli lähinnä suunnattu yrityksiä ja työntekoa varten. Tämä versio oli täynnä hyödyllisiä sovelluksia ja omasi tuen Internetiin ja Internet-pohjaisiin palveluihin, kuten: täysi HTML-tuki selaimessa, YouTube-sovelluksen ja Googlen palveluita, kuten Gmail, Google Maps ja Google Sync. Se oli myös ensimmäinen mobiilikäyttöjärjestelmä, jossa oli ruudun yläosasta alas vedettävä pudotusvalikko**.** (Chowdhury 2012.)

Android 1.5 (Cupcake) julkaistiin huhtikuussa 2009 ja se oli toinen merkittävä julkaisu, mutta vasta ensimmäinen kaupallisesti julkinen versio. Se sisälsi suuren määrän bugien korjausta ja uusia ominaisuuksia. Cupcake mahdollisti kopioi ja liitä ominaisuuden, jota käytetään tänä päivänä paljon. (Android Developer 2009a.)

Android 1.6 (Donut) julkaistiin syyskuussa 2009 sisältäen useita uusia päivityksiä. Hakuja pystyi nyt tekemään sekä tekstillä että äänellä. Parannuksia tehtiin myös kameran toimintaan ja suosituskykyyn. (Android Developer 2009b.)

Android 2.0 (Eclair) julkaistiin marraskuussa 2009 tuoden uudistetun käyttöliittymän. Samalla esiteltiin myös parempi virtuaalinen näppäimistö nopeampaa kirjoittamista varten, mutta jos se osoittautui liian hitaaksi, puhetoiminto oli paras vaihtoehto. Samalla lisättiin Bluetooth 2.1, nopeampaa tiedonsiirtoa varten. (Android Developer 2009c.)

Android 2.2 (Froyo) julkistettiin toukokuussa 2010. Tämä päivitys paransi huomattavasti käyttöjärjestelmän nopeutta ja sovellukset käynnistyivät hetkessä. Uusia ominaisuuksia oli USB-jakaminen ja WiFi hotspot –toiminnallisuus. (Android Developer 2010a.)

Android 2.3 (Gingerbread) julkaistiin joulukuussa 2010 parantaen akun kestävyyttä ja samalla akun hallintatyökaluja parannettiin. Kamera sai myös parannuksen, koska nyt käyttäjä pystyi nopeasti vaihtamaan etukamerasta takakameraan ja toisinpäin. (Android Developer 2010b.)

Android 3.0 (Honeycomb) julkaistiin helmikuussa 2011. Tämä versio oli ensimmäinen nimenomaan tableteille suunniteltu Android. Tämä tarkoitti käyttöliittymän muutosta, sillä fyysisille nappuloille ei ollut tarvetta. Ohjelmakuvakkeista tuli enemmän silmiin pistävämpiä, sillä ohjelmoijilla oli enemmän tilaa käytettävänä. (Android Developer 2011a.)

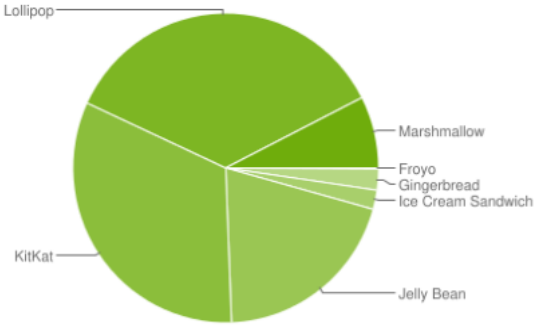
Android 4.0 (Ice Cream Sandwich) julkaistiin lokakuussa 2011. Päivitys toi mukanaan paljon ominaisuuksia ja toimintoja. Esimerkiksi näytön lukituksen voi avata kasvojentunnistuksella. Muita uusia ominaisuuksia olivat uusi Internet-selain ja kameran uudet toiminnot, mitkä mahdollistivat 1080p videokuvauksen. (Android Developer 2011b.)

Android 4.1 (Jelly Bean) julkaistiin kesäkuussa 2012. Tämä versio teki Androidista nopeamman, sulavamman sekä nopeammin reagoivan. Jelly Beanin tarkoituksena oli lähtökohtaisesti parantaa käyttöliittymän toiminnallisuutta ja suorituskykyä. (Android Developer 2012.)

Android 4.4 (KitKat) julkaistiin lokakuussa 2013. Version nimi piti alun perin olla nimeltään ”Key Lime Pie”, mutta se päätettiin vaihtaa, sillä uskottiin, ettei

kovinkaan moni tiedä, miltä limettipiirakka maistuu. Uusi nimitys oli näin ollen ”Kitkat”. Kitkatin tavoitteena oli ottaa huomioon vanhat Android käyttäjät. Kitkatista oli suunniteltu tarkoituksella kevyt, jotta se toimisi monilla erilaisilla matkapuhelimilla. KitKat on tällä hetkellä yleisimmin käytössä oleva versio, kuten kuvasta 2 voidaan huomata. (Android Developer 2013.)

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.0%
4.1.x	Jelly Bean	16	7.2%
4.2.x		17	10.0%
4.3		18	2.9%
4.4	KitKat	19	32.5%
5.0	Lollipop	21	16.2%
5.1		22	19.4%
6.0	Marshmallow	23	7.5%



Kuva 2. Android-versioiden jakauma (Android Developer 2016b).

