

Ostoslista-sovelluksen kehitys Android- käyttöjärjestelmälle

Mika Vehosalmi



Tekijä

Mika Vehosalmi

Koulutusohjelma

Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko

Ostoslista-sovelluksen kehitys Android-käyttöjärjestelmälle

**Sivu- ja lii-
tesivumäärä**
35 + 1

Android on tämän hetken suosituin mobiilikäyttöjärjestelmä, jota käytetään puhelinten ja taulutietokoneiden lisäksi televisioissa, autoissa ja älykelloissa. Android-sovelluskehityksen etuna muihin nähden on sen helppoudessa, edullisuudessa ja sovelluskoodin siirrettävyydessä. Sovelluskehittäjälle Android tarjoaa ilmaisten kehitystyökalujen lisäksi tukea sovelluksen kehityksen eri vaiheissa, eikä sovelluksen julkaiseminen maailmanlaajuisille markkinoille vaadi kehittäjältä suuria investointeja.

Tässä opinnäytetyössä tutkitaan Android-sovelluskehityksen vaiheita tapauskohtaisesti käyttämällä Case-menetelmää. Työn aikana kehitetään Ostoslista-sovellus. Tarkoituksena on kehittää sovellus, jonka avulla käyttäjä pystyy lisäämään sovellukseen tuotteita ja poistamaan niitä.

Työn tuloksena saatiin osoitettua, että Androidia voi hyvin käyttää mobiilisovelluksien kehittämiseen ja se soveltuu mainiosti kehittäjälle, joka on vasta aloittamassa Android-sovelluskehitystä. Sovellus on ladattavissa Google Play -sovelluskaupasta ja siitä on maksettu 25 dollarin kustannus viralliseksi sovelluskehittäjäksi rekisteröitymisestä. Sovellusprojekti on kokonaisuudessaan saatavilla myös GitHub-sivustosta.

Asiasanat

Android, ostoslista, sovelluskehitys, julkaisu, mobiili, GitHub

Termit ja lyhenteet

ADT	Android Development Tools, Kehitystyökalut, jotka mahdollistavat Android-sovelluskehityksen Eclipse ympäristössä
API	Application Programming Interface, Ohjelmointirajapinta, jonka avulla eri sovellukset voivat keskustella keskenään
APK	Android Application Package, Android sovelluksen asennuspaketti
ART	Android Runtime, Virtuaalikone, Dalvik virtuaalikoneen seuraaja, jota Android käyttää sovellusten ajamiseen ja mahdollistaa useamman sovelluksen ajamisen samaan aikaan
AVD	Android Virtual Device, Virtuaalinen Android-laite, jonka avulla voi fyysisen laitteen sijaan testata sovelluksia
CLI	Command Line Interface, Komentorivikäyttöliittymä, joka on käytössä esimerkiksi Git-versionhallintaohjelmassa
DVCS	Distributed Version Control System, Hajautettu versionhallinta, jota käytetään esimerkiksi Gitissä
GUI	Graphical User Interface, Graafinen käyttöliittymä, joka on käytössä esimerkiksi GitHub-työpöytäsovelluksessa
JDK	Java Development Kit, Kehitystyökalut, joiden avulla saadaan tuki ja tarvittavat työkalut Java sovelluksien kehittämiseen
LUA	Skriptikieli, jota voi käyttää Android-sovelluksien kehityksessä
NDK	Native Development Kit, Kehitystyökalut, jotka mahdollistavat Android-sovelluksien kehittämisen C ja C++ ohjelmointikielillä
NFC	Near Field Communication, Tiedonsiirtotekniikka, joka mahdollistaa tiedonsiirron laitteiden välillä lyhyillä etäisyyksillä
OHA	Open Hand Alliance, Yhtymä, joka muodostuu 84 teknologiayrityksestä ja sen tarkoituksena on yhdessä kehittää avoimenlähdekoodin standardeja mobiililaitteille
SDK	Software Development Kit, Kehitystyökalut, jotka mahdollistavat Android-sovelluksien kehittämisen
XML	Extensible Markup Language, Merkintäkieli, jolla sovelluksen käyttöliittymä yleensä rakennetaan Android-sovelluskehityksessä

Sisällys

1	Johdanto	1
2	Android-käyttöjärjestelmä	2
2.1	Android	2
2.2	Versiohistoria ja tärkeimmät uudistukset	2
2.3	Arkkitehtuuri	4
3	Android-sovelluskehitys	6
3.1	Sovelluksen perusteet	6
3.2	Sovelluksen elinkaari	7
3.3	Käytetyt ohjelmointikielet	8
3.4	SQLite	9
3.5	Kehitysympäristö	9
3.6	Versionhallinta	10
3.7	GitHub	12
3.8	Android API-version valinta	13
3.9	Material Design	14
3.10	Sovelluksen testaaminen	16
4	Sovelluskehityssuunnitelma	17
5	Ostoslista-sovelluksen kehittäminen	18
5.1	Vaatimusmäärittely	18
5.2	Suunnittelu	19
5.3	Toteutus	20
5.4	Testaus	24
5.5	Sovelluksen julkaisu	25
6	Pohdinta	28
6.1	Jatkokehitysideat	29
	Lähteet	31

1 Johdanto

Älypuhelimien kehittymisen ja yleistymisen myötä ei ole enää tarvetta käyttää paperisia hankalasti ylläpidettäviä kauppalistoja, vaan on olemassa vaihtoehto helpottamaan ostoksien tekemistä. Mobiilisovellukset ovat kasvattaneet suosiotaan viime vuosina ja sen myötä markkinoilla olevien sovelluksien valikoima on kasvanut kiihtyvää vauhtia. Sovelluskaupoissa on saatavilla monenlaisia ostoslista-sovelluksia, mutta monet niistä ovat hankalia käyttää tai sisältävät ominaisuuksia, joille ei yksinkertaisesti ole käyttöä.

