Android-sovelluskehityksen perusteet Android Studiolla

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Syksy 2017

Matti Väkiparta

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Visamäki
Tietojenkäsittely

Tekijä Matti Väkiparta **Vuosi** 2017

Työn nimi Android Sovelluskehityksen perusteet Android Studiolla

TIIVISTELMÄ

Opinnäytetyön tarkoituksena on tutustua Android Studioon ja Android-sovelluskehitykseen sekä Java-koodikieleen. Työn tavoitteena on toteuttaa yksinkertaisia toimintoja Android Studiolla ja oppia Android-sovelluskehityksen peruselementtejä. Mobiilisovelluskehitys on kuulostanut mielenkiintoiselta eikä siitä ole aikaisempia kokemuksia, joten tämä työ oli oiva tilaisuus opiskella sitä.

Työssä käsitellään Android Studion asennusta, sisältöä sekä sovelluksella ohjelmointia. Ohjelmoinnissa tutkitaan Android Studion ominaisuuksia ja sovelletaan niitä. Java-koodikieleen perehdytään muiden koodikielien tukemana.

Googlen omilta sivuilta löytyvät hyvät infopaketit, jotka toimivat teorian pohjana. Sovelluksen tekoon löytyy apua Googlen kehitysmateriaalisivustoilta. Muutenkin, jos törmäsi ongelmiin tai uuteen asiaan, internetistä löytyi helposti apua sovellettuihin tilanteisiin.

Tulokseksi saatiin useita toimivia toimintoja ja elementtejä, mutta sovellukseen ei ole graafisesti mitenkään panostettu. Sovellukselle löytyy useita kehitysmahdollisuuksia ja sen jatkokehitys olisi erittäin hyvää lisäopiskelua Android-sovelluskehityksestä.

Avainsanat Sovelluskehitys, koodaaminen, Android, Android Studio

Sivut 26 s.

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme in Business Information Technology

Author Matti Väkiparta **Year** 2017

Subject of Bachelor's thesis Basics of Android software development

ABSTRACT

The purpose of this thesis is to study Android software development, Android Studio as a platform and Java as a coding language. Goals are to learn Android software development basics and to accomplish simple and working functions in an Android Software. Mobile software development is interesting and the author had no earlier experiences from it. This thesis was a perfect chance to study it.

The thesis includes some key theory about Android and Android Studio installation and content. Android Studio content is studied by programming with Java. Studying Java is a minor part of the thesis and other coding languages help in that.

Google offers useful information about Android which is the base of the theory. Google also has a development website for Android Studio and the website helps with the programming. If any problems occurred Googles sites or searching online helped a lot.

A few functions and a simple working software were the results of this thesis. The software isn't visually great, but it was easy to study the basics with it. There are many further development options for the software and developing it further would be a good study about Android software development.

Keywords Software development, coding, Android, Android Studio

Pages 26 p.

Sisällys

1 JOHDANTO

Tässä opinnäytetyössä tutustutaan Androidiin ja sen sovelluskehitykseen. Ensiksi perehdytään Androidin rakenteeseen eli siihen mistä Android-käyttöjärjestelmät koostuvat. Androidiin tutustutaan myös yleisellä tasolla esimerkiksi sen historiaan ja nykytilanteeseen.

Työn lähtökohtana ovat muiden koodikielten perustiedot. Opinnäytetyön aihe Androidin opettelusta on täydellinen, sillä mobiiliohjelmointia tai Java-ohjelmointia ei tule ollenkaan ammattikorkeakoulun kautta omien valintojeni johdosta ja aiempaa kokemusta mobiiliohjelmoinnista tai siihen käytettävistä ohjelmistoista ei ole.

Käytäntöä toteutetaan Android Studion avulla. Sen asennukseen ja sisältöön tutustutaan käymällä perusasiat läpi. Android Studion avulla toteutetaan yksinkertaisia sovelluskehityksen peruselementtejä ja niiden määrityksiä projektin muodossa. Opinnäytetyön tarkoitus on opetella Android-sovelluskehityksen tärkeimpiä osia, Android Studiota sekä Java-koodikieltä.

Android Studio ei ole täysin koodiin perustuva kehitysalusta. Koodin logiikan tietämyksellä pääsee pitkälle, ja kun kyseessä ovat yksinkertaiset elementit, vaikeita koodin osia ei tule eteen.

2 ANDROID

Android on Googlen omistama ja kehittämä mobiilikäyttöjärjestelmä, joka toimii Linux Kernelin päällä. Se on suunniteltu pääasiassa älylaitteille ja on tämän hetken suosituin mobiilikäyttöjärjestelmä.

Kuvassa 1. nähdään Androidin arkkitehtuuri. Seuraavissa kappaleissa on lyhyesti selitettynä, mitä jokainen arkkitehtuurin osa tekee ja mitä varten se on.

Pohjalla oleva Linux Kernel tarjoaa laitekehittäjilleen tärkeitä turvallisuusominaisuuksia esimerkiksi käyttäjäpohjaisen oikeusmallin ja prosessien erittelyn. Linux Kernel tunnetaan hyvin, joten se toimii erinomaisena pohjana laitekehittäjille. Osa Androidin tärkeistä ominaisuuksista vaatii Linux Kernelin toimiakseen.

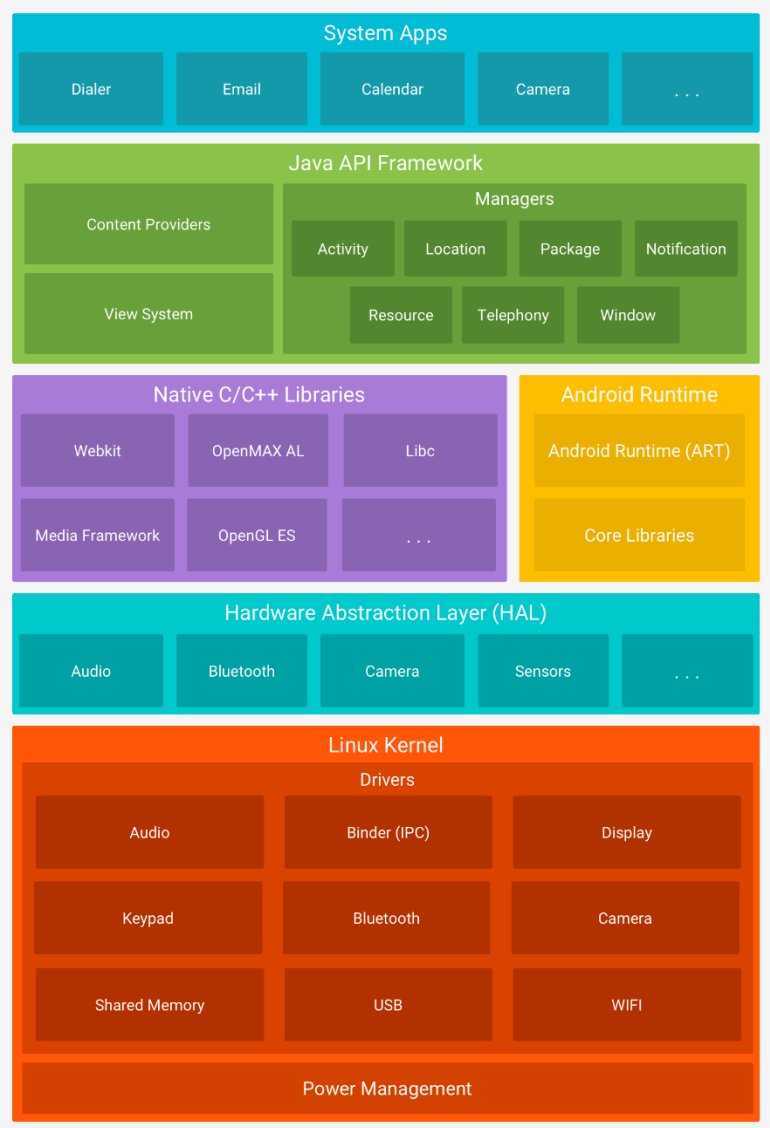
Hardware Abstraction Layer (HAL) tarjoaa käyttöjärjestelmälle kirjastot, joita laitteen komponentit tarvitsevat toimiakseen. Kun käyttäjä haluaa käyttää jotakin komponenttia, järjestelmä aukaisee kyseisen komponentin kirjaston.

Android Runtime on suunniteltu useiden prosessien toimimiseen samaan aikaan. Se on tarkoitettu Android 5.0:lle ja uudemmille versioille. Android Runtime paransi huomattavasti Android-käyttöjärjestelmää lisäämällä prosessien hallintaa.

Native C/C++ Libraries ovat kirjastoja, jotka tukevat Androidin perustoimintoja. Esimerkiksi HAL ja Android Runtime vaativat nämä kirjastot toimiakseen. Kyseiset kirjastot ovat myös saatavilla C/C++-sovelluskehitystä varten.

Java API Framework on ohjelmointirajapinta (API eli Application programming interface), joka sisältää Android-käyttöjärjestelmän ominaisuuksia ja toiminnallisuuksia Javalla koodattuna. Ohjelmointirajapintojen avulla pystytään yksinkertaistamaan sovellusten luontia jo valmiiksi olevien tietojen perusteella. Eri versioiden API:t ovat myös saatavilla netistä sovelluskehittäjien käyttöön.

System Apps ovat järjestelmän oletussovelluksia, jotka vastaavat Androidilta vaadittuja toimintoja sekä laitteessa olevia komponentteja. Nämä oletukset eivät ole lukittuja, vaan käyttäjä pystyy vaihtamaan niitä haluamiinsa sovelluksiin sekä poistamaan oletussovelluksia käytöstä. Peruskäyttäjille Androidin arkkitehtuuri on täysi mysteeri, eivätkä he näe näitä sovelluksia pidemmälle.



Kuva 1. Android arkkitehtuuri.

