Tampereen ammattikorkeakoulu

Android-sovelluksen kehitys ideasta julkaisuun

Lauri Kosonen

OPINNÄYTETYÖ
Toukokuu 2020

Tietojenkäsittelyn koulutus
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutus
Pelituotanto

KOSONEN, LAURI:
Android-sovelluksen kehitys ideasta julkaisuun

Opinnäytetyö 38 sivua, joista liitteitä 2 sivua
Toukokuu 2020

Tässä opinnäytetyössä tutustuttiin Android-mobiilisovelluskehitykseen ja toteutettiin yksinkertainen sovellusprojekti. Tavoitteena oli selvittää, kuinka Android-sovellusprojekti kulkee ideatasolta valmiiksi tuotteeksi, jonka käyttäjät ympäri maailmaa voivat ladata laitteillensa.

Opinnäytetyön tarkoituksena oli kertoa Android Studio -kehitysympäristön ominaisuuksista, yksinkertaisen sovellusprojektin eri vaiheista ja vaatimuksista sekä julkaisuprosessista Google Play -sovelluskaupassa. Työssä käytiin läpi tärkeimpiä Android-alustan ohjelmointirakenteita ja hyödynnettiin Git-pohjaista versionhallintaa.

Työn tuotoksena syntyi Title Generator -nimigeneraattorisovellus, joka julkaistiin Google Playssa. Projektissa käytettiin Android Studion monia eri ominaisuuksia sekä myös muita ohjelmistoja, kuten Notepad++-tekstieditoria ja Sourcetree-versionhallintaohjelmaa.

Projekti osoitti, että Android Studio on monipuolinen ohjelmisto mobiilisovelluksen kehittämisessä. Android Studio ei kuitenkaan riitä yksinään, sillä sovelluskehitys Android-alustalla vaatii joko Java- tai Kotlin-ohjelmointikielen tuntemusta. Lisäksi on otettava huomioon versionhallinta; se virtaviivaistaa tiedostojen varmuuskopioinnin ja jakamisen kehitystiimin kanssa sekä tekee ohjelmakoodiin tehtyjen muutosten seuraamisesta helppoa.

Sovelluksen julkaiseminen Google Play -kaupassa on vaivatonta. Google Play vaatii rekisteröitymisen ja siihen liittyvän kertaluontoisen maksun. Tämän jälkeen sovelluksia voi laittaa sinne julkaistavaksi niin monta kuin haluaa.

Oman mobiilisovelluksen luominen tuo monia etuja: sovellus voi auttaa eri tilanteissa, se kulkee aina mukana älypuhelimessa tai tabletilla, ja se on juuri sitä, mitä siitä teki. Google Play on täynnä erilaisia sovelluksia, jotka on suunniteltu ratkaisemaan eri ongelmia, mutta omaan tilanteeseen sopivan sovelluksen luominen on korvaamatonta.

Asiasanat: android, mobiilisovellus, java, google play

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Game Production

KOSONEN, LAURI:
Development and Release of an Android Application

Bachelor's thesis 38 pages, appendices 2 pages
May 2020

This thesis delved into the development of Android mobile application software and a simple app was created alongside it. The thesis' purpose was to study how an Android application is developed from an idea into a released product.

The thesis' goal was to cover the integrated development environment Android Studio and the release process on Google Play, the store for Android apps. Programming in the Android platform and using Git-based version control were important points of interest.

An app, Title Generator, was created to help illustrate the topics of the thesis. The project used many different features of Android Studio, as well as other software, such as Notepad++ for text editing and Sourcetree for version control. The finished product was released on Google Play.

The project showed that Android Studio is a versatile tool for creating Android application software. However, it alone is not enough as programming skills in Java or Kotlin are also needed. Version control is also very important, because it makes storing files, sharing them with the development team and tracking changes to the program code easier.

Releasing an app on Google Play is not difficult. The service requires registration and a one-time fee, but after that, it is possible to release an unlimited number of apps.

Creating a personal mobile application has many benefits: the app can be helpful in different situations, it can be accessed anywhere with a mobile device, and it does just what it was made to do. Google Play is filled with apps designed to solve different problems, but nothing can replace an app made suitable for a person's own life.

Key words: android, mobile application, java, google play

SISÄLLYS

1 JOHDANTO 6
2 ANDROID-MOBIILISOVELLUKSET 7
3 ALKUVALMISTELUT 8
3.1 Ohjelmistot ja laitteistot 8
3.2 Android Studio -projekti 9
4 KEHITYS 13
4.1 Sovelluksen ohjelmointi 13
4.2 Sovelluksen ajaminen 16
4.3 Käyttöliittymä 19
4.4 Activity-luokat 21
4.5 Tietokanta 22
5 PROJEKTIN KULKU 25
5.1 Versionhallinta 25
5.2 Viimeistely 28
5.3 Julkaisu 29
6 POHDINTA 33
LÄHTEET 35
LIITTEET 37
Liite 1. Typistetty XML-tietokanta 37

ERITYISSANASTO

activity	rajapinta Android-sovelluksen ja käyttäjän välillä
Android	mobiilikäyttöjärjestelmä
Android App Bundle	Google Playhin ladattava tiedosto, joka sisältää sovelluksen
APK	Android application package, Android-laitteelle ladattava sovelluspaketti
commit	tiedostojen muutos Git-versionhallinnassa
Git	versionhallintaohjelmisto
Java	ohjelmointikieli
kehitysympäristö	ohjelmisto, jolla voi kehittää sovelluksia
Kotlin	ohjelmointikieli
XML	Extensible Markup Language, tekstin rakenteen määrittelevä merkintäkieli

1 JOHDANTO

Tässä opinnäytetyössä tutkittiin, kuinka Android-käyttöjärjestelmässä toimiva mobiilisovellus luodaan. Jokaisella älypuhelimen tai tabletin omistajalla on kokemusta mobiilisovellusten käyttämisestä mutta ei välttämättä sellaisen tekemisestä. Sovelluskehitys vaatii ohjelmointitaitoa, mutta pelkästään kirjoittamalla ohjelmakoodia ei synny mobiilisovellusta.

Opinnäytetyössä otettiin huomioon koko projektin kaari eikä vain kehitysvaihetta. Tärkeitä osa-alueita olivat projektin aloittaminen, Android Studio -kehitysympäristön käyttäminen sovellusta kehitettäessä, versionhallinta ja sovelluksen julkaisu.

Mobiililaitteella käytettävän ohjelmiston luontiprosessin selvittämiseksi kehitettiin Title Generator -nimigeneraattorisovellus. Projektissa käytettiin Android Studiota ja tutustuttiin sen moniin eri puoliin. Versionhallinta toteutettiin käyttämällä Git-ohjelmistoa sekä sen kanssa toimivaa GitHub-palvelua. Projekti viimeisteltiin julkaisemalla sovellus Google Play -kaupassa ja jälkeenpäin sille annettiin päivitys.

Android-kehityksestä on olemassa paljon kirjallisuutta, video-oppaita sekä muita tietolähteitä. Tämä opinnäytetyö tiivistää tätä tietoa ja esittää sovelluskehityksen vaiheet alusta loppuun. Ensimmäisen mobiilisovelluksen kehittäminen voi olla hankalaa, mutta Title Generator -sovelluksen antamien esimerkkien avulla opinnäytetyö pyrkii havainnollistamaan käsiteltyjä aiheita ja madaltamaan kynnystä saada sovellus valmiiksi ja julkaistua.

Android-kehitys on aiheena laajempi kuin mitä opinnäytetyössä käsiteltiin, mutta opinnäytetyön painopisteenä olikin auttaa vasta-alkajaa luomaan ensimmäisen mobiilisovelluksensa. Tätä varten opinnäytetyö otti selvää Android Studion merkityksestä sovelluksen kehityksessä ja siitä, miten sovellus julkaistaan Google Playssa.

2 ANDROID-MOBIILISOVELLUKSET

Ohjelmistokehityksellä on monia muotoja. Erilaisia tietokoneohjelmia, verkkosivustoja sekä konsolipelejä on olemassa lukematon määrä, ja mobiilisovellukset eivät ole poikkeus. Kun aiheena on älypuhelin- tai tablettiohjelmistot, on kaksi varteenotettavaa alustaa: iOS ja Android (Friesen 2013, 8). Tämä opinnäytetyö käsittelee Androidia ja sille sovellusten kehittämistä.

Androidille on saatavilla miljoonia sovelluksia, joita lisätään jatkuvasti sovelluskauppa Google Playhin (Statista 2020). Sovellusten kehittäjät ja käyttäjät voivat olla kotoisin mistä vain ympäri maailmaa. Koska sovelluksia on olemassa niin paljon, niiden luomisen on oltava tarpeeksi yksinkertaista, ettei kehittäjän tarvitse olla alan asiantuntija.

Hyvä työkalu Android-sovellusten kehittämiseen on JetBrainsin IntelliJ IDEA -kehitysympäristöön perustuva Android Studio -kehitysympäristö (Android Developers n.d.c). Android Studio tarjoaa kattavasti toimintoja mobiiliohjelmistojen rakentamiseen, ja siksi se otettiin käyttöön Title Generator -sovelluksen kehityksessä. Kehityksessä valjastettiin Android Studion eri ominaisuuksia ja keinoja nopeuttaa luomisprosessia.

Vaikka Android Studio onkin erittäin suuri apu sovelluskehityksessä, se ei korvaa ohjelmointitaitoa. On idea kehitettäväksi sovellukseksi mikä hyvänsä, on osattava ohjelmoida, jotta sen saisi toteutettua. Tässä opinnäytetyössä ei pyritty opettamaan ohjelmoinnin perusteita vaan kertomaan Androidille ominaisista ohjelmointitekniikoista sekä Android Studion hyödyistä niin ohjelmointia kuin muitakin sovellusprojektin osa-alueita varten.

3 ALKUVALMISTELUT

3.1 Ohjelmistot ja laitteistot

Koko projektin ytimessä on kehitysympäristö Android Studio, jolla sovellus kehitetään ja pakataan Google Playhin julkaistavaksi Android App Bundle -tiedostoksi. Sovelluksen voi ladata sivulta https://developer.android.com/studio. Tämän lisäksi kehityksessä voidaan käyttää apuna Notepad++-ohjelmistoa, jonka avulla XML-tiedostojen muokkaaminen on virtaviivaisempaa. Notepad++:n voi ladata sivulta https://notepad-plus-plus.org/downloads/.

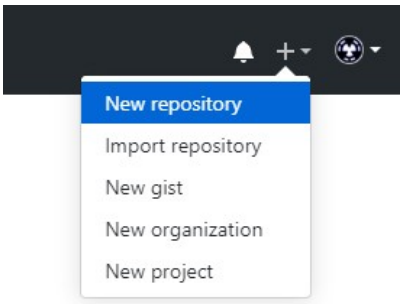
Versionhallintaa varten on saatavilla eri teknologioita ja työkaluja, mutta tähän opinnäytetyöhön valittiin Git, GitHub ja Sourcetree. Git on teknologia, johon käytettävä versionhallinta perustuu. GitHub on pilvipalvelu, joka säilöö projekteja ja pitää tallessa kaikki niiden versiot. Git voidaan ladata osoitteessa https://git-scm.com/ ja GitHub-palveluun voidaan luoda käyttäjätili osoitteessa https://github.com/. Sourcetree on versionhallintaa varten käytettävä graafinen käyttöliittymä, joka mahdollistaa Gitin paikallisen käyttämisen ilman komentorivisyötteitä. Se voidaan ladata sivulta https://www.sourcetreeapp.com/.