Tämän opinnäytetyön tarkoituksena on kehittää yksinkertainen ja helppokäyttöinen Ostoslista-sovellus käyttäen Android Studio kehitysympäristöä ja julkaista se Google Play -sovelluskaupassa. Tavoitteena on tutustua Android-sovelluskehitykseen ja kehittää sovellus, jota voi käyttää päivittäisten ruokaostosten tekemiseen. Sovelluksen avulla käyttäjä voi helposti lisätä ja poistaa tuotteita sovelluksessa.

Opinnäytetyön aiheen valintaan vaikutti kiinnostukseni Android sovelluskehitykseen, halu syventää ohjelmointitaitoja ja kehittää osaamista ohjelmistokehittäjän työssä. Työssä käydään ensin läpi Android käyttöjärjestelmänä ja sitten siirrytään kertomaan tarkemmin Android sovelluskehityksestä. Tämän jälkeen kerrotaan sovellusprojektin sovelluskehitysuunnitelmasta. Seuraavaksi keskitytään Ostoslista-sovelluksen kehittämiseen, jossa käydään läpi sovelluksen vaatimusmäärittelyä, suunnittelua, toteutusta ja testausta ymmärrettävässä järjestyksessä menemättä sovelluksen ohjelmakoodiin yksityiskohtaisesti. Tämän jälkeen kerrotaan sovelluksen julkaisusta ja sen vaiheista Google Play -sovelluskaupassa. Työn lopussa käydään läpi projektista saatuja tuloksia, omaa oppimista ja pohditaan projektin ja sovelluksen onnistumista, sekä projektin ja sovelluksen jatkokehitys mahdollisuuksia.

2 Android-käyttöjärjestelmä

Käsitellään Androidia käyttöjärjestelmänä. Aluksi käydään läpi Androidia yleisesti ja sen historiaa. Tämän jälkeen käsitellään Androidin versiohistoriaa ja suurimpia mukana tulleita uudistuksia, sekä lopuksi Androidin arkkitehtuuria.

2.1 Android

Android on Googlen kehittämä Linux kerneliin perustuva käyttöjärjestelmä. Sitä käytetään pääosin puhelimissa ja tableteissa, mutta nykyään myös kasvavassa valikoimassa laitteita kuten televisioissa, älykelloissa ja autoissa. (Schmidt, 2016)

Android on avoimeen lähdekoodiin perustuva, mikä tarkoittaa, että kuka tahansa voi hakea käyttöjärjestelmän lähdekoodin ilmaiseksi, ja julkaista siitä oman versionsa. Mukana tulee myös täysi sovelluskehys, joka mahdollistaa sen, että kolmannen osapuolen sovelluksia voidaan kehittää, asentaa ja julkaista käyttäjille. (Hildenbrand, 2015)

Android Inc. perustivat Andy Rubin, Rich Miner, Nick Sears ja Chris White, Kalifornian Palo Altossa lokakuussa 2003. Heidän tavoitteenaan oli kehittää käyttöjärjestelmä mobiililaitteille, joka olisi tietoinen käyttäjän sijainnista ja mieltymyksistä. Yrityksen osti Google elokuussa 2005. Käännekohta Android käyttöjärjestelmän kehitykselle tapahtui marraskuussa 2007, kun Google tiedotti 34 yrityksestä muodostuvasta The Open Handset Alliance (OHA), jonka tarkoituksena oli kehittää avoimia mobiililaitte standardeja. Yleisen standardin etuna oli, että käyttöjärjestelmä ei ollut sidoksissa vain yhteen laitteeseen yhdellä valmistajalla. Ensimmäinen Android käyttöjärjestelmällä varustettu älypuhelin HTC Dream, tuli myyntiin lokakuussa 2008. (Brachmann, 2014)

2.2 Versiohistoria ja tärkeimmät uudistukset

Google nimeää jokaisen isomman Android päivityksen käyttämällä makeiden herkkujen nimiä aakkosjärjestyksessä, sitä miksi Google on päätenyt käyttämään näitä nimityksiä, ei tarkkaan tiedetä (Hildenbrand, 2016).

Taulukko 1. Androidin versiot (Android, 2017)

Versio	Koodinimi	Julkaisupäivä	Tärkeimmät uudistukset
1.5	Cupcake	27.4.2009	Pienoisohjelma tuki
1.6	Donut	15.9.2009	Erikokoisten näyttöjen tuki; Nopea haku työkalu; Android Market

2.1	Eclair	26.10.2009	Google Maps; Kotinäytön kustomointi; Teksti puheeksi toiminto
2.2	Froyo	20.5.2010	Toimintojen aktivointi puheella; Puhelimen käyttäminen tukiasemana
2.3	Gingerbread	6.12.2010	Pelien API tuki; NFC tuki; Akun hallintatyökalu
3.0	Honeycomb	22.2.2011	Tuki tablet laitteille, tablet versio; Järjestelmän sisäinen navigointipalkki; Nopeat asetukset palkki
4.0	Ice Cream Sandwich	18.10.2011	Kotinäytön laajempi kustomointi; Mobiilidatan käytön hallinta; Android Beam toiminto
4.1	Jelly Bean	9.7.2012	Toiminnalliset ilmoitukset; Useamman käyttäjätilin tuki laitteella
4.4	KitKat	31.10.2013	Edistyneempi puhe aktivointi; Sovelluksen kokonäyttö tila; Älykäs yhteystietoluettelo
5.0	Lollipop	12.11.2014	Material Design periaatteiden mukainen tyyli sovelluksille; Useamman näytön tuki; Ilmoitukset lukitusnäytöllä
6.0	Marshmallow	5.10.2015	Sovelluksien oikeuksien hallinta; Älykäs akun hallinta
7.0	Nougat	22.8.2016	Useamman sovelluksen ajo samaan aikaan; Suoravastaus ilmoitukseen toiminto