Android ei ole Googlen luoma, vaan Google osti sen vuonna 2005 Android Inc:ltä osaksi Googlen brändiä. Android on parantunut käyttöjärjestelmänä useiden päivityksien avulla ja kehitys jatkuu edelleen.

Android on avoimen lähdekoodin käyttöjärjestelmä. Tämä tarkoittaa sitä, että Google jatkaa sen kehitystä, mutta antaa myös toisten kehittää käyttöjärjestelmää eteenpäin vapaasti ja tarjoaa sitä internetin kautta ilmaiseksi. Google antaa siis mahdollisuuden laitteiden valmistajille kustomoida käyttöjärjestelmää halujensa mukaan ja tehdä laitteista yksilöllisiä. Jopa Androidin kilpailijat voivat käyttää lähdekoodia itselleen: ”Kilpailijat voivat ladata, asentaa, muuttaa ja jakaa lähdekoodia ilmaiseksi. Tuloksena useammat ihmiset saavat käsiinsä mobiiliteknologiaa kuin ikinä aiemmin.” (Google 2017) Useimmat laitteet tulevat Androidin vakio-ominaisuuksien kanssa, mutta sisältävät laitteen valmistajan omia ominaisuuksia, jotka eivät ole avointa lähdekoodia esim. Samsung ja LG. Laitteiden valmistajien on helpompi valita valmis ja toimiva avoimen lähdekoodin järjestelmä kuin kehittää itselle oma järjestelmä. Android-versioita on ollut useita ja uusia julkaistaan jatkuvasti. Uusin versio tällä hetkellä on Android 7 Nougat.

Android-laitteita on olemassa yli 24000. Androidille on saatavilla yli miljoona sovellusta Google Play -kaupasta. Sieltä löytyy paljon ilmaisia sovelluksia, jotka lisäävät Androidin suosiota. Useat sovellukset toimivat käyttöjärjestelmän tavoin avoimen lähdekoodin menetelmällä.

Androidin suosio ja kilpailukykyisyys johtuvat suurimmaksi osaksi hinnasta. Avoimen lähdekoodin käyttöjärjestelmät laitteissa halventavat kustannuksia ohjelmistopuolelta, jolloin peruslaitteen esim. keskivertopuhelimen hinta putoaa lähes pelkkiin valmistuskuluihin. Jos verrataan IOS-(Applen mobiilikäyttöjärjestelmä) ja Android-laitetta, huomataan selvä hintaero suljetun ja avoimen lähdekoodin välillä. Suosiota lisää myös se, että Android on paljon vapaampi käyttöjärjestelmä. Google myös mainostaa sitä sellaisena. (Google 2017)

Android on käytännöllinen ja muokattavissa helposti oman näköiseksi. Aloitusnäyttöjen sisältöjä ja ilmettä voi muokata oman mielen ja maun mukaan. Suosituimpia Android-merkkejä ovat Samsung, Sony ja Huawei. Oletusohjelmien kuten nettiselaimen vaihtaminen onnistuu. Google Play -kaupassa on Applea laajempi valikoima sovelluksia, mutta laadunvalvonta on myös jonkin verran heikompaa, joten perusvalppaus on tarpeen. (Telefinland 2016)

3 ANDROID STUDIO

Sovelluskehityksen aloitus tapahtuu hankkimalla vaaditut ohjelmat. Tässä työssä käytetään Android Studiota, jonka saa ladattua suoraan Googlelta. Android Studion versio on 141.2456560 ja sen jälkeiset päivitykset. Aiemmatkin versiot olisivat käyneet tähän työhön, koska uusimpia ominaisuuksia ei tule käytettyä. Uusin versio on kuitenkin Googlen suosittelema; se on mitä luultavimmin vakain versio sovelluksesta, jos ei käytetä beta-versioita.

Android Studio on Googlen julkaisema virallinen integroitu kehitysalusta (Integrated development enviroment) Android -sovelluskehitystä varten. Se tarjoaa monipuoliset työkalut Android-pohjaiselle sovelluskehitykselle. Ohjelmointikielenä toimii Java. Se käyttää Android SDK:ta (Software Development kit), joka sisältää kaikki Android-sovelluksen luontiin tarvittavat kirjastot. (Google 2017)

Android Studion on Googlen suosittelema ja tukema. Lisäksi Googlelta löytyy hyvät materiaalit sovelluskehityksen opiskeluun ja Android Studion käyttöön osana sovelluksen elämänkaarta. Näitä materiaaleja on helppo käyttää hyödyksi perusasioiden opiskelussa ja ne toimivat pohjana tässä opinnäytetyössä.

Android Studiosta löytyvät internetarvostelut luovat positiivista kuvaa niille, jotka eivät ole ennemmin käyttäneet Android Studiota. Useissa arvosteluissa tosin sanotaan, että vieläkin löytyy jatkokehitystarpeita. Opinnäytetyön aiheena on kuitenkin sovelluskehityksen perusteet, joten Android Studion mahdolliset pienet virheet tai kehitystarpeet eivät vaikuta tähän työhön.

3.1 Android Studion käytön esivaatimukset ja asennus

Android Studio -ohjelmalla on käyttöjärjestelmävaatimuksia, joiden tulee täyttyä ohjelman sujuvan toimivuuden takaamiseksi. Vaatimukset eri käyttöjärjestelmien versioille löytyvät Android Studion lataussivustolta. Tässä työssä käyttöjärjestelmänä on Windows 10, joten vaatimukset ovat seuraavat: vähintään 2 Gigatavua(GB) kovalevytilaa ja 4 GB suositeltu, vähintään 3 GB keskusmuistia ja 8 GB suositeltu, minimiresoluutio 1280 x 800 ja Java Development Kit (JDK). (Google 2017)

Käyttöjärjestelmän vaatimuksena oleva Java Development Kit (JDK) tarkoittaa erillistä Javan kehitystyökalua. Se tulee hankkia valmiiksi ennen Android Studion asennusta tietokoneelle. JDK:n saa hankittua helposti verkosta, mutta siitä on eri versioita. Kannattaa käyttää JDK:n uusinta versiota, sillä se tukee kaikkia uusimpia ominaisuuksia. Ennen Android Studion asennusta asennetaan ladattu Java Development Kit. Tässä työssä on käytetty kahta eri JDK:n versiota. Ensiksi JDK 7, joka päivittyi JDK 8:aan. JDK:n päivittymisellä ei ole mitään muuta vaikutusta työhön.

Kun Android Studio sekä muut tarvittavat tiedostot on ladattu ja JDK asennettu, ajetaan Android Studion asennus latauskansiosta. Kun ohjelma

kysyy mitä asennetaan, asennetaan kaikki komponentit. Jos haluaa muuttaa tiedostojen asennuskansioita, tulee ottaa huomioon, ettei niitä asenneta samaan hakemistoon. Päivitykset eivät toimi, jos ne ovat samassa kansiossa.

Ensikäynnistyksellä Android Studio lataa erilaisia tärkeitä komponentteja. Tätä asennuksen vaihetta ei tule keskeyttää.

Android Studio kannattaa laittaa automaattisesti päivittämään Android SDK:n kirjastot. Oletuksena se ei ole automaattinen. Tämä tapahtuu Android Studion sisältä asetuksista. Asetukset:

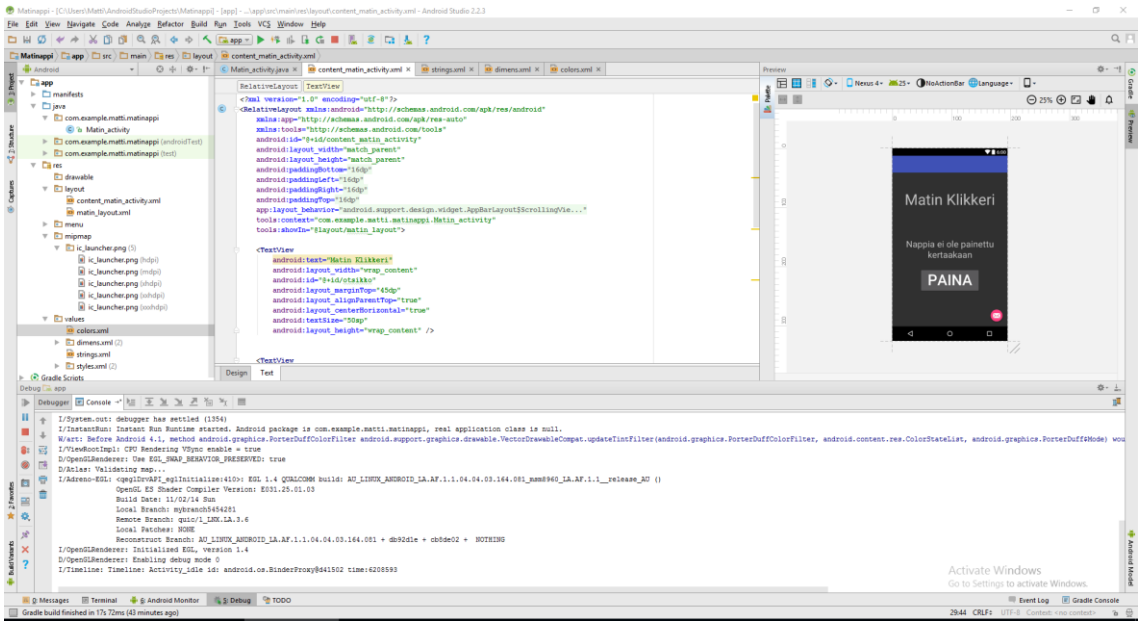
File > Settings > Appereance & Behavior > System Settings > Updates

Ruksataan “Automatically check updates for Android SDK” ja klikataan “OK”. (Google 2017)