Sovelluksen julkaiseminen Google Play -kaupassa edellyttää Google Play kehittäjätilin luomista Google-tilin avulla (Play Console Ohjeet n.d.f). Se vaatii 25 Yhdysvaltain dollarin rekisteröitymismaksun ja jää ainoaksi maksulliseksi asiaksi tässä opinnäytetyössä käsitellyistä asioista mahdollisia laitteistohankintoja lukuun ottamatta.

Tarvittavat laitteistot ovat tietokone työskentelyyn ja mahdollisesti mobiililaite sovelluksen testaamista varten. Testaamiseen käy myös Android Studion mukana asennettava Android-emulaattori, vaikkakin oikean laitteen avulla saa paremman kuvan sovelluksen toiminnasta. Tässä opinnäytetyössä testilaitteena oli Huawei Honor 9 -älypuhelin ja käytetyn tietokoneen käyttöjärjestelmä oli Windows 10.

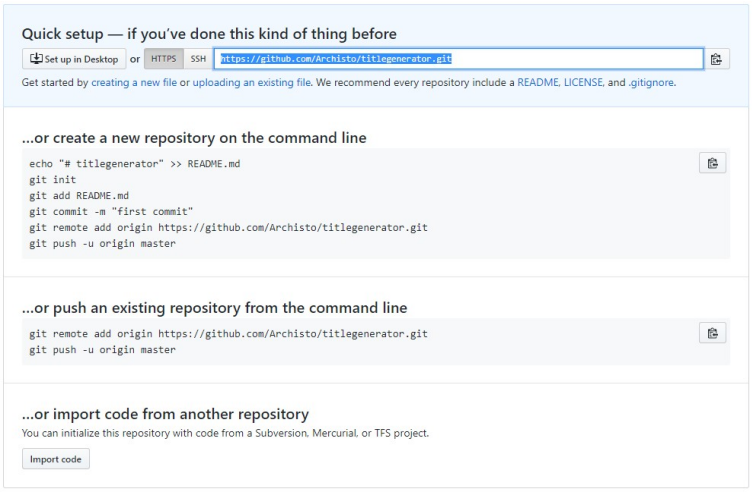
3.2 Android Studio -projekti

Aivan ensimmäiseksi on hyvä luoda tietokoneelle uusi hakemisto projektia varten. Tämä hakemisto määritetään käyttämään Git-versionhallintaohjelmistoa, jotta kaikki siihen laitetut tiedostot siirtyisivät GitHubiin. Tätä varten tarvitaan GitHub-käyttäjä ja säilytyspaikka (Repository) projektille (kuva 1). Palveluun kirjautumisen jälkeen uusi säilytyspaikka luodaan painamalla oikeassa yläkulmassa olevaa plusnappia ja valitsemalla avautuvasta valikosta Uusi säilytyspaikka (New repository).



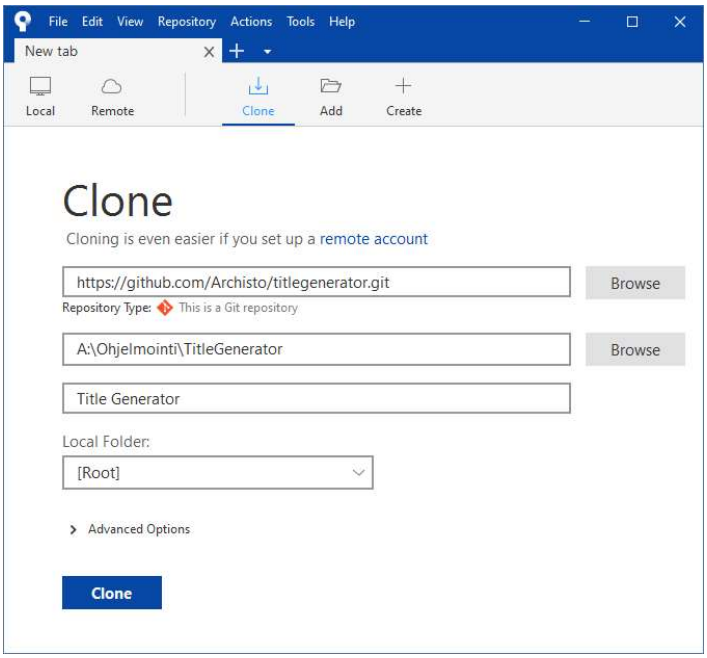
KUVA 1. Uusi GitHub-säilytyspaikka

Säilytyspaikalle annetaan projektia kuvaava nimi. Kuvauksen voi myös antaa, mutta se ei ole pakollista. Lopuksi painetaan Luo säilytyspaikka (Create repository) -nappia. Nyt avautuva ruutu tarjoaa Pika-alustus (Quick setup) -kohdassa uuden säilytyspaikan Git-osoitteen (kuva 2). Sourcetree-sovellus tarvitsee osoitetta yhdistääkseen paikallisen hakemiston ja GitHub-säilytyspaikan.



KUVA 2. GitHub-säilytyspaikan alustus, Git-osoite valittuna

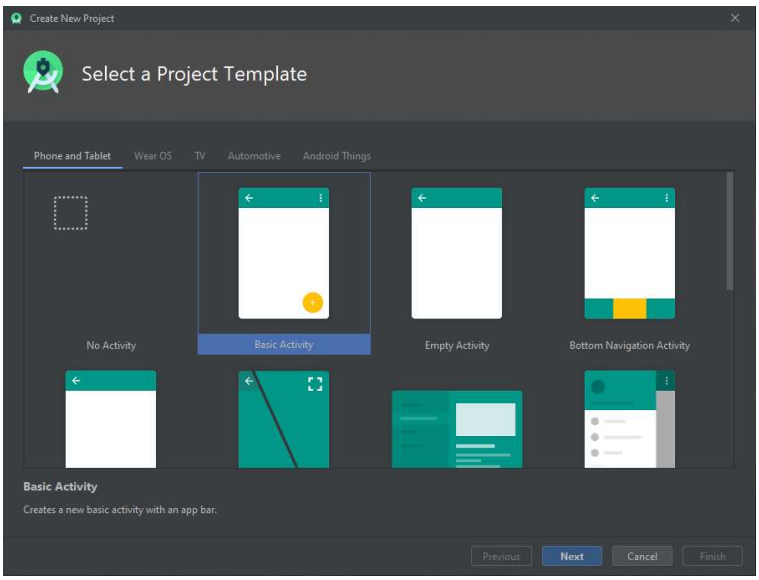
Ensimmäistä kertaa käytettäessä Sourcetree ei näytä mitään projektia. Projektin voi lisätä siihen painamalla Kloonaa (Clone) -painiketta ikkunan ylälaidassa. Nyt näkyviin tekstikenttiin (kuva 3) käyttäjä voi syöttää kloonattavan säilytyspaikan Git-osoitteen sekä projektin hakemiston ja nimen. Hakemiston on oltava tyhjä.



KUVA 3. GitHub-säilytyspaikan kloonaaminen Sourcetreessä

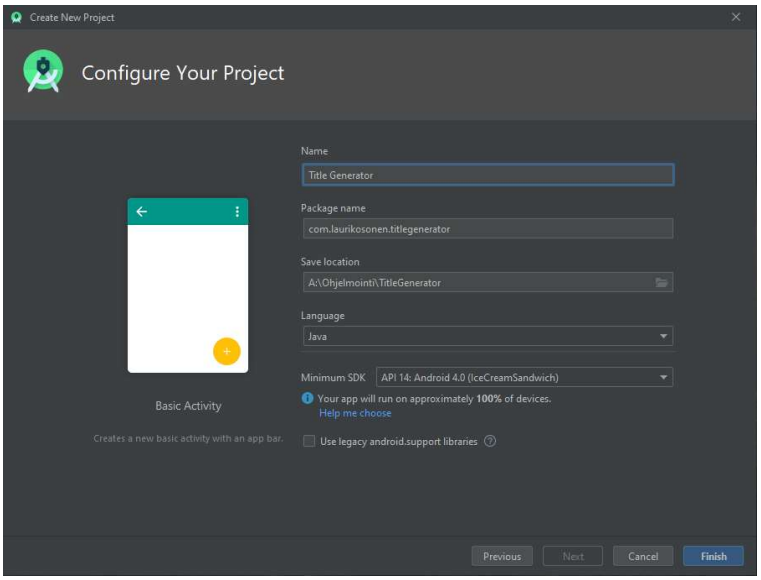
Versionhallinnan alustus viimeistellään painamalla alempaa, tummansinistä Kloonaa-painiketta. Sovellus pyytää GitHub-käyttäjän nimeä ja salasanaa, että saisi oikeuden päästä muokkaamaan projektin säilytyspaikkaa. Sisäänkirjautumistietojen antamisen jälkeen versionhallinta on alustettu. Git-alustettu projektihakemisto sisältää piilotetun .git-alihakemiston, Santacroce (2015, 18) kertoo. Alihakemisto suorittaa Gittiin liittyviä asioita, eikä käyttäjän tarvitse koskaan tehdä sille mitään.

Seuraavaksi alustetaan Android Studio -projekti. Kun Android Studio on asennettu ja käynnistetty, sovellus kysyy, minkälainen projekti luotaisiin (kuva 4). Pohjista kannattaa valita sellainen, joka sopii parhaiten suunniteltuun sovellukseen. Jokaiselle pohjalle voidaan luoda uusi projekti, jotta nähtäisiin, millaisia ne ovat käytännössä. Title Generator ei vaatinut muuta kuin tilaa tekstille ja generointinapin, joten ”Basic Activity” -vaihtoehto oli selvä valinta.



KUVA 4. Android Studion projektipohjat

Painamalla Seuraava (Next) -painiketta avautuu projektin esivalmisteluruutu (kuva 5). Tässä ruudussa valitaan projektin nimi, pakettinimi, tallennushakemisto, ohjelmointikieli sekä vähimmäis-SDK-versio.

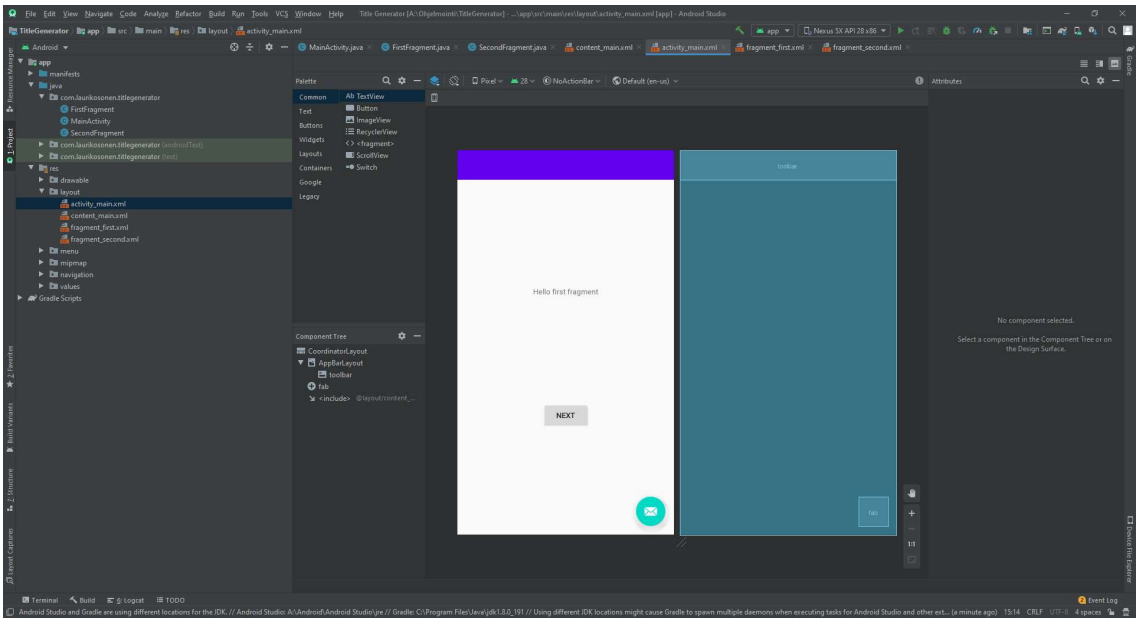


KUVA 5. Android Studio -projektin esivalmistelu

Pakettinimi eli projektin tunniste erottaa projektin muista sovelluksista laitteella ja Google Play -kaupassa. Pakettinimi kirjoitetaan käänteisesti eli organisaation domain-nimi lopusta alkuun, ylimmän tason domain – com, net, fi tai muu – ensimmäiseksi ja sen perään pisteillä erotettuna täsmentäjät. (Marsicano, Gardner, Phillips & Stewart 2019, 40). Esimerkiksi Title Generatorin pakettinimi on ”com.laurikosonen.titlegenerator”. Sovellukselle täytyy asettaa SDK (software

develoment kit) -versio, jotta Google Play voisi päätellä, mitkä Android-päivitysten mukana tulleet ominaisuudet toimivat sovelluksessa, ja jotta sovellus toimisi myös vanhemmilla laitteilla oikein (Android Developers n.d.d).