Android 5.0 (Lollipop) julkaistiin lokakuussa 2014. Lollipopissa parannettiin Androidin suorituskykyä merkittävästi uuden Android Runtimen (ART) avulla, mitä rakennettiin alusta asti perustuen ahead-of-timeen (AOT) ja just-in-timeen (JIT). Lollipop myös uudisti tyylin miten esittää puhelimeen tulevat ilmoitukset. (Android Developer 2014.)

Android 6.0 (Marshmallow) julkaistiin lokakuussa 2015. Marshmallow’n päämääräinen tarkoitus oli kokonaisuudessaan parantaa käyttäjän kokemusta tarjoamalla laajemmat käyttöoikeudet. Marshmallow on tällä hetkellä uusin Android-versio. (Android Developer 2015.)

3 KEHITYSYMPÄRISTÖ

Tässä osiossa läpikäydään lyhyesti Android-alustalle ohjelmoinnin mahdollistavat työkalut, joita on käytetty tämän opinnäytteen tekemisessä.

3.1 JDK

Java Development Kit (JDK) on Oracle Corporationin kehittämä kehitysympäristö, joka sisältää Java-ohjelmoinnille vaadittavia kehitystyökaluja. Olennainen osa JDK:ssa on sen sisältämä Java Runtime Environment (JRE), joka mahdollisaa Javalla luotujen sovellusten ajamisen. JDK on ilmaisessa jakelussa ja sen voi ladata Oraclen sivuilta. (Techotopia 2016a.)

3.2 Android SDK

Android Software Development Kit (Android SDK) on joukko Android-alustalla tarvitsemia kehitystyökaluja sisältävä paketti. Paketti sisältää debuggerin, API-kirjastoja, virtuaaliympäristön, lähdekoodiesimerkkejä ja tutoriaaleja. Joka kerta kun uusi Android-versio julkaistaan, sitä vastaava SDK paketti myös luodaan. (Techotopia 2016b.)

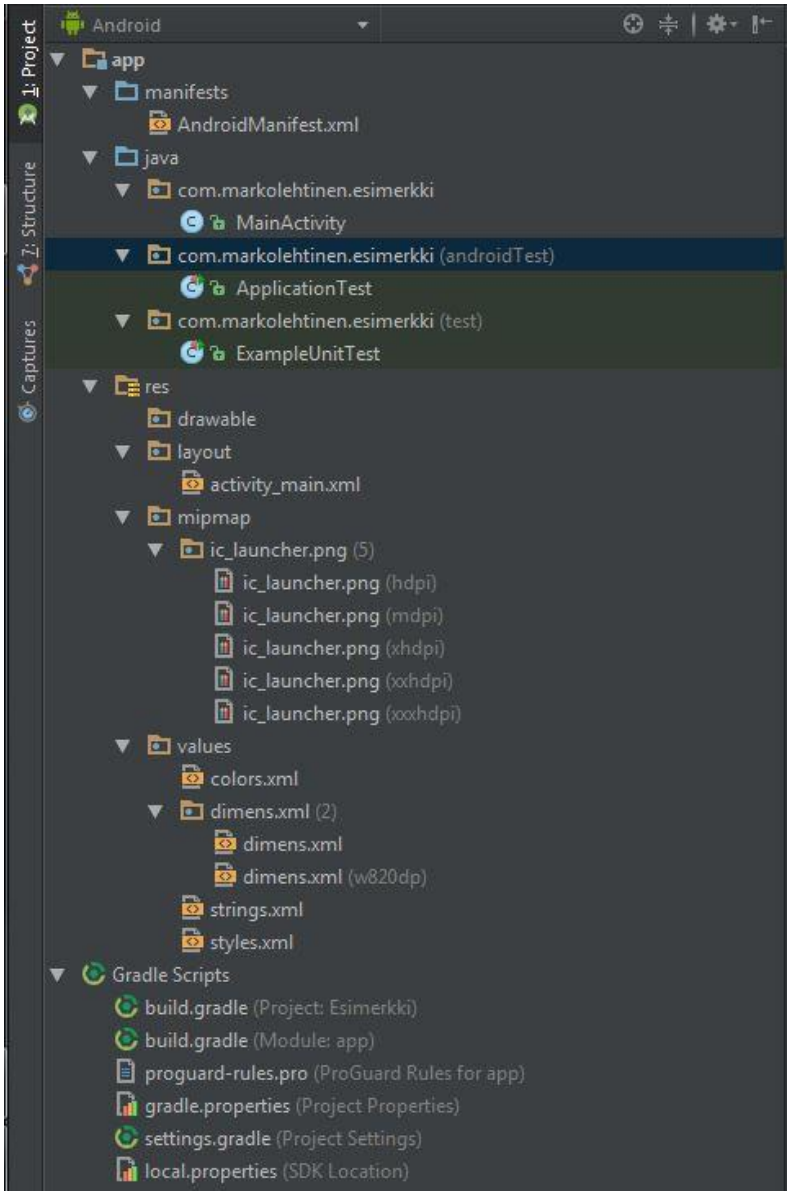
Android SDK:n sisältämät paketit pystyvät jo itsessään mahdollistamaan Android-ohjelmoinnin, mutta yleensä käytetään erillistä kehitysympäristöä. Tässä tapauksessa se on Android Studio.

3.3 Android Studio

Android Studio on Googlen kehittämä virallinen ohjelmointiympäristö Android-kehittäjille. Se tarjoaa kaikki tarpeelliset kehitystyökalut yhtenä pakettina ja mahdollistaa emulointimahdollisuudet eri laitteilla ja Android-versioilla. (Android Developer 2016a.)