3.2 Android Studion käyttöliittymän esittely

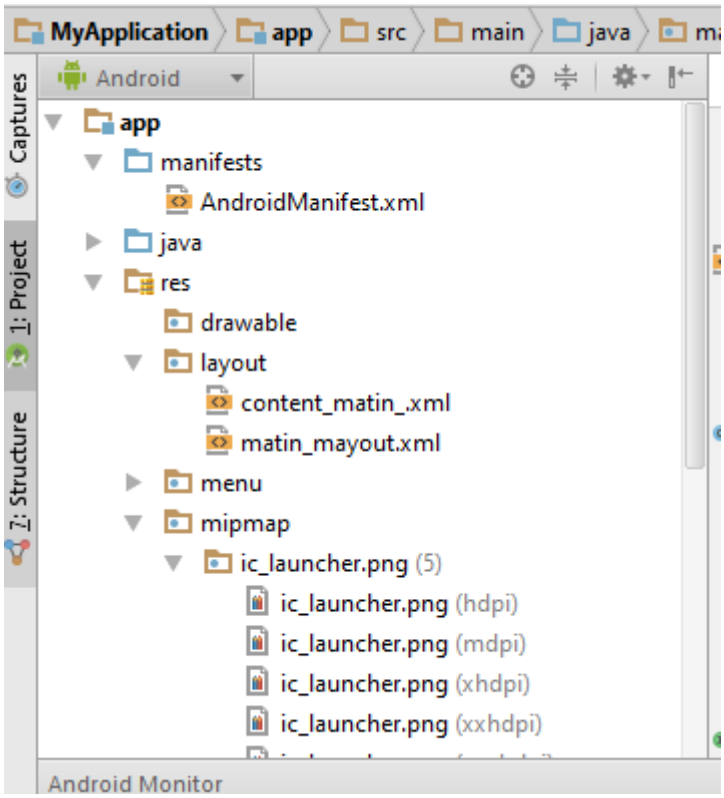
Käyttöliittymää esitellään auki olevan projektin avulla. Tämä kappale on tehty myöhemmin tulevan ”Projektin luonti” -kappaleen jälkeen, joten joitakin jo toteutettuja osia näkyy näissä esittelykuvissa.

Projektin käsittelyn käyttöliittymän yläosassa on kaikki oletustyökalut niin kuin yleensä sovelluksissa on. Kuvassa 2 nähdään Android Studion projektin kokonäkymä.



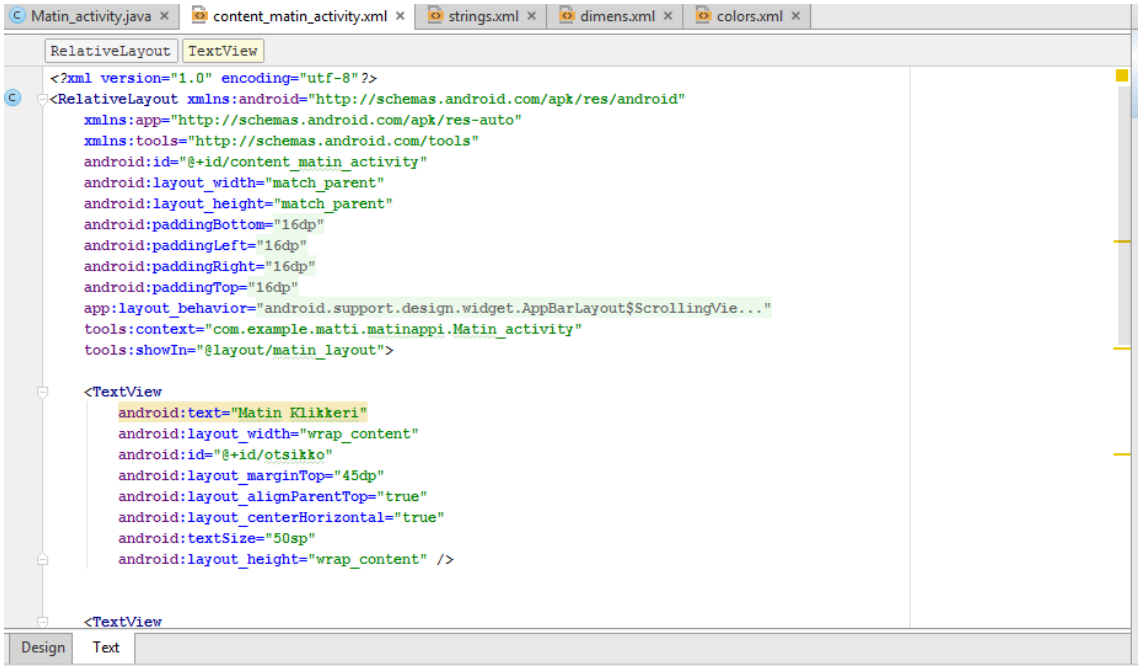
Kuva 2. Koko näkymä.

Ohjelman vasemmalla puolella on hakemisto, jossa näkyvät kaikki tiedostot projektiin liittyen. Niitä osia klikkaamalla pääsee tarkastelemaan tai muokkaamaan niiden koodia.

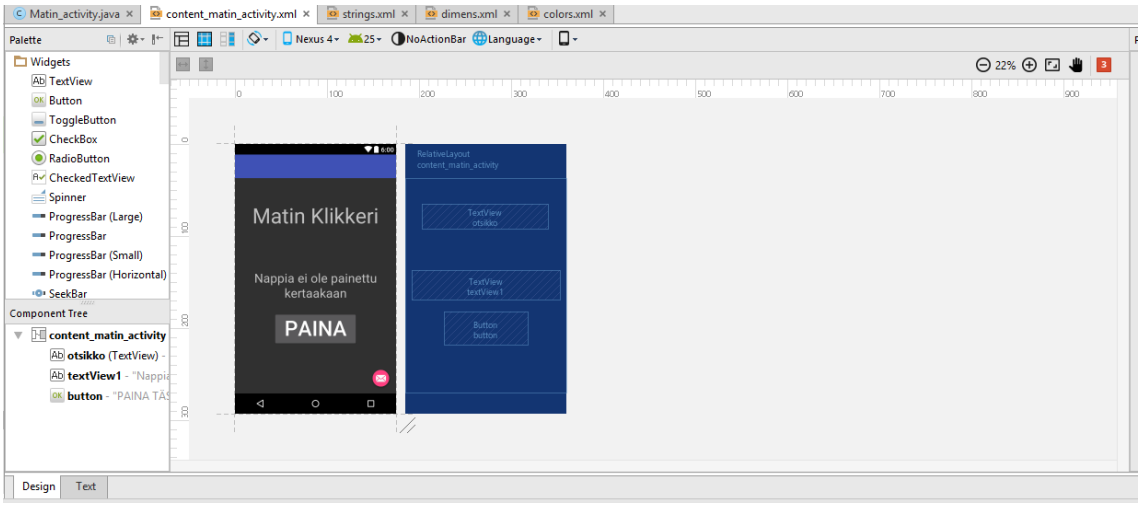


Kuva 3. Hakemisto, jossa kaikki projektin tiedostot.

Tämän hakemiston oikealla puolella on muokkausosio, johon saa useita tiedostoja auki välilehdille. Samaan kohtaan tulevat myös visuaaliset sisällönmuokkaustyökalut sekä esikatselu. Niitä vaihdellaan edestakaisin tarpeen mukaan.

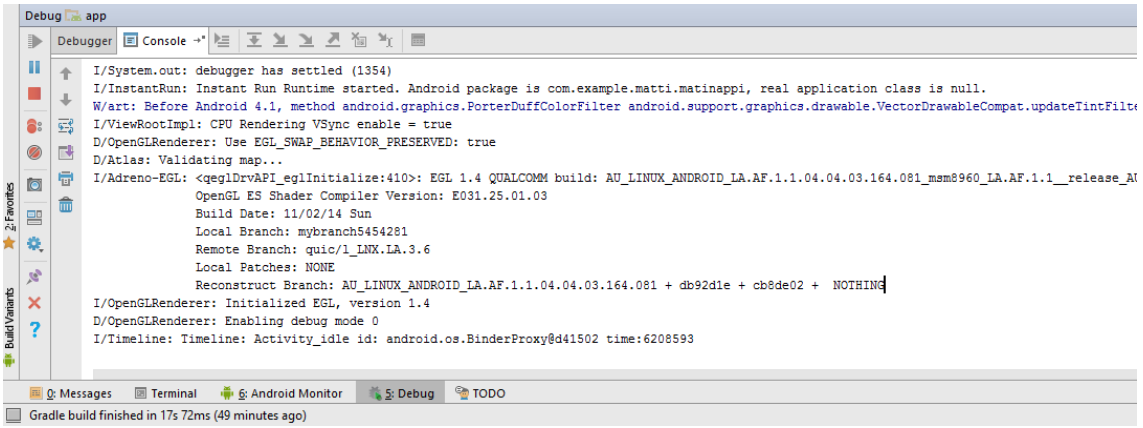


Kuva 4. Koodi editori ja välilehdet.



Kuva 5. Visuaalinen editori ja toimintovalikko.

Ohjelman alaosiossa on loki, joka kertoo tapahtumat esimerkiksi oman sovelluksen ajossa. Tämä on todella hyödyllinen osa Android Studiota, koska kaikki virheet jäävät talteen lokiin.



Kuva 6. Havaintotyökalut.

Käyttöliittymää pystyy helposti muokkaamaan ja käyttämään eri tavoin. Osioita voi siirrellä, poistaa ja lisätä tarpeen mukaan. Oletusasetuksiin ei kuitenkaan koskettu, vaan käytettiin oletuksena olevaa näkymää.

Oletusnäkymä toimii erittäin hyvin ja sen asettelu on käyttäjäystävällinen. Työtä tehdessä ei ollut tarvetta muokata näkymää. Tähän voi olla syynä se, että työ käsittelee vain Android-sovelluskehityksen perusteita eikä vaikeimpia työkaluja ja ominaisuuksia tarvita. Android Studion opiskelussa on hyvä käyttää oletusnäkymää myös sen takia, että suurin osa internetissä olevasta materiaalista on tehty ilman muokkauksia.