Androidin päivityshistoriaa tarkastellessa voidaan todeta, että päivityksien ajankohdat olivat epäsäännöllisiä, mutta tiheitä käyttöjärjestelmän alkuaikoina. Jokainen iso päivitys toi mukanaan tärkeitä uudistuksia, koska tuolloin Android oli toiminnallisuuksiltaan jäljessä muihin käyttöjärjestelmiin verrattuna. (Hill, 2015)

2.3 Arkkitehtuuri

Androidin arkkitehtuuri on ohjelmistokomponenteista muodostuva rakenne, joka on jaettu karkeasti viiteen osa-alueeseen. Nämä ovat sovelluskerros (Applications), sovelluskehys (Application Framework), kirjastot (Libraries), Android Runtime, laitteistoabstraktiokerros (Hardware Abstraction Layer) ja Linux kernel (Tutorialpoint, 2017; Android Developers 2017a). Kuvassa 1 on esitetty Androidin arkkitehtuurin hierarkiaa tarkemmin.



Kuva 1. Android-käyttöjärjestelmän arkkitehtuuri (Android Developers 2017a, mukaeltu).

Alimmassa kerroksessa sijaitsee Linux kernel, joka on koko käyttöjärjestelmän perusta. Android Runtime luottaa Linux kerneliin perustana oleviin toiminnallisuuksiin, kuten muistinhallinta (Android Developers, 2017a). Kernel tarjoaa yhteyden laitteiston ja muiden kerroksien välille ja hoitaa ydinpalvelut, kuten prosessin, muistin, turvallisuuden, verkon ja virranhallinnan sekä laitteistoajurit. (Meier, 2010, 13)

Laitteistoabstraktiokerros tarjoaa yhteyden laitteiston ja sovelluskehityksen välillä. Tämä kerros koostuu useista kirjastoista, joista jokainen toteuttaa käyttöliittymän tietyn tyyppiselle laitteiston osalle. Kun sovelluskehitys pyytää pääsyä laitteistoon, niin Android lataa kirjaston kyseiselle laitteiston osalle. (Android Developers, 2017a)

Seuraavassa kerroksessa sijaitsee kirjastot, kuten SQLite, WebKit, OpenGL ja media manager, jotka tukevat sovelluskehitystä. Android sisältää myös lukuisia C/C++ ydinkirjastoja, kuten libc ja SSL. Samassa kerroksessa on myös Android Runtime, jossa Android sovelluksia ajetaan ja isännöidään. Siellä on Dalvik virtuaalikone (Android Runtime (ART) Android 5.0 versiosta alkaen) ja ydinkirjastot, jotka yhdessä aikaansaavat Androidille sen tyyppillisen toiminnan. (Meier, 2010, 4, 13; Android Developers, 2017)

Dalvik on rekisteripohjainen virtuaalikone, joka huolehtii, että laite kykenee ajamaan useita sovelluksia samaan aikaan ongelmitta. Ydinkirjastojen kautta saadaan Java ja Android kirjastojen tärkeimmät toiminnallisuudet. Android Runtime muodostuu ydinkirjastoista ja Dalvik virtuaalikoneesta, joka luo perustan sovelluskehitykselle. (Meier, 2012, 15)

Android Runtime (ART) esiteltiin Android 4.4 versiossa kokeellisena virtuaalikoneena kehittäjille. Suurin ero Dalvik ja ART välillä on se, että Dalvik kääntää sovelluksen koodin tarpeen mukaan, mutta ART tekee sen etukäteen sovelluksen asennuksen yhteydessä. Virtuaalikoneen vaihdolla haluttiin nopeuttaa sovelluksien latautumisaikoja, parantaa laitteen akkukestoa, toteuttaa 64-bitin tuki ja parantaa tukea moniytimisille prosessoreille. (Konradsson, 2015, 3; Stackoverflow, 2016)

Sovelluskehitys on kerros, joka toimii rajapintana sovelluskehittäjälle, tarjoamalla sovelluksien kehittämiseen vaaditut luokat. Tässä kerroksessa ovat lisäksi managerit, jotka kommunikoivat suoraan sovelluksen kanssa ja hallinnoivat monia puhelimen perustoimintoja, kuten sijaintipalvelut. Sovelluskehitys tarjoaa kehittäjälle yhteyden laitteistoon, hallinnoi sovelluksen käyttöliittymää ja resursseja. (Meier, 2010, 14)

Ylimpänä on sovelluskerros, joka on ainoa käyttäjälle näkyvä osa. Sovelluskerroksessa rakentuvat kaikki sovellukset, niin natiivit kuin kolmannen osapuolen sovellukset. Sovelluskerros ajetaan Android Runtime kerroksessa, käyttämällä luokkia ja palveluita, jotka ovat saatavilla sovelluskehityksestä. (Meier, 2010, 14)

3 Android-sovelluskehitys

Käsitellään Androidin sovelluskehityksen yleisimpiä asioita. Aluksi käydään läpi Android-sovelluksen perusteita ja elinkaarta. Tämän jälkeen kerrotaan Android-sovelluskehityksessä käytetyistä ohjelmointikielistä, SQLite-tietokannasta, kehitysympäristöistä, sekä versionhallinnasta ja GitHubista yleisesti. Lopuksi käsitellään sovelluksen API-version valintaa ja vaikutuksia, Material Design suunnitteluperiaatteita, sekä sovelluksen testaamista.