Valmis (Finish) -painike lataa ja avaa Android Studio -projektin (kuva 6). Oletustiedostot ja -ohjelmakoodi riippuvat valitusta projektipohjasta. Keskellä ruutua näkyy sovelluksen pääasiallinen, ja alussa ainoa, Activity. Activity, eli suomeksi ”toiminta”, hallitsee käyttäjän vuorovaikutusta sovellusnäkymän kanssa, Marsicano, Gardner, Phillips ja Stewart tiivistävät (2019, 35).



KUVA 6. Android Studio -näkymä

4 KEHITYS

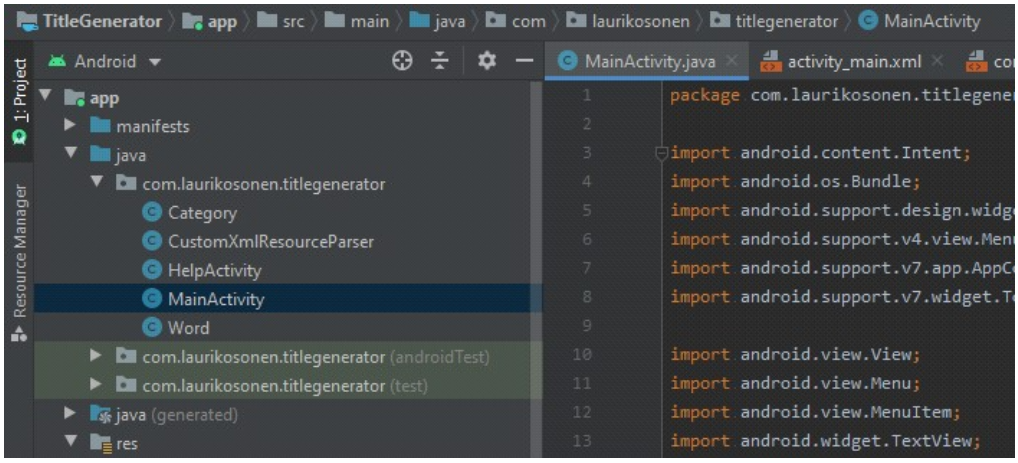
Android sisältää paljon tekniikoita, joilla voi luoda monia erilaisia sovelluksia. Ohjelmistokehitys vaatii aina ohjelmointitaitoa, ja Androidin mahdollisuuksien selvittämiseksi on katsottava, minkälaista Android-ohjelmointi on käytännössä. Kun sovellukseen on kehitetty toimintoja, on vuorossa sen ajaminen mobiililaitteella.

4.1 Sovelluksen ohjelmointi

Android-kehityksessä käytetään joko Java- tai Kotlin-ohjelmointikieltä. Marsicanon, Gardnerin, Phillipsin ja Stewartin (2019, 21) mukaan Kotlin on näistä uudempi ja suuremmassa suosiossa Android-kehittäjien keskuudessa kuin Java. Tässä opinnäytetyössä kuitenkin käsiteltiin vain Javaa, jolla myös Title Generator ohjelmoitiin.

Uutta projektia aloitettaessa Android Studio luo sovelluksen pohjaksi joitakin tiedostoja. Hagos (2019, 4) kertoo, että näistä tärkeimmät ovat activity-, asettelu- ja manifest-tiedosto. Activity on tarkoitettu ohjelmakoodille, asettelutiedosto määrittää sovelluksen ulkonäön ja manifest-tiedoston avulla projektin sisältö sidotaan yhdeksi paketiksi. Sovelluksen ydin ja alkupiste on MainActivity.java: kun käyttäjä laukaisee sovelluksen, hän näkee MainActivityn ensimmäiseksi. MainActivity voi olla myös muun niminen, jos projektin alustusvaiheessa valittiin niin.

Android Studiossa tiedostot voidaan avata joko ylälaidassa olevasta tiedostopalkista tai vasemmassa reunassa olevasta projektin hierarkiasta (kuva 7). MainActivity.java, sovelluksen pääkooditiedosto, sisältää jo tarvittavaa koodia sovelluksen suorittamiseen. Vaikka suuri osa sitä on Androidia varta vasten kirjoitettua koodia, se on aivan tavallista Javaa. Activity-luokkia muokattaessa huomiota kannattaa erityisesti kiinnittää onCreate-metodiin (kuva 8), jota kutsutaan, kun activity alkaa. Koska onCreate on ensimmäinen kutsuttu metodi Activity-luokassa, activityn alustukset tulee hoitaa siinä.



KUVA 7. MainActivity Android Studion projektihierarkiassa

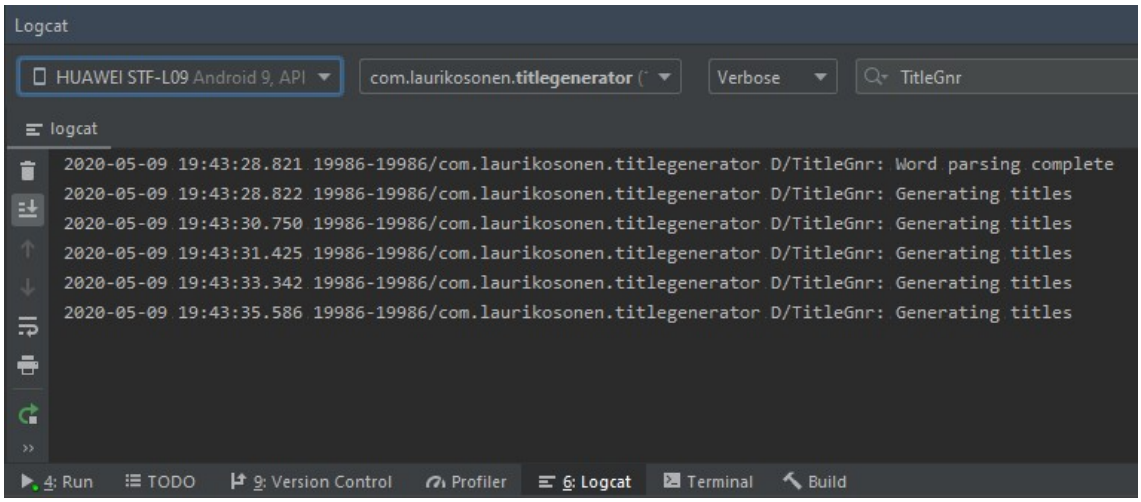
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    initWords();
    initDisplayedCategoryText();
    initTitleSlots();
    initCustomTemplate();
    initCustomTemplateToggle();

    generateTitles();

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener((view) -> { generateTitles(); });