4 SOVELLUSKEHITYKSEN OSAT

Opinnäytetyössä luodaan täysin uusi projekti, joten seuraavat asetukset on hyvä tehdä huolella. Asetukset koskevat yleisiä asioita kuten nimeämisiä sekä projektin sisältämiä oletuksia. Alkuvaiheessa liitetään oma Android-puhelin Android Studioon, jonka kautta pystytään tarkastelemaan ja testaamaan sovellusta. Tästä jatketaan eteenpäin käyttöliittymän luontiin sekä sovelluksen sisäisiin toimintoihin.

4.1 Projektin luonti

Avataan Android Studio ja aloitetaan projekti valitsemalla ”Start a new Android Studio project”. Annetaan projektille nimi ja valitaan tallennuskansio ja painetaan ”Next”.

Sovellus tehdään puhelimelle ja tabletille, joten ”Target Android Devices” kohdassa tulee ruksata vain ”Phone and Tablet”. Samassa kohdassa on myös ”Minimum SDK”, joka tarkoittaa pienintä käytettävää kirjastoversiota. Tämä on yhteydessä Android-käyttöjärjestelmäversioihin. Mitä pienempi versio on kyseessä, sitä vähemmän ominaisuuksia löytyy. Käytössä oleva laite käyttää Android 5.1.1 versiota, mutta projektin tekoon kannattaa valita alhainen Android-versio. Valitaan Android 2.3.0 minimi SDK:ksi, koska työssä opiskellaan yksinkertaisia elementtejä. Kuvassa 7 on esiteltynä tämän kohdan tarjoamia vaihtoehtoja.

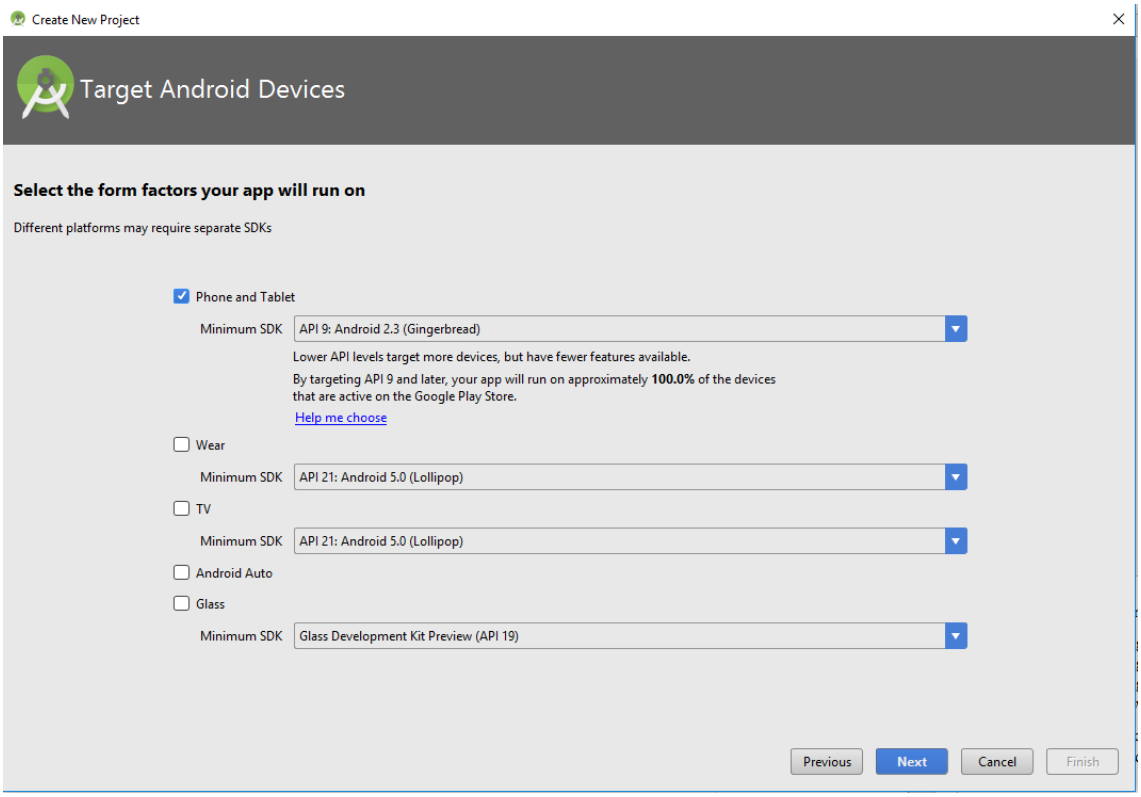
Kun valitaan näin aikainen Android-versio, kaikki uusimpien Android-versioiden tarjoamat ominaisuudet jäävät pois. Tämä kuitenkin tekee sovelluksesta yhteensopivan kaikkien 2.3.0 ja sen yli olevien Android-versioiden ja laitteiden kanssa. Version valinta kertoo, kuinka monta prosenttia Google Play -kaupassa aktiivisena olevista puhelimista pystyy käyttämään kehitettävää sovellusta. Minimum SDK:n ollessa 2.3.0, Android Studio kertoo, että 100 % aktiivisista puhelimista ja tableteista pystyy käyttämään projektin sovellusta. ”Next” -painikkeella eteenpäin.

Seuraavaksi tulee ”Add an activity to Mobile” -kohta, joka tarjoaa erilaisia toimintoja valmiina. Kun kyseessä on uusi sovellus ja perehdytään enemmänkin tärkeisiin Android-sovelluskehityksen osiin, ei valita valmiita toimintoja, vaan tehdään oma alusta lähtien. Tästä syystä valitaan ”Basic Activity” ja painetaan ”Next”.

”Cuztomise the activity” -kohdassa nimetään projektin eri osat. Nämä kannattaa nimetä järkevästi. Opinnäytetyön projektissa osat nimettiin seuraavasti:
Activity Name: Matin_Activity
Layout Name: matin_layout
Title: Matin_activity
Menu Resource Name: matin_menu

Lopuksi painetaan ”Finish”, jolloin Android studio avaa projektin muokkaustilaan. Android Studion projektinluonti luo oletuksella ”Hello World”

-tekstin. ”Hello World” on tyypillinen esimerkki pienestä koodin osista, jolla testataan. Sovelluksen pystyy ajamaan suoraan projektin teon jälkeen.



Kuva 7. Laite ja Android versioiden valinta.

4.2 Oman sovelluksen ajaminen

Ajamisella tarkoitetaan sovelluksen käynnistystä (debugging), jotta pystytään testaamaan miten sovellus toimii. Ajon yhteydessä huomataan mahdolliset virheet sekä korjauskehotukset. Tässä osiossa tutustutaan Android Studion tarjoamiin testausmenetelmiin.

4.2.1 Ajaminen Android-laitteella

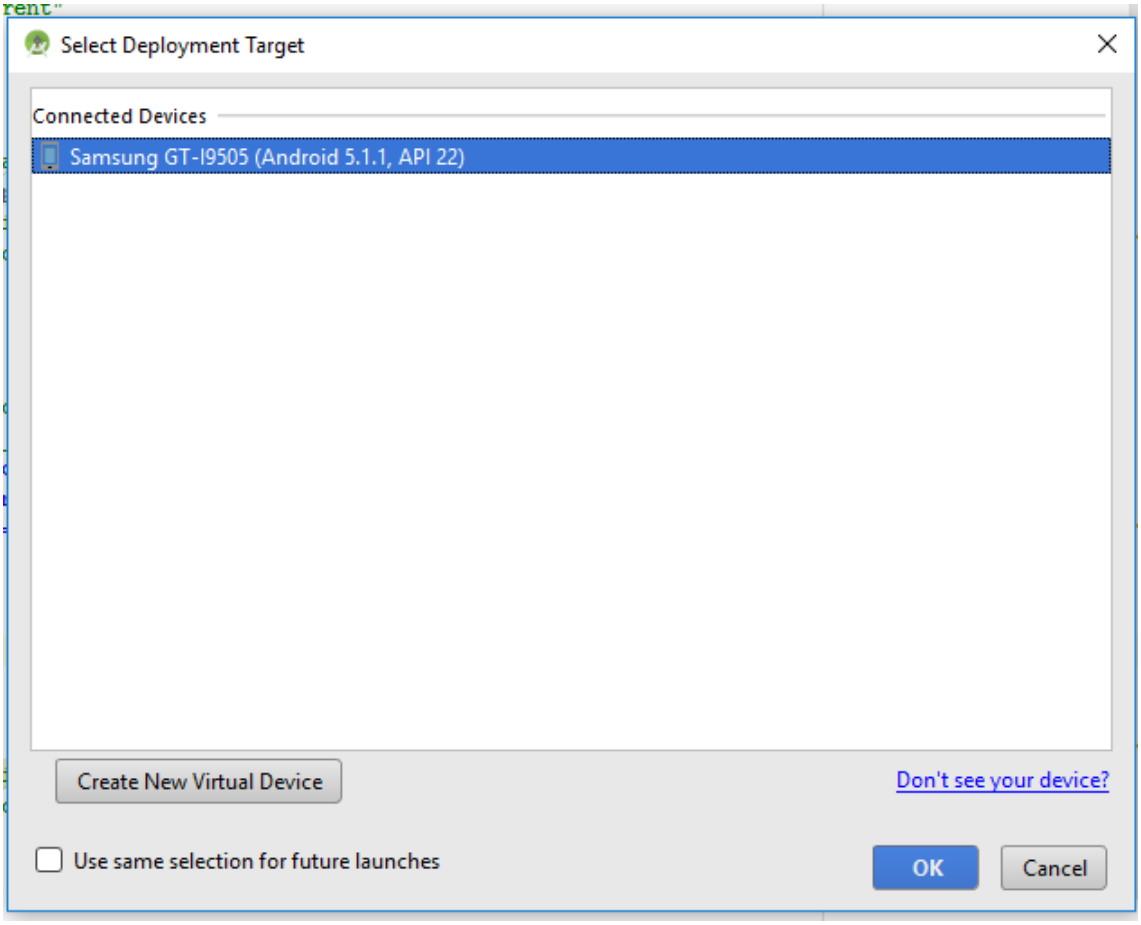
Jotta projektia pystyy ajamaan ja testaamaan laitteen kautta, käytössä täytyy olla sovellusta vastaava laite. Android Studiolla pystyy kehittämään sovelluksia eri laitteille esim. TV:t, puhelimet, tabletit ja kellot.