3.3.1 Projektin rakenne

Kun Android Studiossa luodaan uusi projekti, se luo automaattisesti kaikki tarvittavat rakenteet projektia varten ja näyttää ne projekti-ikkunassa kuten kuvasta 3 voidaan havaita.



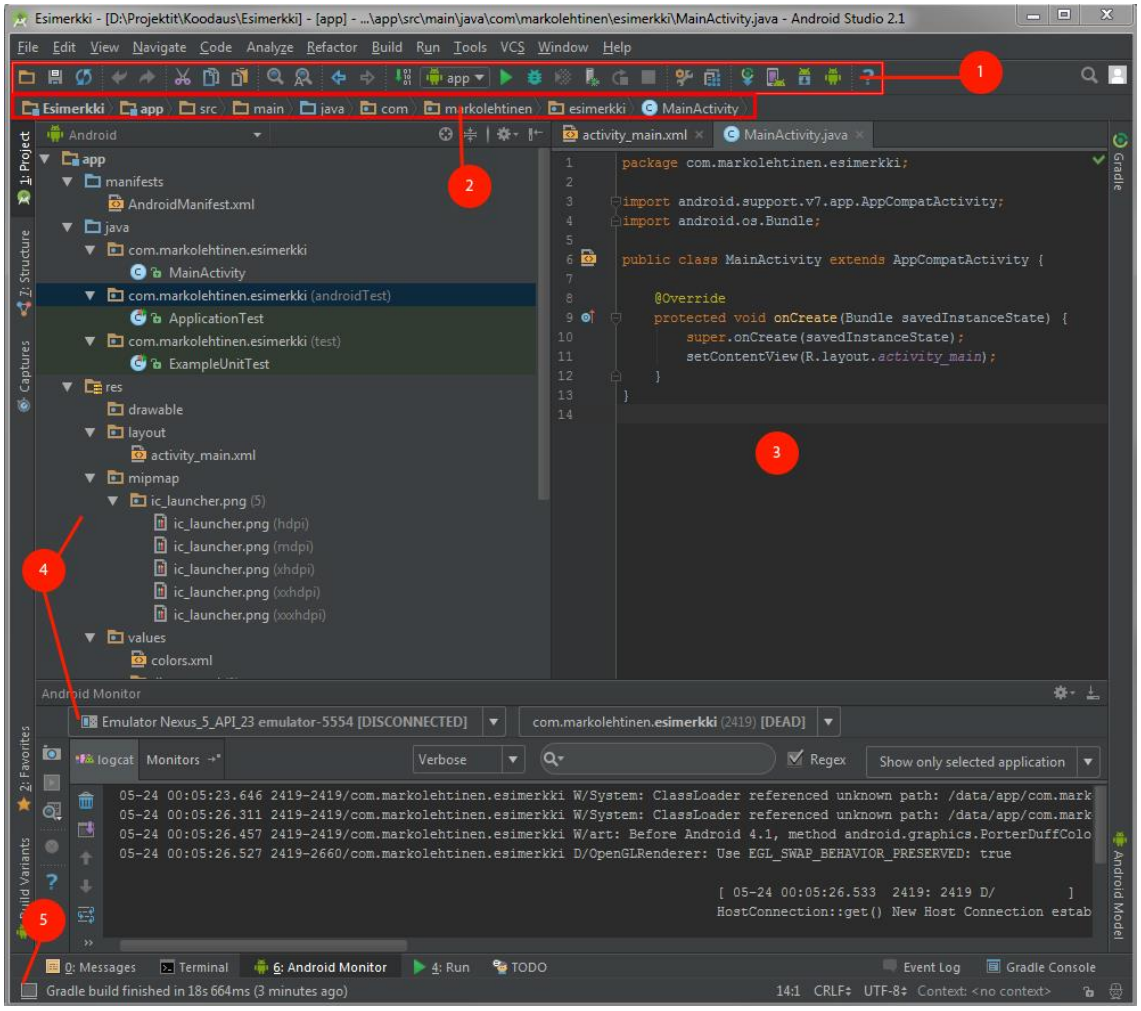
Kuva 3. Esimerkki kuva uuden projektin hakemistohierarkiasta.

Jokainen projekti koostuu kolmesta pääkansiosta: manifest, java ja res. Manifest-kansio sisältää AndroidManifest.xml tiedoston, joka toimii ikään kuin

projektin aivoina. Java-kansio sisältää nimensä mukaisesti kaikki java tiedostot ja res-kansio sisältää ulkoasuun liittyviä tiedostoja. (Android Developer 2016c.)

3.3.2 Käyttöliittymä

Android Studion käyttöliittymä koostuu muutamasta loogisesta osiosta, kuten kuvasta 4 voidaan havaita.



Kuva 4. Esimerkki kuva Android Studion käyttöliittymästä.

1. Työkaluriviltä voidaan suorittaa monia komentoja, kuten esimerkiksi sovelluksen käynnistys.

2. Navigointipalkki helpottaa hahmottamaan missä sijaintisi projektissa on.

3. Muokkausikkunassa luodaan ja muokataan koodia. Riippuen tiedostotyypistä tämä näkymä näkyy erilaisena. Esimerkiksi muokatessa XML-tiedostoa nähdään miltä se fyysisesti näyttää.