    customTemplateToggle.setOnClickListener((view) -> {
            activateCustomTemplate(customTemplateToggle.isChecked());
    });
}
```

KUVA 8. Title Generatorin MainActivityn onCreate-metodi

Activity-tiedoston ei tarvitse sisältää kaikkea ohjelman koodia. Uusia luokkia voidaan luoda painamalla hiiren oikeaa painiketta hierarkiassa sovelluksen pakettinimen päällä ja valitsemalla Uusi (New) -valikosta Java-luokka (Java Class) -vaihtoehto. Uusi luokka toimii aivan samalla tavalla kuin Java-luokat yleensäkin. Activity- sekä muissa luokissa voidaan luoda siitä olioita ja kutsua sen metodeita.

Android-ohjelmoinnissa pätee samat tyylikäytännöt kuin muussakin ohjelmoinnissa. On tärkeää tuottaa puhdasta ja helposti luettavaa ohjelmakoodia, antaa hyviä nimiä metodeille ja muuttujille sekä kirjoittaa koodiin selventäviä kommentteja. Kehittäjät, jotka ovat tottuneet virheenkorjaukseen tulostamalla tekstiä

lokiin, käyttävät Android Studiossa Logcat-tekstilokia (kuva 9). Tekstiä saa tulostumaan lokiin kutsumalla Log.d(”avainsana”, ”tulostettava teksti”). Muitakin lokityyppejä on, kuten Log.e virhetilanteita ja Log.w varoituksia varten. Kirjoittamalla avainsanan Logcatin hakukenttään voi rajata näytettäviä tulosteita.



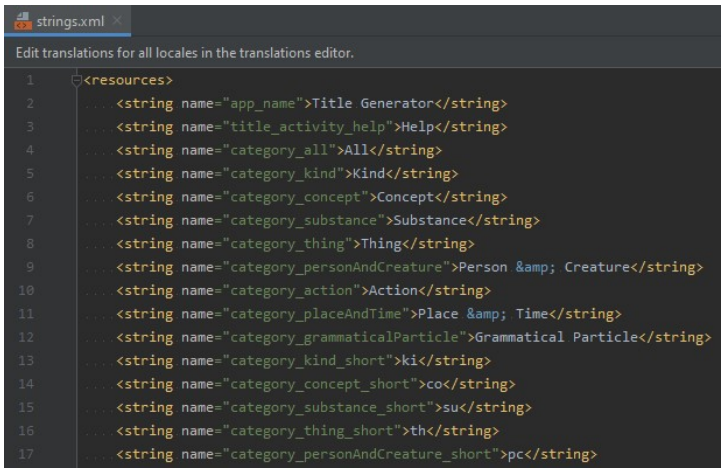
KUVA 9. Android Studion Logcat-tekstiloki

Ainutlaatuista mobiiliohjelmoinnissa on käyttäjän syöte, esimerkiksi laitteen kosketusnäytön antamat koordinaatit ja laitteen kallistuskulma. Android-elementit, kuten activity, käyttöliittymän osat sekä merkkijonot, ovat myös merkittäviä tekijöitä sovelluksessa. Käyttöliittymäelementit on johdettu View-luokasta, ja niillä on aina oma tunnisteensa. Activityssä voidaan päästä niihin käsiksi findViewById-metodin avulla: findViewById(R.id.elementin_tunniste).

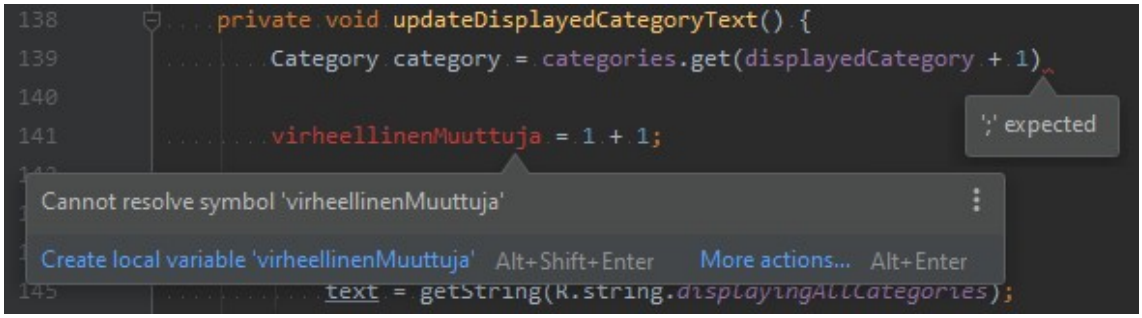
Merkkijonoille on oma tiedostonsa, strings.xml, joka sijaitsee res-hakemistossa. Res on lyhenne sanoista resources, resurssit. Merkkijonot haetaan R-luokan avulla lähes samalla tavoin kuin käyttöliittymäelementit: getString(R.string.merkkijonon_tunniste). Merkkijonot kannattaa pitää strings.xml-tiedostossa (kuva 10) yhdessä paikassa, jotta niitä voisi tarpeen tullen muuttaa nopeasti. Merkkijonojen käännöksiä on myös mahdollista hallita Android Studiossa, mutta Title Generator ei hyödyntänyt tätä ominaisuutta, sillä sen koko toimintaperiaate perustui vahvasti englannin kieleen.

Muiden kehitysympäristöjen tavoin Android Studio ilmoittaa ohjelmointivirheistä. Se huomauttaa joko suoraan koodista (kuva 11) tai yritettäessä ajaa sovellusta. Se voi myös antaa koodille parannusehdotuksia, kuten merkkijonon siirtäminen

strings.xml-tiedostoon. Jos sovellus sammuu yllättäen ajon aikana, Android Studio kertoo, missä tiedostossa, missä metodissa ja millä koodirivillä ohjelma kohtasi odottamattoman ongelman.



KUVA 10. Merkkijonoja strings.xml-tiedostossa



KUVA 11. Android Studio huomauttaa virheistä koodissa

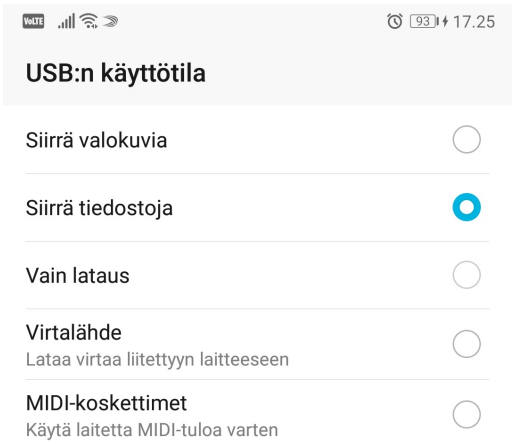
Android-kehitys voi olla aluksi hankalaa. Kun tutustuu lähemmin siihen, miten mobiiliohjelmointi toimii, ja kokeilee kaikkia Androidin tekniikoita, saa kehitettyä entistä monipuolisempia ja älykkäämpiä sovelluksia. Eikä myöskään voi unohtaa, että apua Androidin jokaiseen tekniikkaan saa helposti internetistä.

4.2 Sovelluksen ajaminen

Vaikka emulaattori on toimiva vaihtoehto sovelluksen ajamiseen, paras tapa siihen on käyttää oikeaa Android-mobiililaitetta. Tätä varten laitteen on oltava kehittäjätilassa. Laitteen järjestelmäasetuksissa on Tietoa puhelimesta -sivu, ja siellä napauttamalla Ohjelmistoversion numero -kohtaa seitsemän kertaa kehittäjätila kytkeytyy päälle. Näytölle ilmestyy viesti onnistumisesta. Kehittäjätilan

aktivoiminen mahdollistaa USB-vianetsintätilan ja sen avulla sovelluksen ajamisen laitteella USB-liitännän kautta. (Marsicano, Gardner, Phillips & Stewart 2019, 99.)

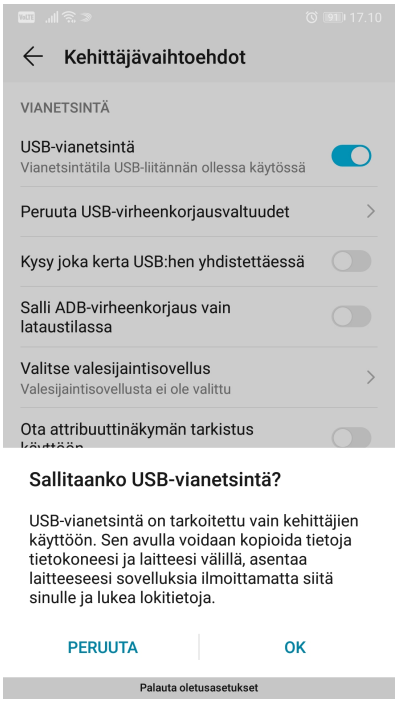
Kun laite on kytketty kaapelilla tietokoneeseen, laitteen pitäisi saman tien kysyä, mitä USB-yhteydellä tehdään. Sovelluskehitystä varten on valittava Siirrä tiedostoja -vaihtoehto. Jos vaihtoehdot eivät avaudu automaattisesti, tiedostojensiirron voi kytkeä päälle napauttamalla laitteen ilmoituksissa olevaa Lataus USB:n kautta -viestiä ja valitsemalla USB:n käyttötilaksi Siirrä tiedostoja (kuva 12).



KUVA 12. USB:n käyttötilavalikko Android-älypuhelimessa

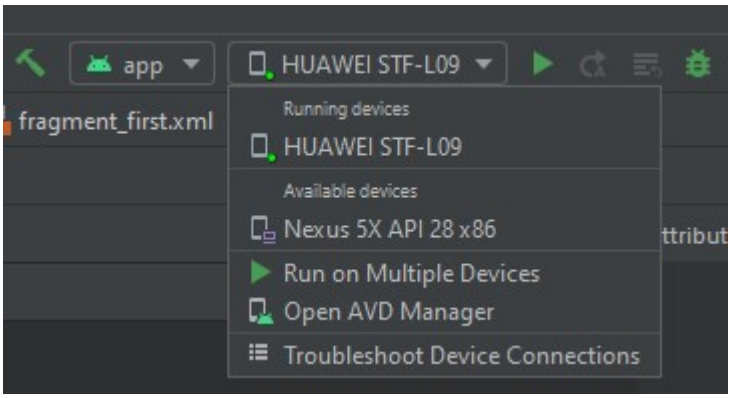
Kehittäjätilan myötä laitteen järjestelmäasetuksiin ilmestyy Kehittäjävaihtoehdot-sivu (kuva 13). Siellä voi kytkeä päälle vianetsintätilan, jonka avulla Android Studio osaa tunnistaa laitteen ja lähettää sovelluksen siihen.

Jos joku edellä mainituista vaiheista jää tekemättä, testaaminen laitteella ei ole mahdollista. On myös huomioitava, että oikea USB:n käyttötila on ehdottomasti valittava ennen USB-vianetsinnän aktivoimista. Muuten vianetsintä ei kytkeydy päälle, vaikka Kehittäjävaihtoehdot-valikon grafiikan mukaan se näyttäisikin siltä.



KUVA 13. USB-vianetsinnän kytkeminen päälle

Vianetsintätilan päälle kytkeytymisen ja sen myötä laitteen ja Android Studion välisen yhteyden toimivuuden voi varmistaa katsomalla Android Studion ylälaidassa olevaa laitevalikkoa (kuva 14). Käynnissä olevien laitteiden lista (Running devices) sisältää yhdistetyn laitteen tunnuksen, joka riippuu laitteen valmistajasta ja mallista. Saatavilla olevat laitteet (Available devices) -lista sisältää emulaattorin, jos sellainen on asennettuna. Tässä opinnäytetyössä käytetyn laitteen tunnus oli HUAWEI STF-L09 ja asennetun emulaattorin tunnus oli Nexus 5X API 28 x86.



KUVA 14. Android Studion tunnistamat testauslaitteet ja sovelluksen käynnistyspainike

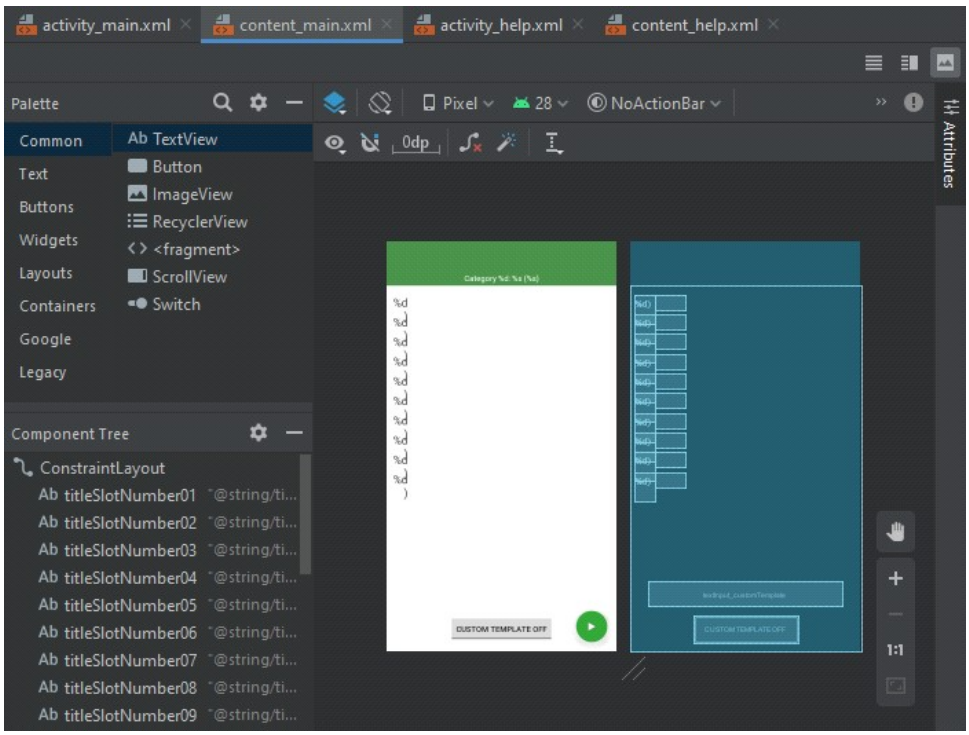
Sovellus ajetaan laitteella painamalla Android Studiossa vihreän nuolen näköistä painiketta laitelistan vieressä. Laite ei saa tällöin olla lukitustilassa. Kestää hetken verran, kun sovellus rakentuu ja tiedostot siirretään laitteelle. Jos sovelluksessa on ongelma, kuten ohjelmointivirhe, joka estää ohjelman toiminnan, Android Studio antaa virheilmoituksen. Jos ongelmia ei ilmene, sovellus käynnistyy. Mikäli sovelluksen koodiin tehdään ajon aikana muokkauksia, ne otetaan käyttöön vasta käynnistettäessä sovellus uudelleen.

4.3 Käyttöliittymä

Käyttöliittymä on ehdottoman tärkeä osa-alue niin mobiili- kuin minkä tahansa muunkin sovelluksen kehityksessä. Mitä käyttäjä voi tehdä ja nähdä, on käyttöliittymän sanelemaa. Androidilla käyttöliittymä luodaan XML-kielen avulla.

Android Studion käyttöliittymäeditorissa on mahdollista muokata käyttöliittymää sekä visuaalisesti että kirjoittamalla XML:ää suoraan (Marsicano, Gardner, Phillips & Stewart 2019, 44-45). Muokkaustilaa voi vaihtaa oikean yläkulman Koodi (Code), Jako (Split) ja Suunnittelu (Design) -painikkeilla. Käyttöliittymän todellinen ulkonäkö riippuu loppujen lopuksi siitä, miten sovellus on ohjelmoitu, eikä Android Studio voi muuta kuin näyttää suurpiirteisen mallin käyttöliittymästä. Kun esimerkiksi vertaa Title Generatorin pää-activityn käyttöliittymää Android Studiossa (kuva 15) sen todelliseen ulkonäköön laitteella (kuva 16), ero on huomattava.

Android-projektissa käyttöliittymätiedostot ovat res-hakemiston layout-alihakemistossa. Layout eli asettelu voidaan jakaa eri tiedostoihin, ja Title Generatorin tapauksessa nämä olivat toiminta (activity) ja sisältö (content). Asettelutiedostot nimetään yleensä niihin liitetyn activityn perusteella (Marsicano, Gardner, Phillips & Stewart 2019, 44). Tällöin esimerkiksi Title Generatorin MainActivityn, pää-activityn, asettelutiedostot olivat activity_main.xml ja content_main.xml.



KUVA 15. Title Generatorin käyttöliittymän muokkaus Design-tilassa Android Studiossa

KUVA 16. Sovelluksen pää-activityn todellinen ulkonäkö

Android-sovelluksen käyttöliittymässä voi olla erilaisia painikkeita, tekstikenttiä sekä muita elementtejä. Elementin sijainti riippuu sen koosta ja määritetystä etäisyydestä toiseen elementtiin. Elementti voidaan ankkuroida toiseen, jolloin sen paikka määräytyy toisen elementin paikan perusteella. Elementin tai tekstin

väriä voi hallita käyttämällä res-hakemiston values-alihakemiston colors.xml-tiedostossa määriteltyjä värejä, joita voidaan luoda tarvittaessa lisää.

Android tarjoaa helpon tavan luoda activityyn pudotusvalikko ruudun oikeaan yläkulmaan. Android Studio luo pohjan sille res-hakemiston menu-alihakemistoon. Kuten käyttöliittymän muitakin osia, pudotusvalikkoa voi muokata sekä visuaalisesti että XML:n kautta. Pudotusvalikon avulla voi antaa käyttäjän vaihtaa asetuksia tai päästä käsiksi toimintoihin, jotka muuten veisivät turhaan tilaa ruudulta. Käyttöliittymä määrää activityn ulkonäön ja käytettävät ominaisuudet, joten se on suunniteltava hyvin.

4.4 Activity-luokat

Activity on pääasiallinen rajapinta Android-sovelluksen ja käyttäjän välillä (Mew 2011, 28). Kaikki, mitä käyttäjä näkee ja tekee, tapahtuu activityn kautta; activity on siis kuin sivu verkkosivustolla. Activity-luokkia voi olla useita, ja ne voivat lähettää dataa toisilleen.

Activityn voi jakaa osiin Fragment-luokkien avulla. Fragment, ”palanen”, on sidoksissa vain yhteen Activity-luokkaan ja hallitsee tiettyä osaa sen toiminnoista tai käyttöliittymästä (Android Developers n.d.b). Activityyn voi liittää useamman Fragmentin tai se voi toimia itsenäänkin. Title Generator ei hyödyntänyt Fragmentteja vaan käytti activityihin ainoastaan Activity-luokkia.

Sovelluksen ohjelmakoodi kirjoitetaan Activity- tai muihin Java-luokkiin, ja sovelluksen käyttöliittymä on sidoksissa activityyn. Toisin sanoen activityt ovat sovelluksen ydin. Yksinkertaisemmat sovellukset tarvitsevat vain yhden activityn saadakseen kaikki toimintonsa käytettäväksi. Sovelluksessa voi myös olla useampi activity sen suunnittelusta ja sisällöstä riippuen. Esimerkiksi Title Generator sisältää kaksi activityä: nimigeneraatio- sekä apusivu.

Vaihtaminen kahden activityn välillä tarvitsee vain vähän koodia. Kuvan 17 osoittamalla tavalla voidaan käyttää startActivity-metodia, joka ottaa argumentiksi activity-vaihdoksen tiedon sisältävän Intent-olion, ”aikeen”. Title Generatoria varten luotiin goToHelp-metodi, jonka kutsuminen vaihtaa sovelluksen activi-

tyn pääohjelmasta apusivulle. Oletuksena Android-sovelluksissa voi palata edelliseen activityyn painamalla laitteen paluunappia, ja tätä käytettiin hyödyksi Title Generatorissa: käyttäjä pääsee siten apusivulta takaisin pääohjelmaan.