Tämän opinnäytetyön projekti on puhelimille ja tableteille. Käytössä oleva laite käyttää Android 5.1.1 -versiota, joten sen asetuksista löytyy ”USB debuggin” -asetus. Tämän tulee olla päällä, jotta Android Studion kautta pystyy ajamaan kehityksessä olevan sovelluksen puhelimella.

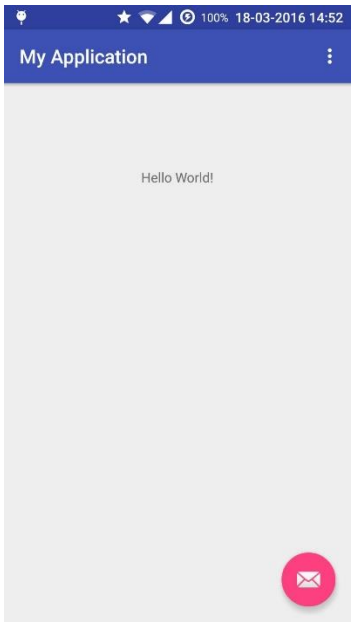
Ensiksi avataan Android Studio ja projekti, jota työstetään. Tämän jälkeen laite tulee liittää tietokoneeseen kiinni, jolloin Android Studio tunnistaa puhelimen ja lokiin tulee ”xxxx connected”. Tämän jälkeen laitteella voi ajaa oman sovelluksensa painamalla ”Run” Android Studion työkalupal-

kista. Valitaan käytössä oleva laite ja painetaan ok. Valinnan esimerkki näkyy kuvassa 8.

Kun sovelluksen ajaa puhelimen kautta, sen ajon aikainen versio asentuu puhelimeen. Jos USB-yhteyden katkaisee jälkeenpäin, oman sovelluksen voi käynnistää pelkästään puhelimen kautta.



Kuva 8. Oman laitteen valinta ajon aikana.

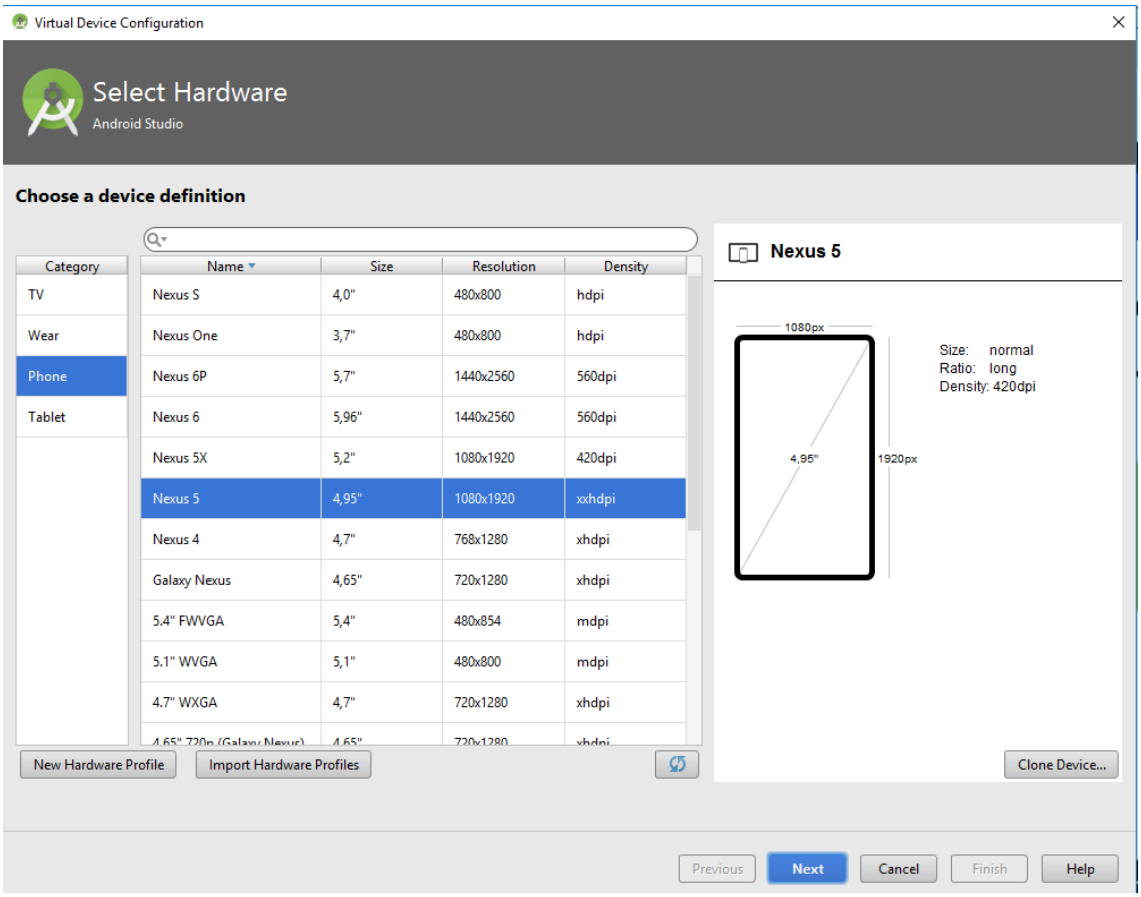


Kuva 9. Sovellus ajettuna ensimmäistä kertaa.

4.2.2 Ajaminen Android Studiossa

Jos ei omista Android-laitetta, sovelluksen pystyy ajamaan virtuaalisesti Android Studion kautta. Ajetaan samoin tavoin kuin omalle laitteelle, mutta laitteen sijaan valitaan ”Create Virtual Device” ja valitaan haluttu laite (näytön koko ja tarkkuus) ja haluttu käyttöjärjestelmä. Ajetaan painamalla ”Run”, jolloin valitaan juuri luotu virtuaalinen laite. Tällöin sovellus ajetaan tietokoneella emuloimalla valittua laitetta.

Tämän opinnäytetyön teossa ei käytetty emulointivaihtoehtoa, sillä on mukavampaa ja käytännöllisempää käyttää olemassa olevaa laitetta sovelluksen ajoon ja testauksiin. Kuvissa 10 ja 11 näkyy emuloinnin asetuksia.



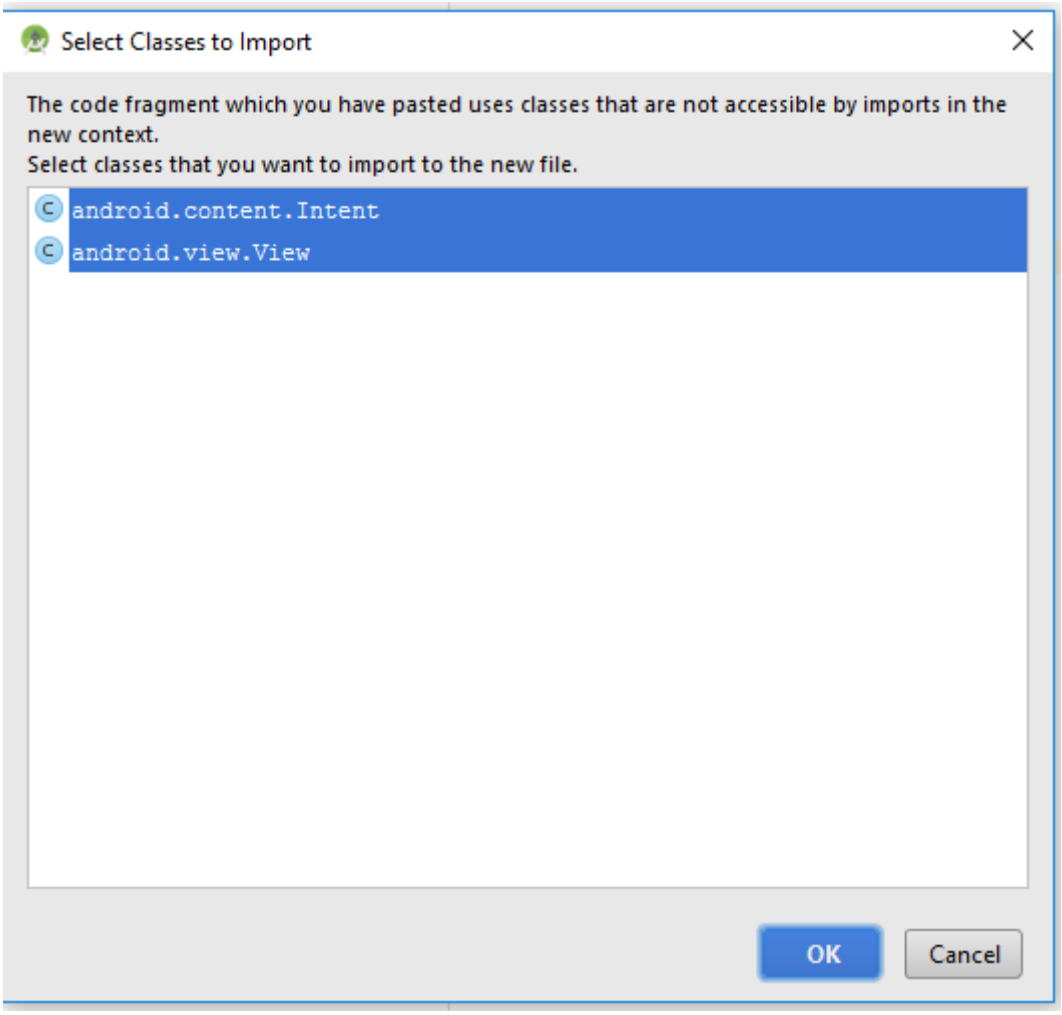
Kuva 10. Virtuaalisen laitteen luonti.