4. Työkaluikkuna antaa pääsyn tiettyihin tehtäviin, kuten projektin hallintaan ja versionhallintaan.

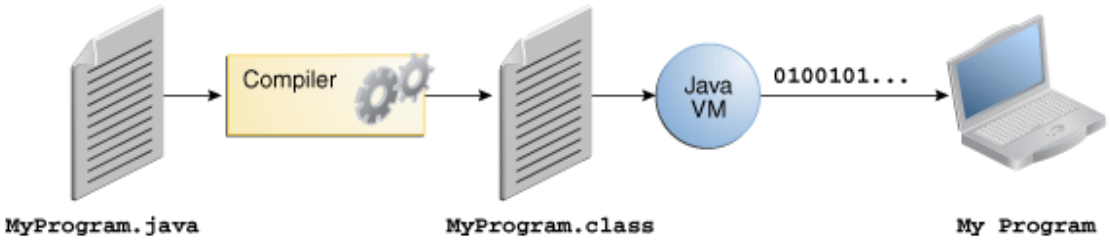
5. Status-palkki kertoo missä tilassa projekti on ja ilmoittaa varoituksista.

Käyttöliittymä on muokattavissa ja esimerkiksi eri osioiden kokoa voidaan helposti muuttaa. (Android Developer 2016c.)

3.4 Ohjelmointikieli Java

Java on Sun Microsystemsin kehittämä oliopohjainen ohjelmointikieli, joka on sittemmin siirtynyt Oracle Corporationin omistukseen. Javaa on kehitetty vuodesta 1991 asti ja se pohjautuu C/C++ -tyyliseen syntaksiin, jotta se olisi helppo omaksua. Alun perin Javasta oli tarkoitus tulla kevyt kieli, jota käytettäisiin erilaisissa pienelektroniikkalaitteissa, kuten kahvinkeittimessä ja televisioissa. Nykyään Javaa käytetään kaikkialla, kuten pelien ohjelmoinnissa ja Android-ohjelmoinnissa. (Javatpoint 2016, Tutorialspoint 2016.)

Java-kielessä kaikki lähdekoodi tallennetaan tavalliseen tekstitiedostoon .java päätteellä. Nämä tiedostot muunnetaan sen jälkeen .class tiedostoiksi javac kääntäjän avulla. Javan .class-tiedosto on tiedosto, joka sisältää tavukoodia, jota Javan virtuaalikone osaa tulkata ja pystyy suorittamaan Java-ohjelman virtuaalikoneessa. Javan virtuaalikoneen avulla sama Java-ohjelma voidaan suorittaa monella eri alustalla. Kuvasta 5 voidaan selkeämmin havaita prosessin eteneminen. (Oracle 2016.)



Kuva 5. Havaintokuva Java-ohjelman käynnistyksestä (Oracle 2016).

3.5 Merkintäkieli XML

XML (eXtensible Markup Language) on merkintäkieli, joka on suunniteltu järjestelmien väliseen tiedonsiirtoon. Merkintäkielen avulla voidaan suuria tietuekokonaisuuksia jäsentää selkeämmäksi ja siten nopeuttaa tietyn tietueen etsintää. XML tallettaa tiedot tavalliseen teksti-tiedostoon mikä mahdollistaa sovelluksesta ja alustasta riippumattoman tavan tallettaa, siirtää ja jakaa dataa. (W3schools 2016.)

4 SUUNNITTELU

Tässä osiossa käsitellään tehdyn sovellusprototyypin suunnittelua. Päämääränä oli kehittää frisbeegolf-sovellukseen tietokanta. Samalla mietitään käyttöliittymän tekoa. Opinnäytetyön tavoitteiden lisäksi luvussa käydään läpi sovelluksen ideaa sekä sen tärkeimmät ominaisuudet ja toiminnat.

4.1 Tavoitteet, kuvaus ja idea

Työn päällimmäisenä tavoitteena oli saada opinnäytetyön tekijä tutustumaan syvemmin Android-ohjelmointiin. Lisäksi sovellukseen luodaan tietokanta tiedontallennusta varten. Tietokannan lisäksi opinnäyttetyön tarkoituksena oli luoda käyttöliittymä.

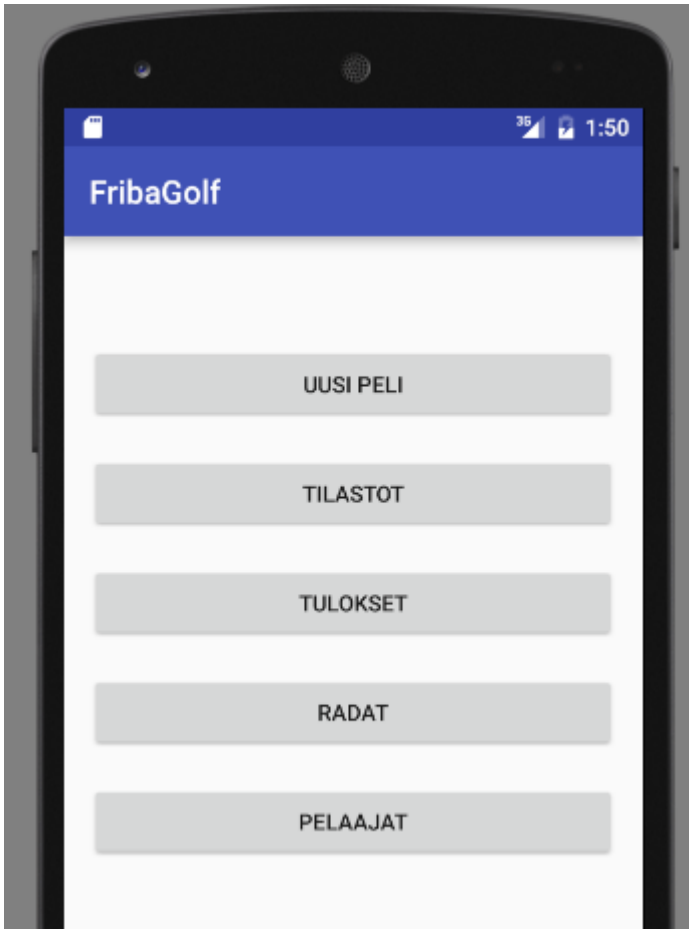
Sovelluksella on tarkoitus pitää kirjaa frisbeegolfista saaduista tuloksista. Se toimii ikään kuin tietokantaohjelmana, joka tallentaa tiedot puhelimen muistiin. Sovellus on alun perin suunniteltu yhden pelaajan käyttöön, joten vain yhden pelaajan tulokset on huomioitu.