3.1 Sovelluksen perusteet

Android-sovellus on yksikkö, joka voidaan asentaa, käynnistää ja käyttää itsenäisesti muista sovelluksista riippumatta. Kaikki sovellukset sijaitsevat omassa suojatussa ympäristössään. Android-käyttöjärjestelmä toteuttaa niin sanottua pienien oikeuksien periaatetta, mikä käytännössä tarkoittaa sitä, että jokainen sovellus saa oletuksena oikeudet vain niihin sovelluskomponentteihin mitä se tarvitsee toimiakseen ja tämä aikaansaa turvallisen ympäristön sovellukselle toimia. (Android Developers 2017b)

Sovellus koostuu sovelluskomponenteista, jotka ovat oleellisia sovelluksen rakentamiselle. Sovelluskomponenttien neljä tyyppiä ovat: Aktiviteetit (Activities), palvelut (Services), sisällöntarjoajat (Content providers) ja lähetyksen vastaanottajat (Broadcast receivers). Nämä komponentit määrittelevät sovelluksen ja tarjoavat kanavan sovellukseen järjestelmälle tai käyttäjälle. (Android Developers 2017b)

Aktiviteetti on sovelluksen käyttöliittymän yksittäinen näyttö, visuaalinen esitys sovelluksesta, joka tarjoaa kanavan kommunikoida käyttäjän kanssa. Sovelluksella voi olla useita aktiviteetteja ja ne käyttävät näkymiä (Views) ja fragmentteja käyttöliittymän luomiseen ja kommunikointiin käyttäjän kanssa. Sovelluksessa voi esimerkiksi olla aktiviteetti, joka näyttää listan viesteistä, toinen millä kirjoittaa viesti ja kolmas, jolla lukea viestejä. Aktiviteetit ovat toisistaan riippumattomia, vaikka ne yhdessä muodostavat käyttäjäkokemuksen sovelluksessa. Toinen sovellus voi ajaa minkä tahansa näistä aktiviteeteista, jos sovelluksen sallii. (Vogel 2016; Android Developers 2017b)

Palvelu on sovelluksen taustalla toimiva, käyttäjältä piilossa oleva komponentti, joka huolehtii pitkäkestoisten toimintojen, tehtävien suorittamisesta taustaprosesseille ja ne voivat kommunikoida muiden sovelluskomponenttien kanssa. Palvelu pysyy käynnissä niin kauan, että sen tehtävä on suoritettu. Se voi esimerkiksi pitää musiikkisovelluksen käynnissä taustalla sen aikaa, kun käyttäjä on toisessa sovelluksessa. (Vogel 2016; Android Developers 2017b)

Sisällöntarjoaja hallinnoi dataa, jota voidaan varastoida sovelluksen tietokantaan, internetiin tai mihin tahansa paikkaan johon sovelluksella on pääsy. Sisällöntarjoajan avulla sovelluksen dataa voidaan sen sallissa joko hakea tai muokata. Sovelluksessa voi esimerkiksi olla käyttäjien yhteystietoja hallinnoiva sisällöntarjoaja, jota mikä tahansa oikeudet omaava sovellus voi lukea tai tehdä tietyn käyttäjän tietoihin muutoksia. (Android Developers 2017b)

Lähetyksen vastaanottaja mahdollistaa järjestelmän ja sovelluksien lähettämät ilmoitukset. Tämän ansiosta sovellus voi vastata koko järjestelmän laajuisiin ilmoituksiin ja järjestelmä voi lähettää ilmoituksia sovellukseen, vaikka se ei olisi käynnissä. Sovellus voi esimerkiksi ilmoittaa käyttäjää tulevasta tapahtumasta tai ilmoittaa muille sovelluksille, että sen lataama data on niidenkin käytettävissä. Järjestelmän lähettämiä ilmoituksia on monenlaisia, kuten esimerkiksi "akku on vähissä". Lähetyksen vastaanottajilla ei ole näkyvää osaa sovelluksen käyttöliittymässä, mutta ne voivat luoda ilmoituksen tilapalkkiin käyttäjälle. (Vogel 2016b; Android Developers 2017b)

3.2 Sovelluksen elinkaari

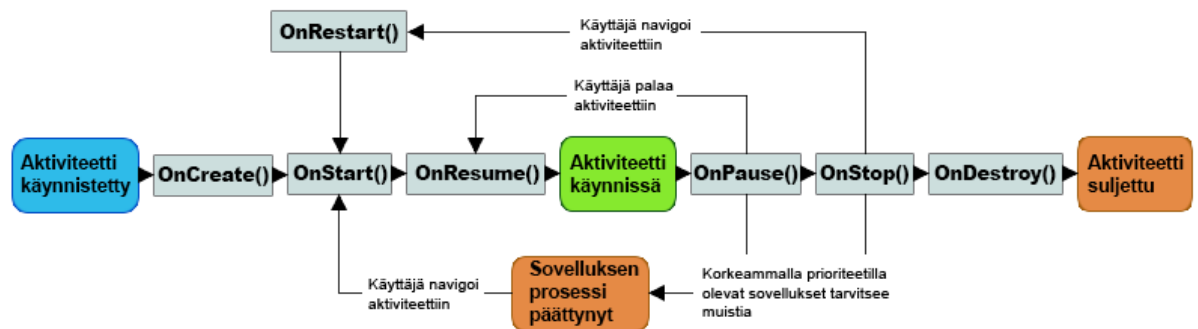
Sovelluksen aktiviteetteja hallinnoidaan aktiviteetti pinoksi kutsutun systeemin avulla. Kun uusi aktiviteetti käynnistetään, se asetetaan pinon päällimmäiseksi ja siitä tulee käynnissä oleva aktiviteetti. Aikaisempi aktiviteetti jää pinossa uuden aktiviteetin alapuolelle, eikä tule etualalle ennen kuin uusi aktiviteetti on suljettu. (Android Developers 2017c)