Kuva 11. Virtuaalinen laite ajo vaihtoehtona.

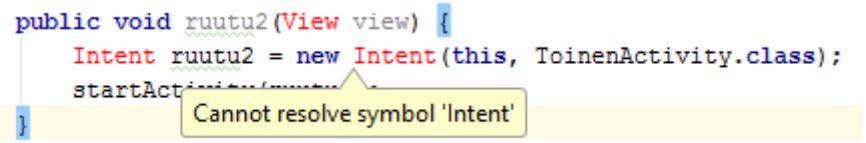
4.3 Koodin luokkien liittäminen projektiin

Joissakin tilanteissa, kun Android Studiolla ohjelmoidaan, tarvitsee käyttää luokkia, jotka eivät ole oletuksena käytössä suoraan. Luokilla tarkoitetaan Android Studioon valmiiksi luotuja toimintoja, jotka helpottavat ohjelmointia.

Luokkien liittämistä ei kuitenkaan tarvitse tehdä etukäteen. Kun koodia luodaan ja luokkia käytetään, Android Studio saattaa tunnistaa luokat ja tarjota mahdollisuutta tuoda ne suoraan projektiin. Jos Android Studio ei tunnista automaattisesti luokkia, se antaa virheilmoituksen ”Cannot resolve symbol xxxx”. Tästä virheestä tietää, että joitakin luokkia puuttuu ja sen takia koodi ei toimi. Tarvitut luokat voi helposti liittää painamalla ALT + ENTER.



Kuva 12. Android Studio tunnisti luokat.



Kuva 13. Android Studion luokka -virheilmoitus.

4.4 Muokkaukset ulkoasuun

Android Studio luo automaattisesti yksinkertaisen ulkoasun. Tätä ulkoasua pystyy helposti muokkaamaan muuttamalla sen koodia. Esimerkiksi otsikon oletus on ”My Application” ja värit sininen ja musta. Näitä pystyy helposti muuttamaan klikkaamalla auki eri tiedostoja Android Studion vasemmalta puolelta. Ulkoasua pystyy muokkaamaan, kunhan tietää mitä mikäkin koodin osa tarkoittaa.

Sovelluksen otsikkoa pystyy muuttamaan ”strings.xml” -tiedostosta. Tiedoston sisältä löytyvät arvot, jotka vaihtuvat korvaamalla tekstin:

```
<resources>
    <string name="app_name">My Application</string>
    <string name="action_settings">Settings</string>
</resources>
```

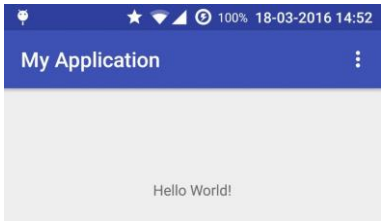
Vaihdetaan:

```
<resources>
    <string name="app_name">Matin Appi</string>
    <string name="action_settings">Settings</string>
</resources>
```

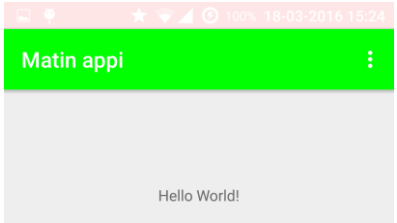
”Values” -alavalikosta löytyvät myös arvot sovelluksen perusväreille. Värit toimivat värikoodeilla, joten etsimällä halutun värin, perusvärit saa vaihdettua melkein miksi vain halutaan.

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Tiedostovalikosta löytyy kaikenlaisia muitakin oletusasetuksia, joita muuttamalla pystyy hienosäätämään sovellusta. Näitä tulee muuttaa sovelluskehitysten tarpeiden mukaan. Tässä työssä ei käydä läpi jokaista asetusta, koska niistä voisi tehdä kokonaan oman tutkimuksen.

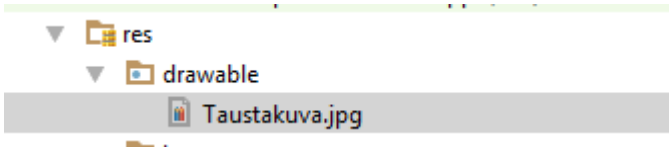


Kuva 14. Oletus otsikko ja värit.



Kuva 15. Otsikko ja värit vaihdettuina.

Sovellukselle pystyy helposti lisäämään halutun taustakuvan. Kuva pitää tuoda osaksi projektin tiedostoja. Tämä tapahtuu kopioimalla haluamansa kuvan ja liittämällä sen ”drawable” -kansion alle Android Studion hakemistoon.



Kuva 16. Taustakuvan liittäminen projektiin.

Kun taustakuva on osana projektin tiedostoja, tulee sovelluksen sisällön koodiin (content.xml) lisätä koodirivi, joka ottaa kuvan käyttöön. Koodi tulee lisätä ”content.xml” alkuun, jotta se pätee koko aktiviteettiin.

Koodi:

```
android:background="@drawable/Taustakuva"
```

Kuva 17. Sovellus taustakuvalla.

Useille elementeille pystyy taustakuvan lisäksi liittämään omia kuvia. Esimerkiksi nappuloiden taustoiksi voi laittaa mahdollista teemaa mukailevat taustat.

4.5 Yksinkertaisen käyttöliittymän luonti

Android Studio tarjoaa sovellukseen helposti liitettäviä perusominaisuuksia (esim. kalenteri, tekstilaatikot, napit, kuvat jne.). Näillä pystyy luomaan käyttöliittymän sovellukselle. Jos haluaa monipuolisemman sovelluksen kuin ”vain yksi ruutu”, täytyy lisätä aktiviteettejä (uusia ”ruutuja”, joihin kytkentä napin tms. toiminnon kautta). Jos aktiviteetille ei lisää kytköstä, se voi toimia myös tausta-aktiviteettina ja tehdä jotakin toimintoa automaattisesti.

Yksinkertaiseen käyttöliittymään liitetään ohjelmointia Javalla, jotta sovellukseen saadaan varsinaisia toiminnollisuuksia. Opinnäytetyön sovelluksen päätoiminnollisuus pohjautuu nappiin sekä sen takana olevaan koodiin. Kyseessä on toiminto, joka laskee sovelluksessa olevan napin painallukset ja päivittää tekstiä sen mukaan.

Seuraavat kappaleet pohjautuvat ”painallusten lasku” -toiminnollisuuteen ja sen avulla tutustutaan vaadittuihin peruselementteihin. Näitä peruselementtejä ovat mm. tekstilaatikko, nappi ja niiden koodi.

4.5.1 Elementtien visuaalinen ja koodipohjainen asettelu

Android Studion luomaa oletusulkoasua ei tarvita, joten olemassa olevat elementit saa poistettua klikkaamalla niitä ja painamalla näppäimistön ”Delete” -nappia.

Uusia elementtejä pystyy asettelemaan helposti suoraan Android Studion tarjoamasta listasta. Seuraavassa osiossa käsitellään tekstiruutuja ja nappuloita.

Sovellukseen liitetään visuaalisesti otsikkoteksti ja painettava nappula. Android Studio luo automaattisesti näille elementeille koodin ”content_matin_.xml:n” sisälle. Ennen kuin muokataan elementtejä, muokataan elementit oikean kokoisiksi ja oikeilla attribuuteilla.

Tekstilaatikoiden ja nappulan koodi:

```
<TextView
    android:text="Matin Klikkeri"
    android:layout_width="wrap_content"
    android:id="@+id/otsikko"
    android:layout_marginTop="45dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textSize="50sp"
    android:layout_height="wrap_content" />


<TextView
    android:text="Nappia ei ole painettu kertaakaan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:id="@+id/textView1"
    android:layout_marginBottom="135dp"
    android:textSize="30sp"
    android:gravity="center"/>

<Button
    android:text="PAINA TÄSTÄ"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="97dp"
    android:id="@+id/button"
    android:width="200sp"
    android:height="80sp"
    android:textSize="50sp"
    />
```

Elementtien tekstejä ja ID:tä voi muokata koodin tai elementtien ominaisuuksien kautta. Molemmat vaihtoehdot saavat saman tuloksen aikaan. Yleisiä muutoksia ovat tekstin sisältö, koko, väri jne.

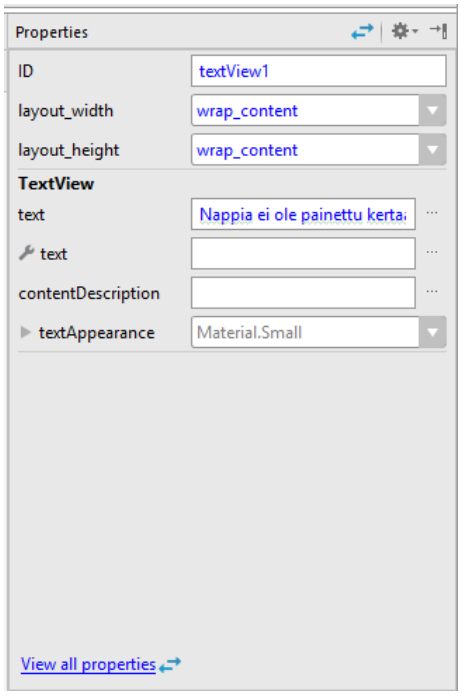
Värin muutoksen esimerkki:

```
android:textColor="#000000"
```

Asettelemalla teksti- ja nappielementit visuaalisesti Android studio luo asettelun mukaisen koodin. Koodiin lisätään osia, jotta saadaan tekstin kokoa (android:textSize) ja napin kokoa muutettua (android:width, android:height). Painalluksia laskevan tekstin asettelua muokataan, jotta se olisi keskitetty (android:gravity=”center”). Kuvassa 18 on esimerkkejä tekstilaatikon ominaisuuksista.

```
<TextView
    android:text="Nappia ei ole painettu kertaakaan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:id="@+id/textView1"
    android:layout_marginBottom="135dp"
    android:textSize="30sp"
    android:gravity="center"/>
```

Kuva 18. Teksti -elementin ominaisuudet koodina.