Sovellukseen laaditaan päävalikon ohella viisi muuta aktiviteettia. Näitä aktiviteetteja varten päävalikkoon luodaan kullekin oma painonappi, jolla voi siirtyä aktiviteettien välillä.

4.2 Toiminta ja ominaisuudet

Jotta käyttäjä voi käyttää sovellusta, hänellä tulee olla Android-laite. Käyttäjä voi olla kuka tahansa. Projektin alussa määritettiin alimmaksi Android-versioksi 4.1, jolla sovelluksen tulee olla vähintään yhteensopiva. Android-versio 4.1 valikoitiin, koska kyseinen versio mahdollistaa paremman käyttöliittymän teon, sekä kattaa noin 95 % Android-käyttäjistä.

Kun sovellus käynnistetään, ensimmäiseksi avautuu päävalikko (kuva 6). Päävalikko koostuu viidestä painikkeesta, joita painamalla avautuu toinen näkymä.



Kuva 6. Päävalikko.

Uusi peli vie näkymään, jossa luodaan uusi peli valitsemalla pelaaja ja pelattava rata. Tilastot-näkymässä voidaan tarkastaa, kuinka tietty pelaaja on menestynyt kullakin radalla. Tulokset-näkymässä voidaan tarkastaa viimeaikaisten pelien menestystä. Radat-näkymässä voidaan tarkastaa nykyisiä ratoja ja luoda uusia ratoja. Pelaajat-näkymässä voidaan tarkastaa nykyisiä pelaajia ja luoda uusia pelaajia.

5 TOTEUTUS

Tässä luvussa käydään läpi tehdyn sovelluksen toteutusta. Sovelluksen elementtien kuvauksen lisäksi pohditaan, mitä lisäominaisuuksia voidaan kehittää.

5.1 Tiedon tallennus

Android käyttää tiedostojärjestelmää, joka muistuttaa hyvin paljon levypohjaista tiedostojärjestelmää, jota muut alustat käyttävät. Kaikilla Android-laitteilla on kaksi tiedontallennusaluetta, sisäinen ja ulkoinen muisti. Vaikka nykyään ei jokaisessa laitteessa olekaan mahdollista olla ulkoista irrotettavaa muistia, kuten muistikorttia, laiteen muisti on silti jaettu näihin kahteen alueeseen. (Android Developer 2016e.)

5.1.1 Sisäinen muisti

Sisäinen muisti on aina saatavilla. Sisäiseen muistiin tallennetut tiedostot ovat oletuksena käytössä vain sen sovelluksen kautta, joka on ne tallentanut. Kun käyttäjä poistaa sovelluksen, samalla poistuu kaikki sovellukseen liittyvät tiedostot. Sisäistä muistia on hyvä käyttää silloin, kun ei haluta muiden sovelluksien pääsevän käsiksi tiedostoihin. (Android Developer 2016e.)

5.1.2 Ulkoinen muisti

Ulkoinen muisti ei ole aina saatavilla, koska käyttäjä voi poistaa esimerkiksi muistikortin laitteesta milloin haluaa. Ulkoiseen muistiin on hyvä tallentaa tiedostoja, joita halutaan jakaa muiden sovellusten kanssa. Kun käyttäjä poistaa sovelluksen, tiedostot ulkoisesta muistista poistuvat vain, jos ne ovat tallennettuna tiettyyn sijaintiin. (Android Developer 2016e.)

5.1.3 SQL-tietokannat

Tietojen tallennus tietokantaan on silloin ideaalia, kun tallennettava tieto on toistuvaa tai jäsenneltyä. Android tarjoaa täyden tuen SQLite-tietokantoihin. Esimerkiksi jos tarkoitus on tallentaa monen henkilön yhteystiedot, kannattaa silloin käyttää SQLite-tietokantaa. (Android Developer 2016d, Android Developer 2016g.)

5.1.4 SQLite

SQLite on relaatiotietokantajärjestelmä, jota käytetään tietojen tallennukseen. SQLite on kevyt tietokantamoottori, joka vie hyvin vähän tilaa ja on siksi hyvä valinta käyttää esimerkiksi kännyköissä. (SQLite 2016a.)

SQLite on kompakti kirjasto. Vaikka siinä olisi kaikki ominaisuudet valittuna sen koko olisi silti alle 500 kilotavua. SQLite on hyvin itsenäinen ja tarvitsee hyvin vähän tukea ulkoisista kirjastoista tai käyttöjärjestelmältä. (SQLite 2016a, SQLite 2016c.)

SQLiten käyttöönotto on helppoa, sillä sitä ei tarvitse ollenkaan asentaa. SQLite lukee ja kirjoittaa suoraan levyllä sijaitseviin tiedostoihin. SQLite-järjestelmä linkitetään suoraan sitä käyttävään sovellukseen, joten erillistä tietokantapalvelinta ei tarvita. (SQLite 2016b, SQLite 2016d.)