Sovelluksella voi olla yksi tai useampi aktiviteetti ja ne tulee määritellä Android-Manifest.xml tiedostoon, siten että yksi on pääaktiviteetti, joka käynnistyy, kun sovellus käynnistyy. (Nordlund 2016, 9)

Aktiviteetti voi olla erilaisissa tiloissa riippuen siitä, miten se keskustelelee käyttäjän kanssa ja nämä neljä oleellisinta tilaa ovat:

- Käynnissä, jos aktiviteetti on etualalla ja keskustelelee käyttäjän kanssa.
- Tauotettu, jos aktiviteetti on näkyvissä, mutta keskittyminen ei ole siihen, järjestelmä voi lopettaa prosessin muistin vapauttamiseksi.
- Pysäytetty, jos aktiviteetti on täysin piilossa toisen aktiviteetin alla.
- Lopetettu, jos aktiviteetin tila on tauotettu tai pysäytetty voi järjestelmä pudottaa aktiviteetin joko kysymällä tai lopettamalla sen prosessin. Ennen kuin aktiviteetti näytetään uudelleen käyttäjälle, se uudelleen käynnistetään ja palautetaan aikaisempaan tilaansa. (Vogel 2016a; Android Developers 2017c)

Kuvassa 2 on esitelty aktiviteetin elinkaaren vaiheet. Aktiviteetin kuusi ydinfunktiota ovat onCreate(), onStart(), onResume(), onPause(), onStop() ja onDestroy(), joista jokainen herätetään aina kun aktiviteetti vaihtaa tilaa. Näitä funktioita ohittamalla kehittäjä voi lisätä toimintoja, kun aktiviteetti liikkuu tilojen välillä. Kaikki aktiviteetit kutsuvat onCreate() funktiota alkuasetuksien tekemiseksi, mutta aktiviteetista riippuen eivät välttämättä jokaista näistä funktioista. Käyttäjälle näkyvä aktiviteetin elinkaari tapahtuu onStart() ja onStop() funktioiden välillä ja värilliset kohdat kuvaavat aktiviteetin eri tiloja. (Android Developers 2017c; Android Developers 2017d; Vogel 2016a)



Kuva 2. Aktiviteetin elinkaari (Android Developers 2017d, mukaeltu).

3.3 Käytetyt ohjelmointikiel

Android-sovelluskehityksessä käytetään pääosin Java ohjelmointikieltä. Sovelluskehitys on mahdollista tehdä Android NDK (Native Development Kit) kirjastoa käyttämällä C ja C++ ohjelmointikielillä, mutta tätä ei kuitenkaan suositella. Googlen ohjeistus on, että Android NDK kirjaston käyttäminen ei hyödytä valtaosaa sitä käyttävistä sovelluksista. Natiivin kirjaston koodin käyttäminen Android-sovelluskehityksessä tekee sovelluksesta monimutkaisemman, eikä aina paranna sen toimintaa huomattavalla tavalla. (Sims, 2016)

Android SDK (Software Development Kit) tarjoaa kaikki Android-sovelluksen kehityksessä tarvittavat kirjastot ja työkalut. Sovelluksen käyttöliittymä rakennetaan käyttämällä XML ohjelmointikieltä, joten pelkän Javan ymmärtäminen ei vielä riitä. Tämän lisäksi sovelluskehityksessä on mahdollista käyttää muita vaihtoehtoisia ohjelmointikieliä. (Sims, 2016)

Android-sovelluskehityksessä voi halutessaan kolmannen osapuolen kirjastojen avulla käyttää ohjelmointikieliä kuten LUA, HTML/CSS, Javascript ja C#, mutta mahdollisuuksien mukaan tulisi sovelluskehitykseen käyttää Java ja XML ohjelmointikieliä, jos sovelluksen toiminnan kannalta ei ole välttämätöntä käyttää muita ohjelmointikieliä. (Nordlund 2016, 11; Sims, 2016)

3.4 SQLite

SQLite on tietokantajärjestelmä, jonka käyttö ei vaadi erillistä tietokannanhallintaohjelmaa, eikä tietokantapalvelinta, koska tietokanta sijaitsee kokonaisuudessaan sitä käyttävän laitteen muistissa ja saa toiminnallisuutensa SQLite-kirjastosta. Tietokanta käyttää tietotyyppijärjestelmää, joka sallii minkä tahansa tietotyypin arvon tallentamisen mihin tahansa sarakkeeseen, riippumatta kyseisen sarakkeen tietotyypistä. Android Studiossa SQLite tietokanta esitellään ja sitä voi käyttää SQLiteOpenHelper luokan avulla. (SQLite 2017a-b; Android Developers 2017k)

3.5 Kehitysympäristö

Kehitysympäristön valinnassa ei ole yhtä oikeaa vaihtoehtoa. Sen valintaan vaikuttavat monenlaiset asiat, kuten projektin luonne, omat mieltymykset ja ohjelmointi osaaminen, mitkä on hyvä ottaa huomioon valintaa tehdessä. Oikean kehitysympäristön valinta voi parhaimmassa tapauksessa säästää turhautumisilta projektissa ja näin ollen vaikuttaa siihen miten koko projekti onnistuu. (Heller, 2016; Sinicki, 2016)

Android Studio on Googlen kehittämä virallinen IntelliJ IDEA perustuva kehitysympäristö, jonka mukana tulee Android SDK (Software Development Kit) kehitystyökalut. Android Studiossa käytetään Javaa ja Android SDK:n ansiosta esimerkiksi erilaisiin käyttöliittymä elementteihin pääsy ja toimivan käyttöliittymän rakentaminen XML käyttämällä (eXtensible Markup Language) onnistuu helposti. Android Studio tarjoaa hyvät apuvälineet Android-sovelluskehitykseen. (Sinicki, 2016; Android Developers 2017j)