```
private void goToHelp() {
    Intent intent = new Intent( packageContext: MainActivity.this, HelpActivity.class);
    intent.putExtra( name: "customTemplate", customTemplateInput.getText().toString());
    startActivity(intent);
}
```

KUVA 17. Activityn vaihtaminen pääohjelmasta apusivulle Title Generatorin koodissa

4.5 Tietokanta

Jos sovellus tarvitsee suuren määrän dataa, sen säilyttämiseen kannattaa käyttää tietokantaa. Tietokanta voi sisältää tekstimuodossa mitä tahansa tietoa. Tässä opinnäytetyössä rakennettiin Title Generator -sovellusta varten paikallinen XML-tietokanta, jonka tehtävä oli säilyttää kaikki nimigeneraation käyttämät sanat. Typistetty versio siitä on opinnäytetyön liitteissä (liite 1).

XML on tekstin rakenteen ja tietosisällön määrittelevä merkintäkieli, ja sen nimi on muodostettu sanoista Extensible Markup Language, ”laajennettava merkintäkieli”. Tekstidokumentin jakamiseksi eri osiin ja näiden osien tunnistamiseksi käytetään tageja, ”nimilappuja”. Tagit eivät ole ennalta määritettyjä, vaan ne päätetään itse. (Harold 2004, 13.)

Kuvassa 18 on esimerkki XML:n syntaksista. Harold (2004, 24) selittää, että kulmasulkeilla ympäröidyt tagit aloittavat ja lopettavat elementtejä, jotka voivat sisältää attribuutteja, tekstiä tai sisempiä elementtejä. Attribuutin muodostaa nimi ja arvo yhtäläisyysmerkillä erotettuna. Tagit päätetään niin, että ne kertovat hyvin niiden sisältämän tiedon tarkoituksen (Harold 2004, 25), ja sama pätee attribuuttien nimiin. Esimerkissä punaisella värillä merkityt sanat ovat attribuuttien nimiä ja violetilla värillä merkityt ovat niiden arvoja.

```
<?xml version="1.0" encoding="utf-8"?>

<elementit>
    <elementti attribuutti="A" />
    <elementti attribuutti="B" />
    <elementti attribuutti="C" attribuutti2="Toinen attribuutti">
        <sisempiElementti>Sisemmän elementin teksti.</sisempiElementti>
    </elementti>
</elementit>

<!-- Tämä on kommentti. -->
```

KUVA 18. Esimerkki XML:n syntaksista

Ensimmäinen rivi esimerkissä on XML-tiedostossa pakollinen; se määrittelee XML:n version ja tekstin formaatin. Mikään muu XML-tiedoston sisällössä ei ole määrättyä. Elementtien, sisäkkäisten elementtien ja attribuuttien lukumäärällä ei ole rajaa, mutta samalla elementillä ei voi olla kahta tai useampaa samannimistä attribuuttia ja lopputagin on vastattava alkutagia. Jos elementillä ei ole sisempiä elementtejä tai tekstiä, se voidaan tiivistää yhteen tagiin korvaamalla lopputagi vinoviivalla tagin lopussa.

XML on ihanteellinen tietokannan luomista varten, koska sen avulla suurikin määrä tietoa voidaan pitää järjestyksessä helposti luettavassa muodossa (Harold 2004, 14). Tästä syystä Title Generatorin sanakirja kirjoitettiin XML-kielellä. Sanakirja koostui elementtien avulla kahdeksasta kategoriasta ja tuhansista sanoista. Sanojen eri muodot toteutettiin attribuuteilla.

Title Generatorin tietokanta suunniteltiin toimimaan yhdessä Java-koodin kanssa. Tietokantaa varten kirjoitettiin sitä lukeva luokka, joka poimi sanat sekä niiden eri muodot ja asetti ne listaan. Kuvassa 19 on esimerkki DatabaseParser-nimisestä Java-luokasta, joka ottaa vastaan tietokannan resurssitunnisteen, lukee tietokannan sisällön ja tulostaa sen Logcatiin. Kun example-niminen esimerkkitietokanta on res-hakemistoon lisätyssä xml-alihakemistossa, tietokannan lukijaa kutsutaan Activity-luokasta komennolla DatabaseParser.parseDatabase(getResources(), R.xml.example). Komennon jälkimmäinen argumentti on tietokannan resurssitunniste, jonka muoto on R.res_alihakemisto.tietokannan_nimi.

Lukija eli parseri iteroi tietokannan jokaisen rivin läpi tekstidokumentin loppuun asti. Riviä vaihdetaan parserin next-metodilla. Jotta lukija tietäisi, minkä tyyppistä XML-ominaisuutta se milläkin hetkellä lukee, on otettava ylös tapahtumatyyppi sen getEventType-metodin avulla. Vertaamalla tätä kokonaislukua XmlPullParserin sisältämään START_TAG-, END_TAG- ja TEXT-arvoon voidaan selvittää, onko kyseessä alkutagi, lopputagi vai teksti. Attribuutin arvo saadaan parserin getAttributeValue-metodilla, joka ottaa attribuutin nimen argumenttina.

Android Studio soveltuu hyvin XML:n kirjoittamiseen ja jopa ilmoittaa mahdollista virheistä kuten puuttuvista lopputageista. Title Generatorin kehityksessä tietokannan muokkamiseen käytettiin kuitenkin pääasiassa Notepad++-ohjelmistoa, sillä se käynnistyy nopeasti, mahdollistaa rivien järjestämisen aakkosjärjestykseen sekä tarjoaa tehokkaamman haku- ja korvaustoiminnon kuin Android Studio. Kummallakin ohjelmistolla on vahvuutensa, ja kehittäjän oma mieltymys ratkaisee, kumpaa käyttää tietokannan luomisessa.