5.2 Tietokannan toteutus

5.2.1 Tietokantaluokan ja aliluokan luominen

Suositeltu tapa SQLite-tietokannan luomiseen on luoda SQLiteOpenHelper-luokan aliluokka. Ennen tätä tulee Android Studiossa luoda ensin täysin uusi luokka. Tämä onnistuu valitsemalla valikkopalkista File -> New -> Java Class,

jolloin uuden luokan luonti-ikkuna avautuu. Nimeksi annetaan DatabaseHelper ja uusi luokka on luotu, kuten kuvasta 7 voidaan huomata.

```
public class DatabaseHelper {

}
```

Kuva 7. Uusi tietokantaluokka.

Seuraavaksi luokkaa täytyy muokata, jotta se saa SQLiteOpenHelperin ominaisuudet. Se onnistuu lisäämällä luokkaan extend SQLiteOpenHelper, jolloin se on muodossa public class DatabaseHelper extends SQLiteOpenHelper { }. (Android Developer 2016f, Android Developer 2016g.)

5.2.2 Tietokannan määrittely

Sovelluksen tietokannan nimeksi tulee fribagolf.db ja tietojen tallennusta varten tullaan tarvitsemaan neljää taulua. Taulut ovat pelaajien tietojen, paikkatietojen, ratojen tulosten ja pelikertojen tallentamiseen.

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "fribagolf.db";

    public static final String TABLE_PELAAJA = "pelaaja_taulu";
    public static final String COLUMN_PELAAJA_ID = "pelaaja_id";
    public static final String COLUMN_PELAAJA_NIMI = "pelaaja_nimi";

    public static final String TABLE_PAIKKA = "paikka_taulu";
    public static final String COLUMN_PAIKKA_ID = "paikka_id";
    public static final String COLUMN_PAIKKA_NIMI = "paikka_nimi";

    public static final String TABLE_VAYLA = "vayla_taulu";
    public static final String COLUMN_VAYLA_ID = "vayla_id";
    public static final String COLUMN_VAYLA_TULOS_1 = "vayla_tulos_1";
    public static final String COLUMN_VAYLA_TULOS_2 = "vayla_tulos_2";
    |
    |
    public static final String COLUMN_VAYLA_TULOS_17 = "vayla_tulos_17";
    public static final String COLUMN_VAYLA_TULOS_18 = "vayla_tulos_18";

    public static final String TABLE_PELIKERTA = "pelikerta_taulu";
    public static final String COLUMN_PELIKERTA_ID = "pelikerta_id";
    public static final String COLUMN_PELIKERTA_PAIKKA_ID = "pelikerta_paikka_id";
    public static final String COLUMN_PELIKERTA_PELAAJA_ID = "pelikerta_pelaaja_id";
    public static final String COLUMN_PELIKERTA_VAYLA_ID = "pelikerta_vayla_id";
    public static final String COLUMN_PELIKERTA_RATA_LKM = "pelikerta_rata_lkm";
    public static final String COLUMN_PELIKERTA_PVM = "pelikerta_pvm";
```

Kuva 8. String-muuttujat taulukoille.

Kuvassa 8 on määritelty taulukoille String-muuttujat, joita tullaan myöhemmin käyttämään onCreate- ja onUpgrade-metodeilla.

5.3 Tietokannan luominen ja päivittäminen

OnCreate-metodissa luodaan tietokanta ja taulut. Taulujen luominen tapahtuu execSQL-metodin avulla (kuva 9).

```
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + TABLE_PELAAJA +
            " (PELAAJA_ID INTEGER PRIMARY KEY AUTOINCREMENT," +
            "PELAAJA_NIMI TEXT NOT NULL)");
    db.execSQL("create table " + TABLE_PAIKKA +
            " (PAIKKA_ID INTEGER PRIMARY KEY AUTOINCREMENT," +
            "PAIKKA_NIMI TEXT NOT NULL)");
    db.execSQL("create table " + TABLE_VAYLA +
            " (VAYLA_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "VAYLA_TULOS_1 TEXT, " +
            "VAYLA_TULOS_2 TEXT, " +
            "VAYLA_TULOS_17 TEXT, " +
            "VAYLA_TULOS_18 TEXT)");
    db.execSQL("create table " + 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, P" +
            "ELIKERTA_RATA_LKM INTEGER, " +
            "PELIKERTA_PVM TEXT)");
}
```

Kuva 9. Taulukoiden luominen onCreate-metodissa.

OnUpgrade-metodia kutsutaan, kun tietokantaa pitää päivittää (kuva 10).

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PELAAJA);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PAIKKA);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_VAYLA);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PELIKERTA);
    onCreate(db);
}
```

Kuva 10. Taulujen poistaminen onUpgrade-metodissa.