Eclipse on Eclipse Foundation kehittämä vapaaseen lähdekoodiin perustuva kehitysympäristö, jota ei ole suunniteltu yksinomaan Android-sovelluskehitykseen, vaan sitä voidaan käyttää monilla eri ohjelmointikielillä. Android-sovelluskehityksessä Eclipsessä käytetään Javaa, mutta se tarvitsee erillisen ADT (Android Development Tools) lisäosan, jotta siihen saadaan tarvittavat Android-kehitystyökalut. Eclipse oli virallinen Androidin kehitysympäristö, ennen Android Studion julkaisua. Google ilmoitti ADT kehityksen ja tuen loppumisesta Android Studio 2.2 version julkaisun yhteydessä. Eclipse Foundation on käynnistänyt Eclipse Andmore nimellä kulkevan projektin, jonka tarkoituksena on jatkaa Android-sovelluskehityksen tukemista Eclipse ympäristössä. (Eclipse 2017a-b; Sinicki 2016; Eason 2016; Nordlund 2016, 12)

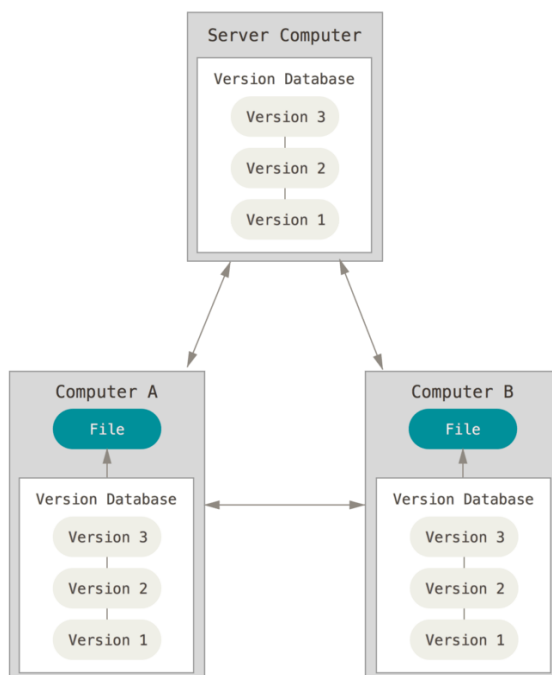
IntelliJ IDEA on JetBrains kehittämä Eclipsen tapaan useita eri ohjelmointikieliä tukeva kehitysympäristö, joka sisältää tuen Android-sovelluskehitykselle, mutta siihen on erikseen ladattava JDK (Java Development Kit) ja Android SDK kehitystyökalut. IntelliJ IDEA

on saatavilla Ultimate ja Community versioina. Community-versio on ilmainen ja tarkoitettu Java, sekä Android-sovelluskehitykseen. Ultimate-versio tuo laajemmin tukea eri ohjelmointikielille ja alustoille, mutta siinä on 30 päivän kokeilujakso, jonka jälkeen se on maksullinen. IntelliJ IDEA:n käytännölliseksi tekee lukuisat yhteisön tuottamat lisäosat, joiden ansiosta se on laajalti muokattavissa. (JetBrains 2017a-d; Furlan 2016)

3.6 Versionhallinta

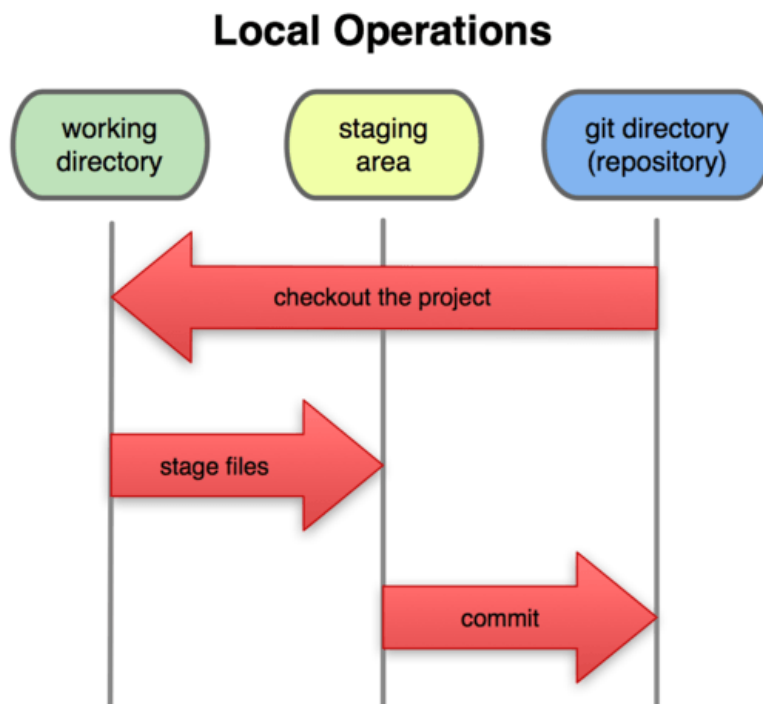
Versionhallinta on järjestelmä mikä tallentaa yhdessä tai useammassa tiedostossa tapahtuvat muutokset, jotta ne voidaan tarvittaessa myöhemmin palauttaa tilaan missä ne ovat toimineet. Sen käyttäminen omissa pienissä projekteissa on suotavaa, mutta erityisesti isoissa ohjelmistokehitysprojekteissa välttämätöntä. Versionhallinta mahdollistaa tiimin jäsenien vapaan työskentelyn, projektin muutoksien yhdistämisen ja niiden tarkastelun tarkemmin. Sen avulla voidaan vikatilanteissa tehtyjen muutoksien perusteella selvittää mahdollinen vian aiheuttaja ja koko projekti pystytään palauttamaan tilaan missä se on vielä toiminut. (Git 2017a; Tower 2017; Olevsky 2013)