```
import android.content.res.Resources;
import android.content.res.XmlResourceParser;
import android.util.Log;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import java.io.IOException;

class DatabaseParser {
    static void parseDatabase(Resources resources, int resourceID) {
        XmlResourceParser parser = resources.getXml(resourceID);

        try {
            parser.next();
            int eventType = parser.getEventType();

            while (eventType != XmlPullParser.END_DOCUMENT) {
                if (eventType == XmlPullParser.START_TAG) {
                    Log.d( tag: "databaseExample",  msg: "Start tag: " + parser.getName());
                    Log.d( tag: "databaseExample",  msg: "Attribute: " +
                            parser.getAttributeValue( namespace: null,  name: "attribuutti"));
                    Log.d( tag: "databaseExample",  msg: "Attribute 2: " +
                            parser.getAttributeValue( namespace: null,  name: "attribuutti2"));
                }
                else if (eventType == XmlPullParser.TEXT) {
                    Log.d( tag: "databaseExample",  msg: "Text: " + parser.getText());
                }
                else if (eventType == XmlPullParser.END_TAG) {
                    Log.d( tag: "databaseExample",  msg: "End tag: " + parser.getName());
                }

                eventType = parser.next();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        catch (XmlPullParserException e) {
            e.printStackTrace();
        }
    }
}
```

```
Activity DatabaseParser.parseDatabase(getResources(), R.xml.example);
```

KUVA 19. Esimerkkitietokantaa lukeva Java-luokka ja lukijaa kutsuva komento

5 PROJEKTIN KULKU

Sovelluskehitys ei ole vain ohjelmoimista. On myös tärkeää tietää, kuinka ohjelmistoprojekti etenee ja mitä tehdään, kun ohjelmisto saavuttaa version 1,0. On mahdollista jakaa ohjelmisto vain omalle lähipiirilleen, mutta projektin ei tarvitse päättyä siihen.

5.1 Versionhallinta

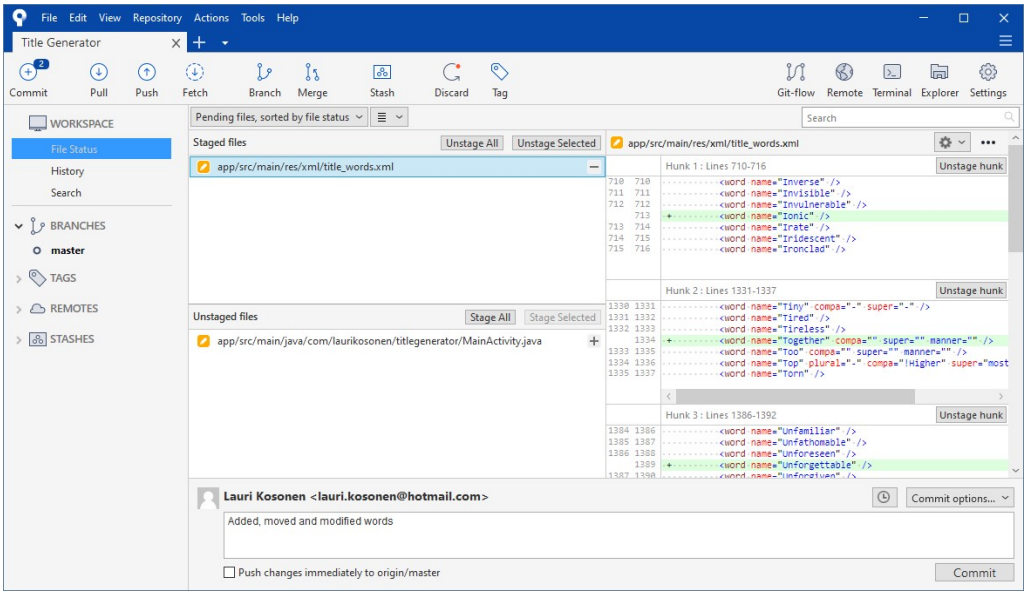
Santacrocen (2015, 17) mukaan versionhallinnan avulla työ helpottuu ja projektin edistyminen on kitkattomampaa. Projektin koosta riippumatta on aina kannattavaa käyttää jotakin versionhallintatyökalua. Tällaisista työkaluista käytetyimmäksi on noussut avoimen lähdekoodin ohjelmisto Git.

Git-säilytyspaikka koostuu järjestetystä ketjusta committeja. ”Commit” englannin kielessä tarkoittaa ”sitoutua”. Commitit sisältävät tietoa ohjelmakoodi- sekä muihin tiedostoihin tehdyistä muutoksista. Committia luotaessa sille annetaan kuvaus, joka kertoo, minkälaisen ominaisuuden lisäyksen, virheenkorjauksen tai poiston se sisältää. (Santacroce 2015, 22.) On hyvä käytäntö pitää commit yhtenä kokonaisuutena, eli siinä tulisi olla vain yksi valmiiksi saatu uusi ominaisuus tai korjaus. Jos commit aiheuttaa ongelman sovelluksen toiminnassa, se on näin helpompi löytää ja korjata.

Jos versionhallinta on oikein alustettu, kaikki projektin muokatut, siihen lisätyt sekä siitä poistetut tiedostot näkyvät Sourcetreen tiedostostatus (File Status) -välilehdellä. Commit luodaan kuvan 20 osoittamalla tavalla. Lisätyt, muutetut ja poistetut tiedostot asetetaan Staged files -kohtaan käyttämällä Aseta kaikki (Stage All) tai Aseta valitut (Stage Selected) -painiketta, jonka jälkeen kirjoitetaan kuvaava viesti ikkunan alalaidan tekstilaatikkoon. Commit hyväksytään painamalla Commit-painiketta.

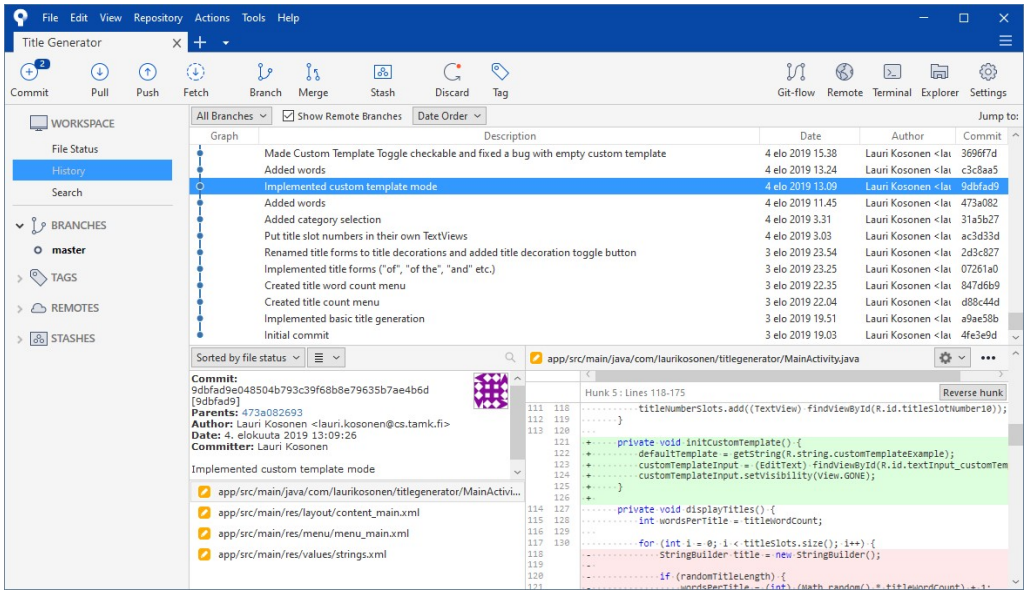
Muutokset voi perua painamalla tiedoston nimeä hiiren oikealla painikkeella ja valitsemalla Poista (Discard) -vaihtoehto. Toinen poistovaihtoehto, Remove, poistaa koko tiedoston. Lisätyt, muutetut ja poistetut tiedostot voidaan erottaa

toisistaan niiden nimen vasemmalla puolella olevan kuvakkeen perusteella. Muun muassa vihreä tarkoittaa lisättyä, keltainen muokattua ja punainen poistettua tiedostoa.



KUVA 20. Tiedostostatusnäkymä Sourcetree-ohjelmassa ja commitin luonti

Kaikki luodut commitit ovat Sourcetreen historiavälilehdellä järjestyksessä (kuva 21). Siellä käyttäjä voi tarkastaa aikajanalla kaikki luodut commitit sekä niiden tekijät ja luontipäivämäärät. Valitsemalla commitin voi nähdä listan tiedostoista, joihin se vaikuttaa. Listasta voi valita tiedoston, ja tällöin Sourcetree korostaa commitissa tiedostoon lisätyt rivit vihreällä ja siitä poistetut rivit punaisella.



KUVA 21. Commit-historia Sourcetree-ohjelmassa

Commitit saa talletettua GitHubiin painamalla Sourcetreen ylälaidan työkalunauhassa olevaa Puske (Push) -painiketta, valitsemalla avautuvasta versiohaara (Branch) -ikkunasta se, johon commit kuuluu, ja hyväksymällä pusku.

Oletuksena projektissa on vain yksi versiohaara, master, jolla samanniminen osoitin merkitsee haaran uusimman commitin, Laster (2016, 16) selittää. Toisin sanoen versiohaara pitää kirjaa siitä, mikä on viimeaikaisin päivitys, ja tällöin versiohaara pysyy ajankohtaisena. Lasterin mukaan versiohaarojen avulla projektin tiedostojen tietyt versiot voidaan pitää yhdessä paikassa saman nimen alla. Versiohaaran mistä tahansa commitista voidaan aloittaa uusi haara painamalla Sourcetreen työkalunauhan Branch-painiketta, ja aikajanalla voidaan tarkastella näitä kaikkia.

Jos projektitiimissä on kaksi jäsentä tai useampi, ohjelmakoodi sekä muut tiedostot on jaettava kaikkien kesken. Tätä varten Sourcetreessä on Vedä (Pull) -painike. Sen avulla käyttäjä saa ladattua kaikkien GitHubiin puskettujen versiohaarojen kaikki commitit omalle tietokoneellensa.

Jokaista projektin tiedostoa ei tarvitse säilöä versionhallinnan avulla. Git voidaan asettaa jättämään pois .gitignore-tiedostossa määritellyt hakemistot tai tiedostot. Tämä projektikansion juuressa oleva tiedosto, jolla ei ole muuta nimeä kuin sen tiedostopääte, on hyödyllinen epäolennaisten tiedostojen pitämisessä pois versionhallinnasta. Tiedostot joko eivät vaikuta sovelluksen toimintaan tai ne ovat tarkoitettu vain Android Studion paikalliseen käyttöön. Kuvassa 22 on esitelty Title Generatorin .gitignore-tiedoston sisältö.