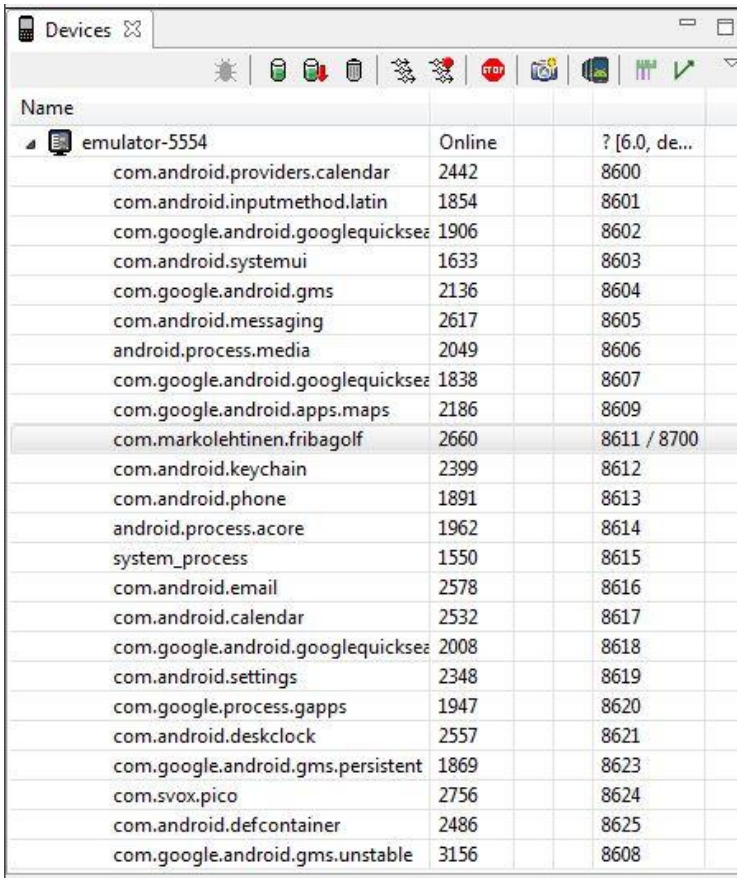
ExecSQL-metodia käytetään poistamaan taulut tietokannasta DROP TABLE IF EXISTS SQL-komennon avulla. Kun taulut on poistettu kutsutaan onCreate-metodia luomaan päivitetyt taulut uudestaan. (Android Developer 2016f.)

5.4 Luodun tietokannan tarkastus

Helpoin tapa tarkastella SQLite-tietokantoja on Android-emulaattorin kautta. Emulaattori on virtuaalinen ympäristö, joka pyrkii toimimaan mahdollisimman samalla tavalla kuin alkuperäinen laite. Jotta voidaan tarkastaa, onko tietokanta luotu, tulee käynnistää sovellus. Tämän jälkeen jätetään sovellus käyntiin ja siirrytään Android Device Monitor -näkymään (kuva 11).

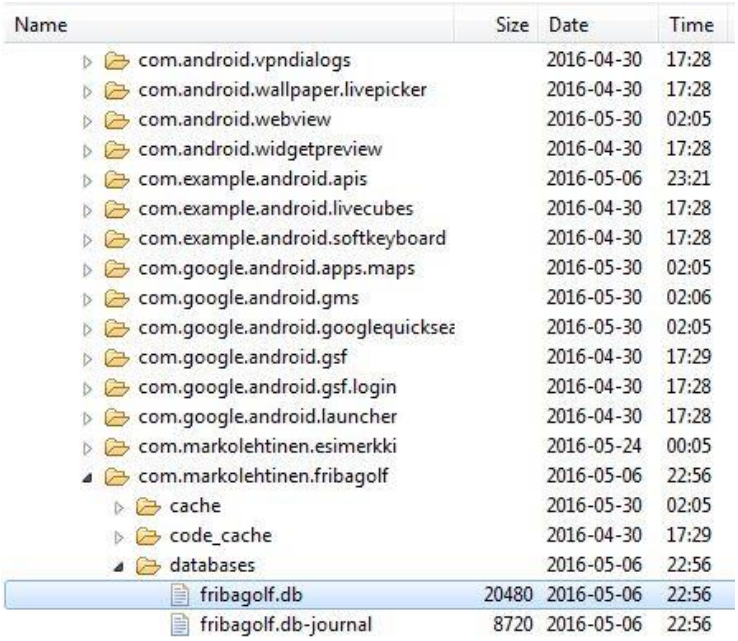
Kuva 11. Nappi Android Device Monitor -näkymään.

Android Device Monitor näkymässä nähdään vasemmalla puolella joukko käytössä olevista laitteista. Laitteista valitaan emulaattori. Kun emulaattori on valittu, aukeaa kuvassa 12 näkyvä lista pakettinimistä ja valitaan sovelluksen pakettinimi, joka tässä tapauksessa on com.markolehtinen.fribagolf.



Kuva 12. Pakettinimen valinta.

Kun pakettinimi on valittu, siirrytään näkymän oikealle puolella ja valitaan File Explorer, joka näyttää paketin sisällön. Kahden data-kansion alta löytyy sovelluksen pakettinimi, joka sisältää databases-kansion, mistä löytyy tietokantatiedosto fribagolf.db (kuva 13).



Kuva 13. Tietokantatiedoston sijainti.

Ohjelmalla nähtiin, että tietokanta on tallentunut oikein, joten tästä on hyvä jatkaa.

5.5 Jatkokehitys

Jatkossa tultaisiin lisäämään tiedontallennus funktiot, jotta tietokantaan voidaan lisätä tietoja. Samalla kehitettäisiin käyttöliittymää, jotta tämä olisi mahdollista. Tärkeää on myös keskittyä sovelluksen ulkoasuun ja tehdä siitä kutsuvan näköinen. Myös Google Maps -sovelluksen implementointia kannattaa harkita yhdessä GPS:n kanssa, parempian ratakarttojen saamiseksi ja samalla nähtäisiin millainen radan väylä on.