Git on ilmainen, avoimeen lähdekoodiin perustuva, alun perin Linus Torvaldsin kehittämä ja nykyisin Junio Hamanon ylläpitämä hajautettu versionhallintaohjelmisto (Distributed Version Control System, DVCS), joka on yksi käytetyimmistä versionhallintatyökaluista, koska se toimii niin monissa eri käyttöjärjestelmissä ja kehitysympäristöissä. (Atlassian 2017; Halkola 2014, 15)



Kuva 3. Hajautettu versionhallinta Gitissä (Git 2017a)

Kuvassa 3 on esitelty hajautetun versionhallinnan toimintaperiaate. Git käyttää hajautettua versionhallintaa, joka tarkoittaa käytännössä sitä, että jokaisella kehittäjällä projektissa on paikallisesti tallennettu kopio projektista sisältäen kaikki muutokset ja tarvittaessa mitä tahansa näistä kopioista voidaan käyttää palauttamaan projekti palvelimelle tai toiselle kehittäjälle. (Git 2017a; Atlassian 2017; Halkola 2014, 14)



Kuva 4. Gitin paikalliset operaatiot. (Git 2017b)

Kuvassa 4 on esitelty Gitin toimintaperiaate pääpiirteittäin. Gitissä tiedostoilla on kolme keskeistä tilaa: muokattu (Modified), vaiheistettu (Staged) ja kommitoitu (Committed), missä ne voivat olla. Kun tiedostoon on tehty muutoksia, mutta niitä ei ole vielä tallennettu, niin tiedosto on muokattu-tilassa. Tiedosto on vaiheistettu-tilassa silloin, kun sitä on muokattu ja se on merkattu odottamaan tallentamista. Kun muutokset tiedostossa on tallennettu tietokantaan, niin tiedosto on kommitoitu-tilassa. Tiedoston tilojen lisäksi projektilla on omat osionsa Gitissä. (Git 2017b)

Gitissä projektilla on kolme pääasiallista osiota: työskentelyhakemisto (Working directory), vaiheistusalue (Staging area) ja git-hakemisto (Git-directory), missä se voi olla. Työskentelyhakemisto sisältää yhden version, lokaalisti tallennetun kopion Git-hakemistosta, johon voidaan tehdä muutoksia. Vaiheistusalue on Git-hakemistossa sijaitseva tiedosto, joka sisältää tiedot kaikista tiedostoista, jotka on merkattu odottamaan tallennusta. Git-hakemisto on osio, johon on tallennettu kaikki projektiin tehdyt muutokset ja tietokanta. Tästä osiosta tehdään kopio aina, kun projektia halutaan työstää jollain toisella koneella. (Git 2017b; Halkola 2014, 14)

3.7 GitHub

GitHub on graafisella käyttöliittymällä varustettu webpohjainen palvelu Git-versionhallintaa käyttäville projekteille, jossa kehittäjä voi jakaa ja julkaista omia projektejaan muille kehittäjille. Projektien säilytyspaikan ja versionhallinnan lisäksi se tarjoaa myös muita ominaisuuksia, kuten wikin ja tehtävienhallinnan projekteille. GitHub on tarkoitettu lähinnä sovelluskoodille, mutta sinne voi tallentaa ja käyttää sitä hallinnoimaan myös muun tyyppisiä tiedostoja. (Finley 2012; Halkola 2014)

Käyttäjätilin luominen GitHubissa on ilmaista, mutta jos projektin haluaa julkaista yksityisenä, niin siitä joutuu maksamaan. GitHub Education-sivusto tarjoaa opiskelijoille mahdollisuutta hakea käyttäjätiliin koko opiskeluaajan voimassa olevan ilmaisen paketin, minkä jälkeen voi rajattomasti julkaista yksityisiä projekteja. (Finley 2012; Halkola 2014; GitHub Education 2017)

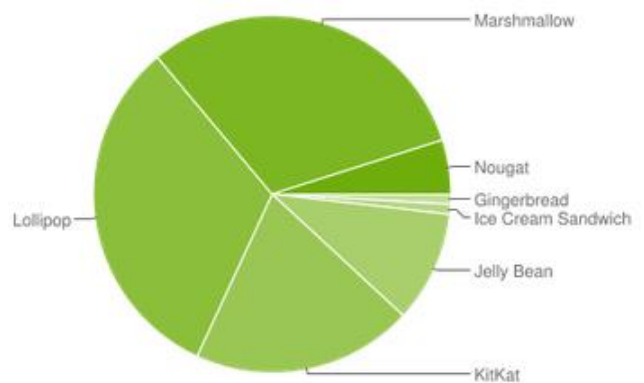
GitHubin erikoisuus on mahdollisuudessa ottaa osaa muiden käyttäjien projekteihin. Projektin voi kopioida toiselta käyttäjältä, vaikka siihen ei ole kirjoitusoikeutta, tehdä projektiin muutoksia ja jakaa ne takaisin alkuperäiselle käyttäjälle, joka voi halutessaan helposti yhdistää muutokset projektiin. (Finley 2012; Halkola 2014, 15)

Projektin lisäämiseksi GitHubiin on ensin ladattava joko Git-versionhallintaohjelma tai vaihtoehtoisesti GitHub-työpöytäsovellus, joiden mukana tulee tarvittavat GUI-työkalut projektin lisäämiseksi sivustolle. Sovellukset eroavat toisistaan siinä, että Gitissä on komentorivipohjainen käyttöliittymä (CLI, Command Line Interface) ja GitHub-työpöytäsovelluksessa on graafinen käyttöliittymä (GUI, Graphical User Interface). Molemmat näistä sovelluksista on saatavilla Gitin virallisilta sivuilta. Helpoin tapa projektin lisäämiseksi sivustolle on käyttää työpöytäsovellusta, koska se ei vaadi Gitin ohjeistusten ja termien tuntemusta. (GitHub Guides 2017; Git 2017c; Halkola 2014)