Kuva 19. Teksti -elementin ominaisuudet visuaalisesti.

Seuraavaksi tuleva napin toiminnollisuus korvaa tekstin oletusarvon (override).

4.5.2 Elementtien toiminnallisuudet

Ennen kuin napille ja laskurille koodataan niiden toiminnot, tulee siistiä koodi jo aiemmin poistetuista visuaalisista elementeistä (esim. toolbar eli yläpalkki), joita ei enää tarvita. Aiemmin opiskelluista koodikielistä on hyötyä koodia siistittäessä. Vaikka Java onkin vielä tuntematon, koodin pystyy helposti puhdistamaan. Jos yrittää poistaa jotakin tärkeätä koodista, Android Studio huomauttaa heti asiasta, eikä sovellusta tarvitse erikseen ajaa huomatakseen mahdolliset virheet.

Toiminnollisuuden lisääminen saattaa tuottaa hankaluuksia, mutta siihen löytyy apua internetistä. Sieltä löytää helposti esimerkiksi, miten napille saa asetettua toiminnon sekä miten napin painallus muokkaa tekstiä (Android Developers, 2016). Painikkeen koodista ei tullut kovin pitkä:

```
public class Matin_Activity extends AppCompatActivity {

    int counter = 0

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.matin_mayout);


        Button button=(Button) findViewById(R.id.button);
         text = (TextView) findViewById(R.id.textView1);
        button.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                counter++;
                text.setText("Olet painanut nappia\n" +
counter + "\nkertaa");
            }
        });

    }
    }
```

Koodiin pitää sisällyttää muuttuja ”int counter = 0” (int eli numeraalinen), jotta laskurin saa toimimaan. Muutoin alkuosa koodista on Android Studion oletuksena luomaa. Elementit, jotka asetettiin visuaalisesti, piti ”tunnistaa” niiden ”ID:n” avulla, jotta sovellus tiesi mistä napista ja mistä tekstistä on kyse.

ID:n tunnistus

```
(Button button=(Button)findViewById(R.id.button);
text=(TextView)findViewById(R.id.textView1);
```

Tämän jälkeen napille pystyy asettamaan painallus -toiminnon, jossa käytettetään alussa määritettyä laskuria (counter). Napin painallus asettaa uuden tekstin vanhan tilalle (text.setText), kun sitä painetaan. Tästä syystä laskurin lukeman lisääntyminen on tärkeää (counter++; lisää laskuriin +1 edelliseen.). Uusi teksti on siis ”Olet painanut nappia + (nykyinen painalluksien määrä) + kertaa”. Painalluksien jälkeen tulevassa tekstissä on ”\n”. Tämä tarkoittaa rivinvaihtoa. Koodiin lisättiin rivinvaihto, jotta teksti olisi selvempi.

Toimintojen liittäminen eri elementeille on helppoa, kun tietää mitä hakee. Android Studio ehdotti mahdollisia vaihtoehtoja sekä täydensi koodia automattisesti oikeaan suuntaan.

4.5.3 Reset-painike

Kun toiminnallisuutena on painallusten laskuri, tulee sovelluksessa olla myös Reset-nappi. Laskuriin käytettyä koodia muokkaamalla saadaan helposti tehtyä Reset-painike, joka korvaa (override) jälleen kerran tekstilaatikon tekstin.

Ennen koodin muokkaamista uudelle koodille tehdään oma nappi visuaalisella puolella, jolle annetaan samoin tavoin asettelua ja ulkoasua muokkaavat ominaisuudet kuin laskurin napille.

Laskurin koodi:

```
Button button=(Button) findViewById(R.id.button);
final TextView text = (TextView) find-
ViewById(R.id.textView1);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        counter++;
        text.setText("Olet painanut nappia " + counter + "
kertaa");
    }
});
```

Laskurin koodi muokattuna resetiksi:

```
Button button2=(Button) findViewById(R.id.button2);
final TextView text2 = (TextView) find-
ViewById(R.id.textView1);
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        counter = 0;
        text2.setText("Nappia ei ole painettu kertaakaan");
    }
});
```

Uudessa koodissa luotiin uudet muuttujat (button2, text2), jotta nappulat eivät tee samoja asioita. Muuttuja ”counter” vaihtuu napin painalluksesta numeroon ”0” ja teksti vaihtuu oletustekstiksi. Nappeja voi painella miten

haluaa, koska niiden ainoa ”yhteinen tekijä” on tekstilaatikko, jonka sisältö korvataan joka painalluksella.

4.6 Aktiviteetit

Aktiviteetti on ns. yksi näkymä sovelluksesta. Aktiviteettien välillä pystyy liikkumaan useilla eri tavoilla, jolloin sovelluksesta tulee monipuolisempi eikä ”vain yksi ruutu”. Niiden välillä liikutaan esimerkiksi nappuloiden avulla.

Aktiviteetit ovat tärkeimpiä peruselementtejä, sillä niiden päälle luodaan muut toiminnollisuudet. Aktiviteeteillä sovelluksista voi tehdä useita eri toimintoja sisältäviä kokonaisuuksia.

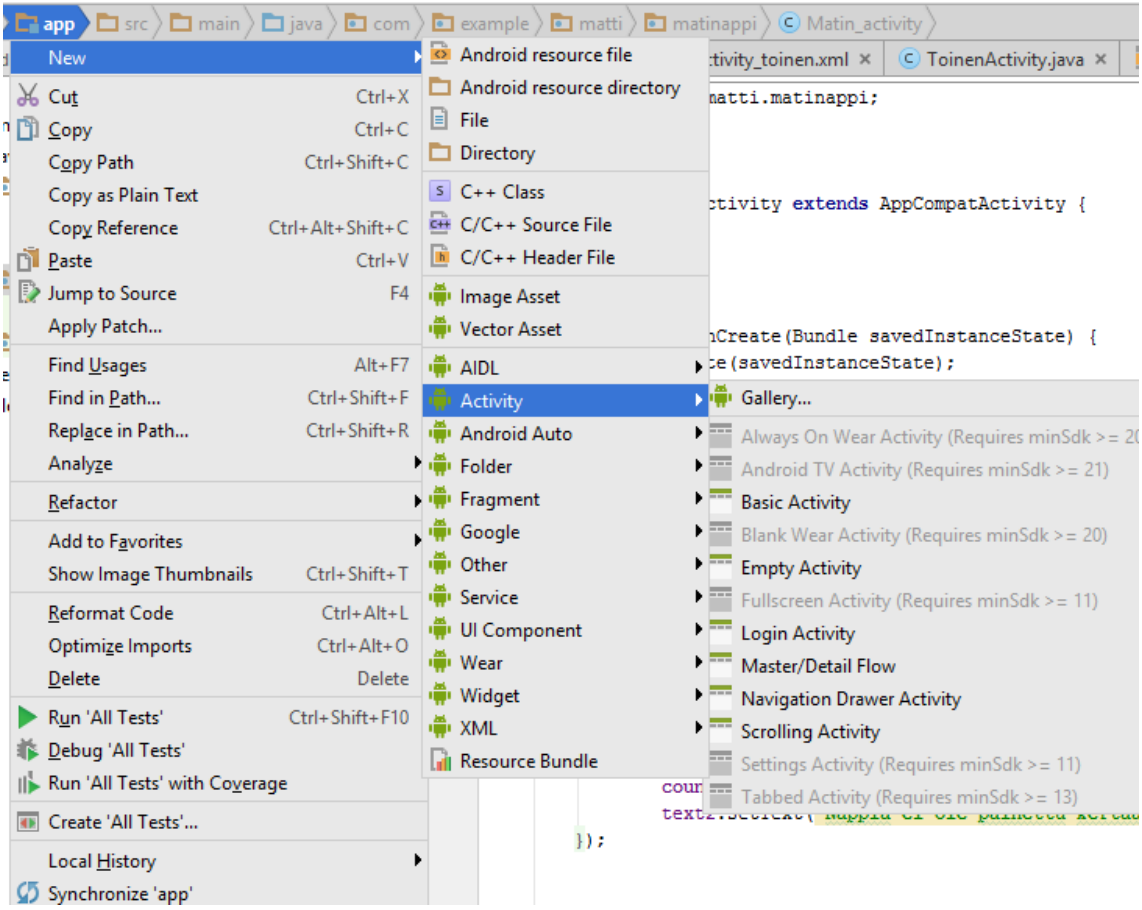
Projektin alussa luotiin ensimmäinen aktiviteetti ja tähän saakka vain sitä on käytetty sovelluksen pohjana. Jotta sovellukselle saadaan monipuolisuutta, tulee lisätä uusi aktiviteetti ja tehdä ”liitos” vanhan ja uuden aktiviteetin välille.

Uutta aktiviteettia varten tulee luoda toiminto, joka vaihtaa aktiviteetteja. Toimintona voi käyttää mitä vain laukaisinta, joista helpoin mahdollinen on uusi nappula. Luodaan uusi nappula ja annetaan sille muotoiluun tarvittavat attribuutit.