6 LOPUKSI

Opinnäytteen tekeminen oli hyvin rankkaa aikaa, mutta se oli silti opettavaista. En ole koskaan pitänyt itseäni minkäänlaisena koodarina ja olen ymmärtänyt vain perusteita. Siksi halusinkin haastaa itseni tämän tekemiseen. Opin myös priorisoimaan asioita opinnäytteen aikana ja luovuin joistakin asioista, jotta se valmistuisi nopeammin. Myös tekoprosessia oli mietittävä, jotta oli helpompaa havainnoida mihin aika riittäisi ja mihin ei. Koin oppivani opinnäytteen aikana projektinhallintaa ja itsenäistä työskentelyä, mikä on hyvä asia tulevaisuuden kannalta.

Opinnäytteessä tarkasteltiin mobiiliohjelmointia Androidilla ja käsiteltiin Androidin historiaa. Lisäksi käsiteltiin, mikä on SQLite ja luotiin sillä tietokanta sovellukseen. Lopputuloksena saatiin tavoitteita vastaava valmis prototyyppi.

Mikäli sovellusta vielä kehitetään, lisättäisiin tiedontallentamisen mahdollistavat funtiot ja lisäykset käyttöliittymään. Hyvä lisäys olisi myös Google Maps –sovelluksen yhteistyö GPS:n kanssa, jotta saataisiin paremmat ratakartat.

LÄHTEET

Android Developer. 2009a. Android 1.5 Platform Highlights. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-1.5-highlights.html

Android Developer. 2009b. Android 1.6 Platform Highlights. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-1.6-highlights.html

Android Developer. 2009c. Android 2.0 Platform Highlights. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-2.0-highlights.html

Android Developer. 2010a. Android 2.2 Platform Highlights. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-2.2-highlights.html

Android Developer. 2010b. Gingerbread. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-2.3-highlights.html

Android Developer. 2011a. Honeycomb. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-3.0-highlights.html

Android Developer. 2011b. Ice Cream Sandwich. Viitattu 16.12.2015.
https://developer.android.com/about/versions/android-4.0-highlights.html

Android Developer. 2012. Jelly Bean. Viitattu 16.12.2015.
http://developer.android.com/about/versions/jelly-bean.html

Android Developer. 2013. Android KitKat. Viitattu 16.12.2015.
http://developer.android.com/about/versions/kitkat.html

Android Developer. 2014. Android Lollipop. Viitattu 16.12.2015.
http://developer.android.com/about/versions/lollipop.html

Android Developer. 2015. Android 6.0 Changes. Viitattu 16.12.2015.
https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html

Android Developer. 2016a. Android Studio. Viitattu 25.5.2016.
https://developer.android.com/studio/index.html

Android Developer. 2016b. Dashboards. Viitattu 16.12.2015.
http://developer.android.com/about/dashboards/index.html

Android Developer. 2016c. Meet Android Studio. Viitattu 25.5.2016.
https://developer.android.com/studio/intro/index.html

Android Developer. 2016d. Saving Data in SQL Databases. Viitattu 29.5.2016.
https://developer.android.com/training/basics/data-storage/databases.html

Android Developer. 2016e. Saving Files. Viitattu 29.5.2016.
https://developer.android.com/training/basics/data-storage/files.html

Android Developer. 2016f. SQLiteOpenHelper. Viitattu 29.5.2016.
https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html

Android Developer. 2016g. Storage Options. Viitattu. 29.5.2016.
https://developer.android.com/guide/topics/data/data-storage.html

Android Suomi. 2013. Mikä on Android?. Viitattu 16.12.2015. http://blog.androidsuomi.fi/mika-on-android/

Chowdhury, R. 2012. A Look Into: Android Evolution [Cupcake – Jelly Bean]. Viitattu 16.12.2015. http://www.hongkiat.com/blog/android-evolution/

eLinux. 2011. Android Architecture. Viitattu 16.12.2015. http://elinux.org/Android_Architecture

Javatpoint. 2016. History of Java. Viitattu 25.5.2016. http://www.javatpoint.com/history-of-java

Open Handset Alliance. 2007. FAQ. Viitattu 16.12.2015. http://www.openhandsetalliance.com/oha_faq.html

Oracle. 2016. About the Java Technology. Viitattu 25.5.2016. https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

Sharma, N. 2013. The Beginner's Guide to Android: Android Architecture. Viitattu 16.12.2015. http://www.edureka.co/blog/beginners-guide-android-architecture

SQLite. 2016a. About SQLite. Viitattu 29.5.2016. http://www.sqlite.org/about.html

SQLite. 2016b. SQLite Is A Zero-Configuration Database. Viitattu 29.5.2016. http://www.sqlite.org/zeroconf.html

SQLite. 2016c. SQLite Is Self-Contained. Viitattu 29.5.2016. http://www.sqlite.org/selfcontained.html

SQLite. 2016d. SQLite Is Serverless. Viitattu 29.5.2016. http://www.sqlite.org/serverless.html

Techotopia. 2014. An Overview of the Android Architecture. Viitattu 16.12.2015. http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

Techotopia. 2016a. Android SDK. Viitattu 23.5.2016. https://www.techopedia.com/definition/4220/android-sdk

Techotopia. 2016b. Java Development Kit (JDK). Viitattu 23.5.2016. https://www.techopedia.com/definition/5594/java-development-kit-jdk

Tutorialspoint. 2016. Java – Overview. Viitattu 25.5.2016. http://www.tutorialspoint.com/java/java_overview.htm

W3schools. 2016. Introduction to XML. Viitattu 25.5.2016. http://www.w3schools.com/xml/xml_whatis.asp