```
1  *.iml
2  .gradle
3  /local.properties
4  /.idea/caches
5  /.idea/libraries
6  /.idea/modules.xml
7  /.idea/workspace.xml
8  /.idea/navEditor.xml
9  /.idea/assetWizardSettings.xml
10 .DS_Store
11 /build
12 /captures
13 app/release/
14 .externalNativeBuild
```

KUVA 22. Title Generator -projektin .gitignore-tiedosto

Tiivistettynä projektin rytmi on siis seuraavanlainen: Tiimin jäsen tekee muokkauksen työn alla olevaan sovellukseen, muodostaa muokkauksesta commitin ja puskee commitin GitHubiin. Tämän jälkeen muut tiimin jäsenet vetävät commitin päivittääkseen sovelluksen omilla tietokoneillaan. Kierto on jatkuvaa, ja yhteistyö on kitkatonta.

5.2 Viimeistely

Jokaisella sovelluksella täytyy olla nimi. Oletuksena sovelluksen pää-activityn ylälaidassa oleva nimi noudattaa Android Studio -projektin alussa sovellukselle annettua nimeä. Jos nimeä haluaa muuttaa, se on mahdollista strings.xml-tiedostossa eli samassa paikassa kuin kaikki sovelluksen muutkin tekstit. Merkkijono tunnuksella ”app_name” määrittää sovelluksen nimen.

Sovelluksen nimen muuttaminen voi tarkoittaa sitä, että sen pakettinimi ei enää täsmää. Sitäkin voi muuttaa nimeämällä se uudestaan projektin hierarkiassa. Android Studio antaa tällöin varoituksen ja kertoo, mitä tulee tapahtumaan, jos nimenmuutosta jatkaa. Pakettinimen tarkoitus on erottaa sovellus muista, joten pakettinimen vaihtamisen jälkeen testilaite pitää sovellusta uutena, eikä se siis korvaa vanhaa versiota. Vanha versio on tällöin poistettava laitteelta käsin.

Nimen lisäksi tärkeä osa jokaista sovellusta on sen kuvake. Kuvake on kuin sovelluksen kasvot: se erottaa sovelluksen muista ja on ensimmäinen asia, jonka perusteella mahdolliset käyttäjät muodostavat siitä mielipiteen sovelluskaupassa. Lisäksi kuvaketta napauttamalla sovellus käynnistyy laitteella. Android Studiossa kuvakkeen voi asettaa manifests-hakemistossa olevassa AndroidManifest.xml-tiedostossa. Olennaiset kohdat siellä ovat android:icon sekä android:roundIcon.

Myös kuvakkeen luomista varten Android Studiossa on työkalu. Työkalu avautuu luotaessa uutta kuvaelementtiä (Image Asset). Paikka kuvaelementeille on res-hakemiston alla oleva mipmap-alihakemisto. Kuvakkeen luontityökalu avataan painamalla mipmapia hiiren oikealla painikkeella, avaamalla valikosta Uusi (New) -alivalikko ja sen alta valitsemalla Image Asset. Työkalu antaa mahdolli-

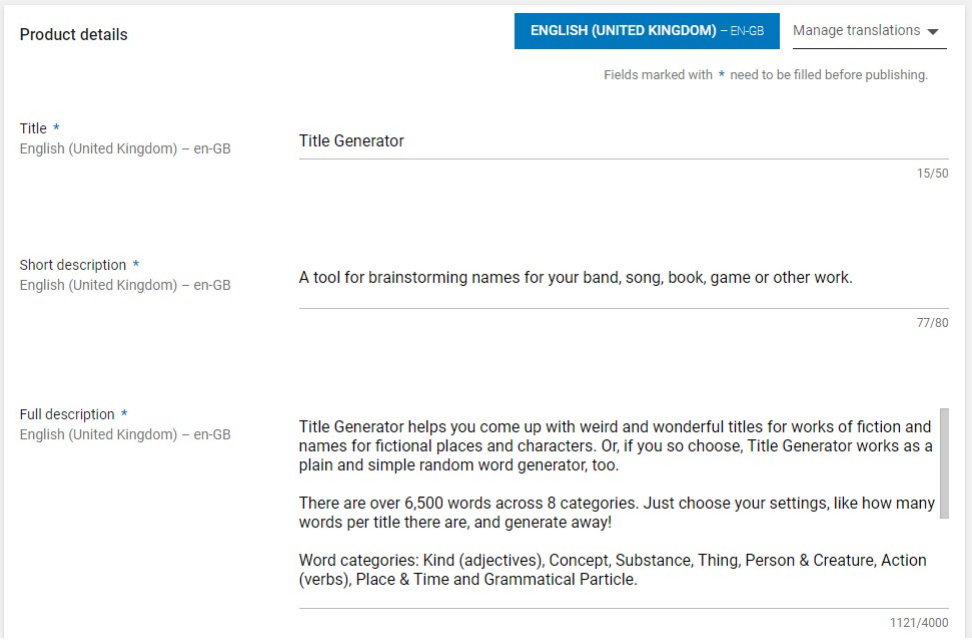
suuden hallita kuvakkeen nimeä, edustaa, taustaa, muotoja ja värejä. Kun kuvake on valmis, sen nimi asetetaan AndroidManifest.xml-tiedoston kuvakekohtiin muodossa ”@mipmap/kuvakkeen_nimi”.

Sovellus vaatii testausta ennen kuin sen voi sanoa olevan valmis. Testaaminen yhdellä laitteella varmistaa sen, että ohjelma sisältää vaaditut ominaisuudet, mutta sen toimivuus eri laitteilla ei ole itsestään selvää. Jos testauslaitteita ei ole kuin yksi, voi esimerkiksi antaa perheen tai ystävien ladata sovelluksen, jos heillä on sopiva Android-laite. Tätä varten on tehtävä sovelluksesta jaettavaksi sopiva Android-sovelluspaketti (Android application package, APK). Sovelluspaketti luodaan valitsemalla Android Studion ylälaidan Rakenna (Build) -valikosta Build Bundle(s) / APK(s) -vaihtoehto ja sen alta Build APK(s).

Sovelluspaketin rakentumisaika riippuu sovelluksen koosta. Kun APK on luotu, Android Studio näyttää viestin rakentamisen onnistumisesta. Käyttäjä voi valita viestistä paikanna (locate) -vaihtoehdon nähdäkseen APK:n hakemiston. Tiedoston nimi on oletuksena app-debug.apk, mutta sen voi muuttaa sopivammaksi. Kun tämän tiedoston lataa Android-laitteelle, sovelluksen voi asentaa sille ilman sen kytkemistä Android Studioon tai sovelluksen julkaisemista Google Playssa.

5.3 Julkaisu

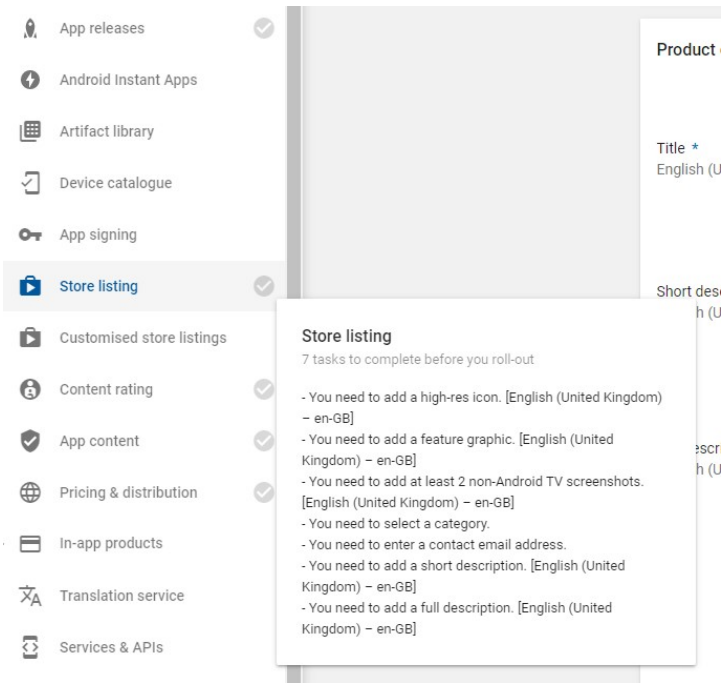
Jos sovellus on valmis suuren yleisön käyttöön, on aika julkaista se. Google Play -kaupan kehittäjäpuoli, Google Play Console, on paikka tälle. Rekisteröitymisen ja siihen liittyvien toimien jälkeen ensimmäinen sovellus lisätään Google Play Consoleen painamalla Kaikki sovellukset (All applications) -välilehdeltä Luo sovellus (Create application) -painiketta. Avautuvassa ikkunassa annetaan sovellukselle sen oletuskieli ja nimi. Tämän jälkeen aukeaa sovelluksen tietosivu (kuva 23). (Play Console Ohjeet, n.d.h.)



KUVA 23. Julkaistavan sovelluksen tietosivun muokkaaminen Google Play Consolessa

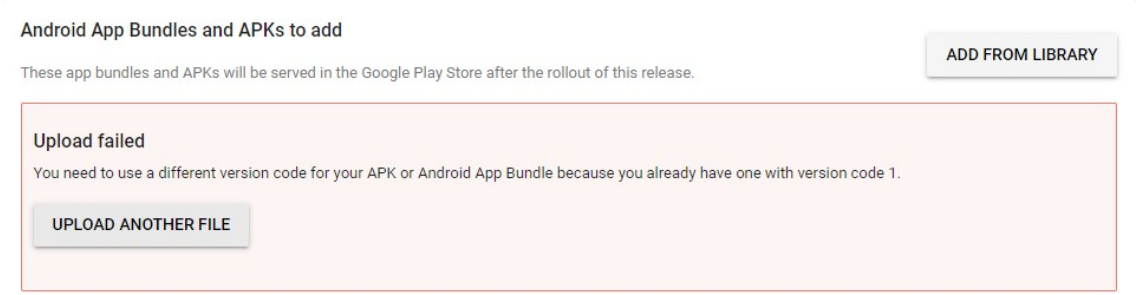
Tietojen, kuten kuvauksen ja kuvakaappausten, lisäämisen jälkeen on vielä muutama muu vaihe ennen kuin sovellus julkaistaan (kuva 24). Yksi näistä on itse sovelluksen lähettäminen Android App Bundle -tiedoston muodossa. Bundle rakennetaan Android Studiossa Build-valikon kautta, samassa paikassa kuin sovelluspaketti. Bundle, joka myös tarkoittaa suomeksi ”pakettia”, on Google Playn suosima julkaisuformaatti (Play Console Ohjeet, n.d.h).

Julkaistava sovellus pysyy luonnostilassa, kunnes kehittäjä on suorittanut vaaditut tehtävät (Play Console Ohjeet n.d.g). Tämän jälkeen sovellus on valmis julkaistavaksi -tilassa ja odottaa, että kehittäjä käynnistää julkaisun. Julkaisutoimenpiteiden jälkeen alkaa sovelluksen tarkistusprosessi. Tarkistus ottaa huomioon sovellukselle asetetun ikärajan sekä muut annetut tiedot, ja se voi kestää viikon tai kauemmin. Prosessin päätyttyä sovelluksen kehittäjä saa sähköpostin lopputuloksesta. Jos sovellus arvioidaan käyttäjille turvalliseksi, sovellus on julkaistu ja ladattavissa Google Playsta.



KUVA 24. Play Console näyttää sovelluksen julkaisua varten vaadittavat tehtävät

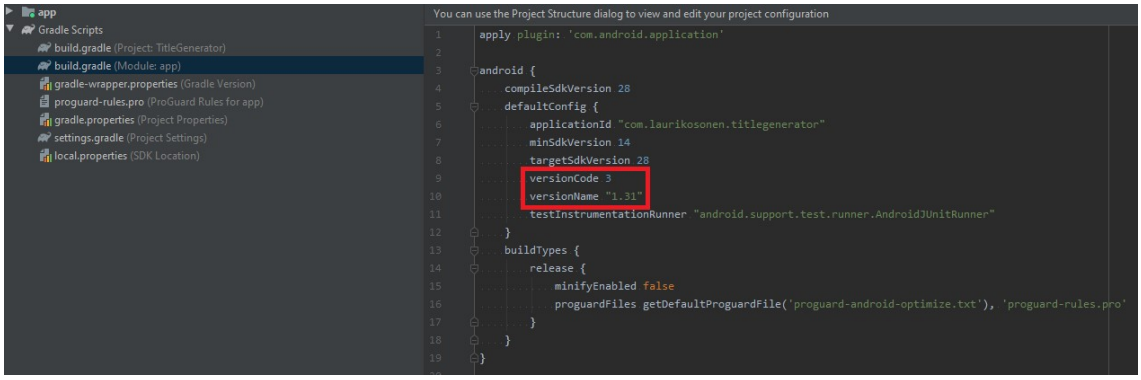
Päivitysten tekeminen on myös tärkeä osa ohjelmistokehitystä, ja nekin on julkaistava, jotta kaikki käyttäjät voisivat nauttia niistä. Ei riitä, että vain tekee ohjelmistoon muokkauksia, luo uuden Android App Bundlen ja lähettää sen Play Consoleen, sillä sen versiokoodi on jo käytetty. Play Console antaa tästä virheilmoituksen (kuva 25).



KUVA 25. Play Console ilmoittaa lähetetyn Android App Bundlen virheellisestä versionumerosta

Android-sovelluksen versiomuutosta varten on muokattava build.gradle-tiedostossa (kuva 26) kahta arvoa: versionCode ja versionName. Pelkästään positiivisen kokonaisluvun hyväksyvä versiokoodi määrittää sen, mikä versio on uudempi, kun kahta vertaillaan. Merkkijonomuotoinen versionimi puolestaan on

tarkoitettu käyttäjien nähtäväksi. (Android Developers, n.d.e). Versionimi on näkyvillä esimerkiksi sovelluksen Google Play -sivun alalaidassa.



KUVA 26. Sovelluksen build.gradle-tiedosto Android Studiossa versiomuuttujat korostettuna

Kun versiokoodi ja -nimi ovat kunnossa, päivityksen julkaisu onnistuu. Samalla kannattaa varmistaa, että kauppasivun kuvaus ja kuvakaappaukset ovat ajankohtaisia. Sovellus käy läpi uuden tarkistusprosessin, ja jos ongelmia ei ole, päivitetty versio on pian käyttäjien ladattavissa.

6 POHDINTA

Android on hyvä alusta mobiilisovellusten luomiseen. Alkuun pääsemiseen tarvitaan vain Android-laite, tietokone ja Android Studio. Lisäksi on osattava Java- tai Kotlin-ohjelmointikieltä. Android-kehitys on laaja ja monimutkainen aihe, mutta yksinkertaisen sovelluksen saa aikaiseksi rajallisellakin osaamisella. Kehitystyössä käytettyjen teknologioiden omaksuminen voi viedä aikaa, mutta apua jokaiseen tässä opinnäytetyössä käsiteltyyn aiheeseen on helposti löydettävissä internetistä.

Suureen kehitysbudjettiin ei ole tarvetta, jos haluaa kehittää ja julkaista Android-sovelluksen; laitteiden kertahankinnat ovat suurimmat menoerät. Google Play vaatii rekisteröitymismaksun, mutta 25 dollaria on pieni hinta siitä, että saa julkaista rajattoman määrän sovelluksiaan kaupassa.

Vaikka omilla sovelluksillaan ei tienaisikaan rahaa, on niiden kehittäminen rikastuttava kokemus. Ohjelmointi on arvokas kyky, jonka kehittäminen ei koskaan ole haitaksi. Se parantaa ongelmanratkaisutaitoa, loogista päättelykykyä ja teknistä osaamista, ja jos vie ohjelmistoprojektin loppuun asti, on palkintona valmis sovellus. Jos mielii ohjelmistokehitysuralle, ohjelmointitaito ja luodut sovellukset ovat erittäin merkittäviä. Omien ohjelmistojen voidaan myös ajatella olevan kuin taideteoksia: kuvataiteilija maalaa taulun ja ohjelmoija luo sovelluksen.

Sovelluskehitys tuo valtaa ja vapautta: ohjelmistot ovat aina sellaisia, kuin niistä on itse tehnyt. Omat mobiilisovellukset ovat käytettävissä missä ja milloin vain, ja ne voivat olla hyödyllisiä omassa tai lähipiirin käytössä. Ne voivat olla silkkaa viihdettä tai auttaa arjen askareissa, harrastuksissa tai työpaikalla.

Ohjelmiston julkaiseminen sovelluskaupassa on viimeinen merkittävä vaihe projektissa. Siten sen saa koko maailman ladattavaksi ja nautittavaksi. On erittäin mahdollista, että kaupassa on jo tarjolla samankaltainen sovellus tai useampi, mutta sovellus voidaan huomata siitä huolimatta.