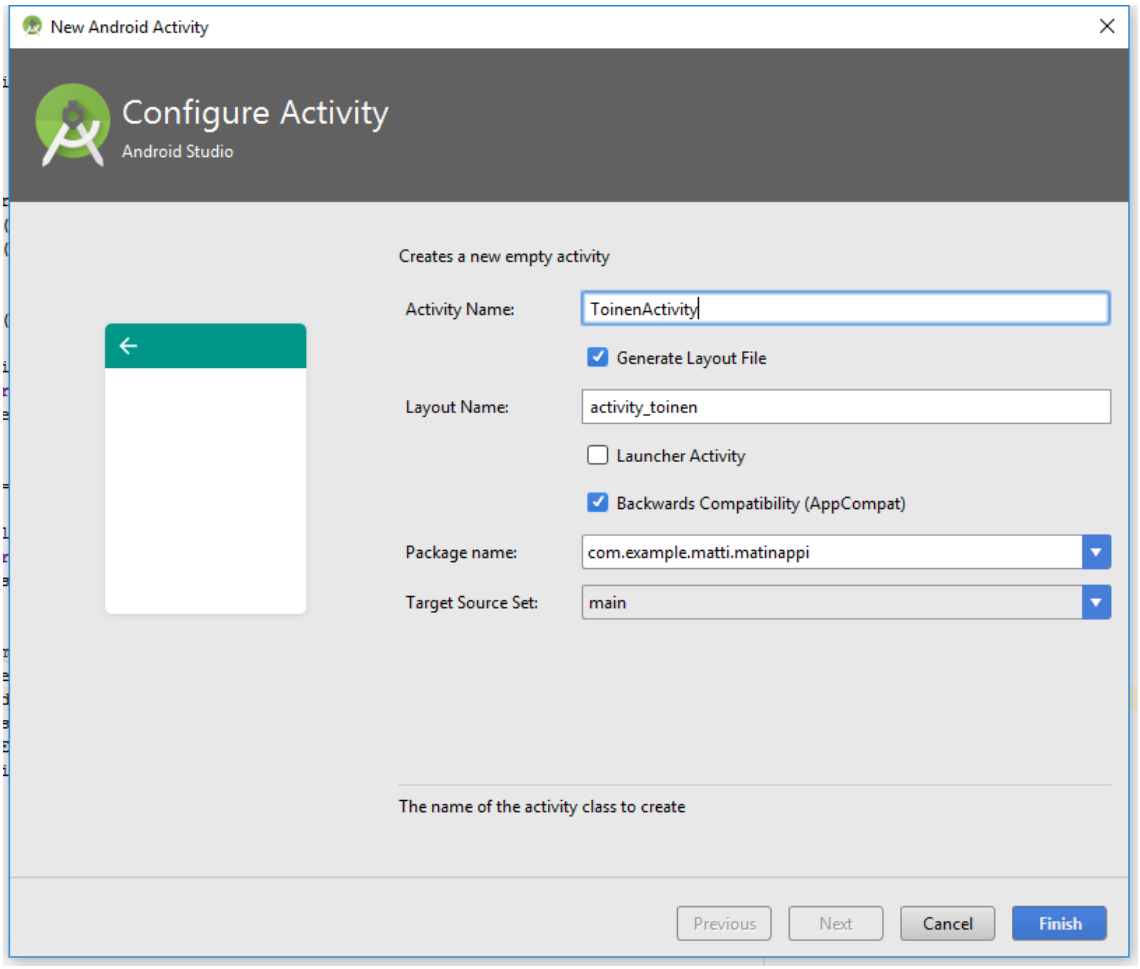
Nappulalle liitetään toiminto muotoilun jälkeen. Napin attribuutteihin tulee liittää ”onclick”-funktio seuraavasti:

```
android:onClick="ruutu2"
```

Kun nappia painetaan, se kutsuu koodista ”ruutu2” -nimistä metodia, joka luodaan seuraavaksi. Koodattu metodi määrittää mitä napin painalluksesta tapahtuu. Ennen kuin koodilla pystyy siirtymään uuteen aktiviteettiin, se pitää luoda. Kuvasta 20 näkyy, mistä aktiviteetti luodaan. Kuvassa 21 näkyy uuden aktiviteetin luonnin asetukset. Aktiviteetin nimeämiset kannattaa tehdä huolellisesti. Niitä käytetään osana koodia, jotta tiedetään mihin siirrytään.



Kuva 20. Uusi aktiviteetti.



Kuva 21. Aktiviteetin luonti.

Kun uusi aktiviteetti on luotu, tulee alkuperäinen ja uusi liittää yhteen. Liittämiseen käytetään Intent-objektia. Se tarjoaa ajonaikaisen liitoksen kahden erillisen komponentin välille kuten tässä tapauksessa kahden aktiviteetin välille. Intent-objekteja pystyy käyttämään todella monipuolisesti Android-ohjelmoinnissa. Se kuvastaa sovelluksen aikomusta tehdä jotain (Intent on suomennettuna aikomus). Pää-aktiviteettiin liitetään pohjalle seuraava koodi, jota kutsutaan aiemmin mainitulla ”nappulan painallus” -attribuutilla:

```
public void ruutu2(View view) {
    Intent ruutu2 = new Intent(this, ToinenActivity.class);
    startActivity(ruutu2);
}
```

Nyt nappulalla tulisi päästä uuteen tyhjään aktiviteettiin. Siellä ei ole mitään toimintoja, joten niitä pitää luoda. Ne voidaan luoda erillään pääaktiviteetista. Tällainen yksinkertainen aktiviteetista toiseen siirtyminen toimisi täydellisesti esim. sovelluksen valikon osana.

Luodaan uuteen aktiviteettiin ”paluunappi” samalla tavalla kuin pääaktiviteettiin. Luodaan nappi tyhjään aktiviteettiin visuaalisesti ja liitetään napille painalluksesta aiheutuva koodi:

```
android:onClick="ruutu1"
```

Liitetään uuden aktiviteetin koodiin sama koodi eri muuttujilla:

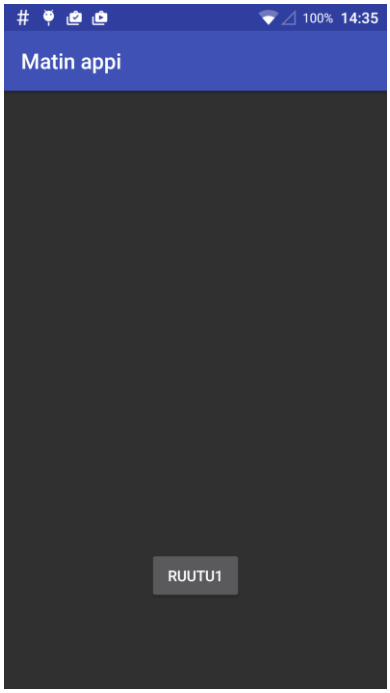
```
public void ruutu1(View view) {
    Intent ruutu1 = new Intent(this, Matin_activity.class);
    startActivity(ruutu1);
}
```

Liitos on luotu ja sovellusta ajettaessa pystyy siirtymään aktiviteettien välillä vapaasti.

Uudessa projektissa yllä mainitut koodit saattavat aiheuttaa ”Cannot resolve” -virheilmoituksen. Tässä tilanteessa kyseisten toimintojen luokat puuttuvat ja ne täytyy tuoda projektiin painamalla ALT + ENTER.

Kuva 22. Sovelluksen lopputulos.

Kuva 23. Sovelluksen lopputulos.



Kuva 24. Sovelluksen lopputulos toinen aktiviteetti.

5 YHTEENVETO

Perehtyminen Android Studioon ja sen käyttöön vaati lähes yhtä paljon aikaa kuin itse sovelluksen toteuttaminen. Alussa työn aihe oli liian laaja ja sitä joutui karsimaan huomattavasti. Alkuperäisestä tavoitteesta olisi tehnyt kaksi ellei kolmekin tämän mittaista työtä.

Lopputulokseksi saatiin toimiva sovellus, jonka avulla opeteltiin Java-koodikieltä ja Android Studion ominaisuuksia. Niitä kaikkia ei edes ole järkevä dokumentoida tähän työhön. Sovelluksesta tuli loppujen lopuksi jopa ihan järkevä sovellus. Sen ulkoasua pystyy helposti muokkaamaan eteenpäin, mutta design ei ollut tämän työn tavoitteissa mukana. Sovelluksen myötä Android-sovelluskehitys tuli selvemmäksi ja sen mahdollisuudet aukenivat enemmän. Itse opiskeltuna opinnäytetyön aihe on työläs ja jokainen uusi asia vaatii jonkin verran opettelua. Sovelluksen teko ja opinnäytetyö etenivät hyvässä tahdissa ilman liian vaikeita asioita. Projekti antoi hyvän tuntuman Java-koodikieleen. Aiemmin opituista koodikielien logiikoista oli paljon hyötyä. Projektin tekoon sai apua Android Studion sivustolta. Sieltä löytyy Android Studioon hyvät ohjeistukset.

Sovellusta pystyy helposti jatkokehittämään ja muokkaamaan, jos haluaa opiskella lisää tämän opinnäytetyön tavoin. Tämän projektin jatkaminen olisi hyvää opiskelua Android-sovelluksien ohjelmoinnista. Näin alkuvaiheessa oleva projekti kattaa vain pienen osan ja syventymällä enemmän Android Studioon pystyisi luomaan jopa kaupallistettaviakin sovelluksia. Kuvissa 22, 23 ja 24 näkyy sovelluksen lopputulos.

LÄHTEET

Google, 2017a. Android. Haettu 15.01.2017
https://www.android.com/

Google, 2017b. Android arkkitehtuuri. Haettu 17.01.2017
https://developer.android.com/guide/platform/index.html

Google, 2017c. Android Studio Overview. Haettu 20.01.2017
http://developer.android.com/tools/studio/index.html

Google, 2017d. Android Studio system requirements. Haettu 20.01.2017
http://developer.android.com/sdk/index.html#Requirements

Google, 2017e. Button. Haettu 28.01.2017
https://developer.android.com/reference/android/widget/Button.html

Google, 2017f. SDK Manager. Haettu 20.01.2017
http://developer.android.com/tools/help/sdk-manager.html

Google, 2017g. Suoraviite Powered by Choice. Haettu 25.01.2017
https://www.android.com/everyone/enabling-opportunity/

Telefinland, 2016, Valitako Android vai Apple? Haettu 27.2.2017
https://www.tele.fi/ostoksille/televinkit/artikkeli/android-vai-apple