Google Play hyötyy siitä, että yhä enemmän ja enemmän ihmisiä julkaisee sovelluksiaan siellä. Tämä hyvä syy sille, miksi Android-sovellusten tekemisestä ja julkaisemisesta on tehty niin yksinkertaista. Google ottaa joka sovelluksen ja sen sisältämien ostosten myynnistä osuuden itselleen. Sovellusten näyttämät mainokset tuottavat myös merkittävää tuloa. Kehittäjän ei kuitenkaan tarvitse veloittaa sovelluksen lataamisesta mitään, eikä sovelluksessa tarvitse olla mainoksia, joten sovelluksia voi kehittää vain harrastuksena miettimättä raha-asioita.

Sovellusten luominen on henkisesti kehittävää ja kaiken vaivan arvoista. Mitä enemmän ohjelmoi, sitä enemmän oppii. Projektityöskentelyn kautta saa kokemusta työkuorman hallitsemisesta sekä ajankäytön optimoinnista. Sovelluksen saaminen valmiiksi tuo onnistumisen tunteen, ja sovelluksiaan voi myydä sovelluskaupassa. Toisin sanoen sovelluskehityksestä ei ole muuta kuin hyötyä.

LÄHTEET

Android Developers. n.d.a. Android App Bundle. Wiki. Luettu 12.5.2020. https://developer.android.com/platform/technology/app-bundle

Android Developers. n.d.b. Fragment. Wiki. Luettu 5.3.2020. https://developer.android.com/reference/android/app/Fragment.html

Android Developers. n.d.c. Meet Android Studio. Wiki. Luettu 13.5.2020. https://developer.android.com/studio/intro

Android Developers. n.d.d. Meet Google Play's target API level requirement. Wiki. Luettu 22.4.2020. https://developer.android.com/distribute/best-practices/develop/target-sdk

Android Developers. n.d.e. Version your app. Wiki. Luettu 13.5.2020. https://developer.android.com/studio/publish/versioning

Friesen, J. 2013. Learn Java for Android Development. 2. painos. New York City, NY, USA: Apress.

Hagos, T. 2019. Android Studio IDE Quick Reference. A Pocket Guide to Android Studio Development. 1. painos. New York City, NY, USA: Apress.

Harold, E. R. 2004. XML 1.1 Bible. 3. painos. Hoboken, NJ, USA: John Wiley & Sons.

Laster, B. 2016. Professional Git. 1. painos. Birmingham, UK: Wrox.

Marsicano K., Gardner B., Phillips B. & Stewart C. 2019. Android Programming. The Big Nerd Ranch Guide. 4. painos. Atlanta, GA, USA: Big Nerd Ranch.

Mew, K. M. 2011. Android 3.0 Application Development Cookbook. 1. painos. Birmingham, UK: Packt Publishing.

Play Console Ohjeet. n.d.f. Play Consolen käyttö. Käyttöohjeet. Luettu 2.3.2020. https://support.google.com/googleplay/android-developer/answer/6112435

Play Console Ohjeet. n.d.g. Sovelluksen julkaisu. Käyttöohjeet. Luettu 12.5.2020. https://support.google.com/googleplay/android-developer/answer/6334282

Play Console Ohjeet. n.d.h. Sovelluksen lähettäminen. Käyttöohjeet. Luettu 12.5.2020. https://support.google.com/googleplay/android-developer/answer/113469

Santacroce, F. 2015. Git Essentials. 1. painos. Birmingham, UK: Packt Publishing.

Statista. 2020. Number of available applications in the Google Play Store from December 2009 to March 2020. Luettu 10.5.2020. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

LIITTEET

Liite 1. Typistetty XML-tietokanta

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <category id="0" name="Kind" shortName="ki">
        <word name="Abandoned" />
        <word name="Abhorrent" />
        <word name="Able" plural="-" manner="" />
        <word name="Abolished" />
        <word name="Absent" />
        <word name="Absolute" plural="-" />
        <word name="Absolved" />
        <word name="Abstract" />
        <word name="Ace" plural="-" compa="" super="" manner="" />
        <word name="Active" />
        <word name="Adamant" />
        <word name="[Number]" />
    </category>
    <category id="1" name="Concept" shortName="co" implicitPlural="true">
        <word name="Abandonment" plural="" actor="4ed" />
        <word name="Aberration" actor="4nt" />
        <word name="Ability" actor="!Able" />
        <word name="Abomination" />
        <word name="Absence" plural="" actor="2t" />
        <word name="Absolution" plural="" actor="3e" />
        <word name="Accolade" />
        <word name="Accord" actor="ant" />
        <word name="Accordance" plural="" actor="2t" />
        <word name="Account" actor="ant" />
        <word name="Accuracy" actor="2te" />
        <word name="Ache" actor="-" />
    </category>
    <category id="2" name="Substance" shortName="su">
        <word name="Acid" plural="-" />
        <word name="Adhesive" plural="-" />
        <word name="Aero" />
        <word name="Aether" />
        <word name="Air" plural="-" />
        <word name="Ambrosia" />
        <word name="Amethyst" plural="-" />
        <word name="Antidote" plural="-" />
        <word name="Aqua" />
        <word name="Ash" plural="-" />
        <word name="[Color]" />
    </category>
    <category id="3" name="Thing" shortName="th" implicitPlural="true">
        <word name="Accelerator" />
        <word name="Acre" />
        <word name="Action" />
        <word name="Address" />
        <word name="Alarm" />
        <word name="Almond" />
        <word name="Ammunition" plural="" />
        <word name="Amoeba" />
        <word name="Amulet" />
        <word name="Analysis" plural="2es" />
        <word name="Anchor" />
        <word name="Angle" />
    </category>
    <category id="4" name="Person &amp;amp; Creature" shortName="pc" implicitPlural="true">
        <word name="Ace" />
        <word name="Achilles" plural="" />
        <word name="Acolyte" />
        <word name="Adversary" />
        <word name="Alchemist" />
        <word name="Alien" />
        <word name="Alliance" />
        <word name="Ally" />
        <word name="Angel" />
        <word name="Animal" />
        <word name="Anubis" plural="" />
        <word name="Anybody" plural="" />
        <word name="Aphrodite" plural="" />
        <word name="[Character's Name]" />
        <word name="[Nationality]" />
    </category>
```

```
    <category id="5" name="Action" shortName="ac" implicitPlural="true" defaultPrepos="by">
    <!-- Attributes: plural, noun, present participle, present, past, past perfect,
         actor, prepositions, no default prepositions -->
        <word name="Abandon" plural="" noun="ment" />
        <word name="Abduct" noun="-" />
        <word name="Absolve" noun="2ution" prepos="From" />
        <word name="Absorb" noun="1ption" />
        <word name="Abuse" />
        <word name="Accelerate" noun="-" prepos="Towards" />
        <word name="Accept" noun="ance" />
        <word name="Adapt" noun="ation" prepos="to" noDefPrepos="-" />
        <word name="Add" noun="ition" prepos="on, to" />
        <word name="Adore" noun="1ation" />
        <word name="Advance" prepos="to, Towards" noDefPrepos="-" />
        <word name="Affect" noun="!Effect" />
        <word name="Agree" noun="ment" prepos="With, to" />
        <word name="Arise" past="!Arose" perf="-" noDefPrepos="-" />
        <word name="Awaken" noun="ing" perf="!Awoken" actor="!Waker" prepos="to" />
    </category>
    <category id="6" name="Place &amp;amp; Time" shortName="pt" implicitPlural="true">
        <word name="Abode" />
        <word name="Abyss" />
        <word name="Academy" />
        <word name="Age" />
        <word name="Altar" />
        <word name="Always" plural="" />
        <word name="Apartment" />
        <word name="Arcade" />
        <word name="Archipelago" />
        <word name="Archive" />
        <word name="Arctic" plural="" />
        <word name="Area" />
        <word name="Arena" />
        <word name="[Month]" />
        <word name="[Place Name]" />
    </category>
    <category id="7" name="Grammatical Particle" shortName="gp">
        <word name="About" noArticle="-" notLast="-" />
        <word name="Above" />
        <word name="After" />
        <word name="Again" noArticle="-" />
        <word name="Anymore" noArticle="-" />
        <word name="Apart" noArticle="-" />
        <word name="Around" noArticle="-" />
        <word name="Astray" />
        <word name="Away" noArticle="-" />
        <word name="and" lowercase="-" notLast="-" />
        <word name="as" lowercase="-" notLast="-" />
        <word name="at" lowercase="-" notLast="-" />
        <word name="but" lowercase="-" notLast="-" />
        <word name="by" lowercase="-" notLast="-" />
        <word name="for" lowercase="-" notLast="-" />
        <word name="in" lowercase="-" notLast="-" />
        <word name="of" lowercase="-" notLast="-" />
        <word name="on" lowercase="-" notLast="-" />
        <word name="or" lowercase="-" notLast="-" />
        <word name="so" lowercase="-" />
        <word name="to" lowercase="-" notLast="-" />
        <word name="vs" lowercase="-" notLast="-" />
        <word name="yet" lowercase="-" />
    </category>
</resources>
```