3.8 Android API-version valinta

Androidin API-versio on arvo, joka määrittelee Androidin-version ja sen tarjoaman viitekehysten version sovellukselle. Uusi Android-versio tuo mukanaan uuden API-version, joka tuo uusia ominaisuuksia ja tapoja käyttää sovellusta. Uusin API-versio on aina yhteensopiva muiden vanhempien API-versioiden kanssa, koska muutokset uudessa versiossa joko lisäävät tai korvaavat vanhemman version ominaisuuksia. Versiopäivityksen mukana tulevat muutokset, jotka korvaavat vanhemman API-version tuomia ominaisuuksia ei yleensä poisteta, vaan ne vanhenevat ja pysyvät edelleen sovelluksen käytettävissä. (Nordlund 2016, 14-15; Android Developers 2017f)

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.9%
4.1.x	Jelly Bean	16	3.5%
4.2.x		17	5.1%
4.3		18	1.5%
4.4	KitKat	19	20.0%
5.0	Lollipop	21	9.0%
5.1		22	23.0%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	4.5%
7.1		25	0.4%



Kuva 5 Androidin versiot, päivitetty 3.4.2017 (Android Developers 2017e)

Kuvassa 5 on esitelty Androidin eri versioiden käyttöasteet ja levinneisyys. Kaikki Android-versiot tukevat poikkeuksetta vain yhtä API-versiota. Sovellukselle määritellään minimi API-versio, jonka se tarvitsee toimiakseen. API-versio tulisi valita aina sovelluksen tarpeiden mukaan, koska sovelluksen Android-versio tuen lisäksi se vaikuttaa ominaisuuksiin, joita sovelluksessa voidaan käyttää. Vanhemman API-version Android-laitteita on edelleen paljon käytössä, joten kehittäjän on mietittävä, että pitääkö tärkeämpänä sovelluksen monipuolisia ominaisuuksia vai mahdollisesti laajempia käyttäjämääriä. Jos sovellus vaatii ominaisuuksia tietyistä API-versiosta, niin silloin on erikseen määriteltävä haluttu versio. Sovelluksen API-version voi määrittää Android Studiossa projektin build.gradle-tiedostossa. (Nordlund 2016, 14-15; Android Developers 2017f)

```

android {
    ...
    defaultConfig {
        ...
        minSdkVersion 14
        targetSdkVersion 24
    }
}

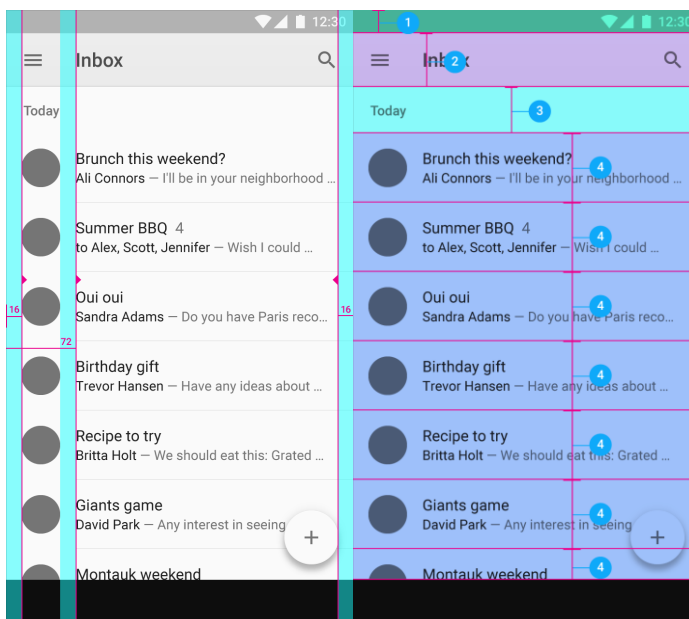
```

Kuva 6 API-version määrittely Android Studiossa (Android Developers 2017g)

Kuvassa 6 on esitelty tarkemmin API-version määrittäminen Android Studion build.gradle tiedostossa. Tiedostossa on minSdkVersion asetus, joka määrittää sovellukselle alimman Android-version, jossa sovellus toimii. Toinen asetus on targetSdkVersion, joka määrittää sovelluksen suunnitellun Android-version ja mahdollistaa myös määritellyn version ominaisuuksien käyttämisen sovelluksessa. Molemmat asetukset määrittellään API-version arvolla. Androidin-käyttöjärjestelmä vertaa ennen sovelluksen asennusta tiedoston asetusten arvoja laitteen Android-versioon ja varmistaa näin ettei sitä voida asentaa kuin laitteisiin, joissa on määritetty Android-versio. (Android Developers 2017g)

3.9 Material Design

Material Design on Googlen laatima ohjeistus kehittäjille sovelluksen ulkonäön suunnitteluun, joka julkaistiin samassa yhteydessä Android 5.0 versiopäivityksen kanssa. Tämän ohjeistuksen tavoitteena on luoda hyvän suunnittelun periaatteita noudattava visuaalinen kieli, joka mahdollistaa yhtenäisen käyttäjäkokemuksen luomisen käytettävästä laitteesta tai näytön koosta riippumatta. (Nordlund 2016, 12-13; Google 2017)



Kuva 7 Listanäkymän ohjeistuksia Material Design-sivustolla (Google 2017b)

Kuvassa 7 on esitelty ohjeistuksia listanäkymän eri käyttöliittymä osien toteuttamiseen ohjeistusten ja periaatteiden mukaisesti. Material Design-sivusto tarjoaa yksityiskohtaisia ohjeistuksia, jotka kehittäjän on hyvä opetella tuntemaan ja alkaa käyttää niitä parempien käyttäjäkokemusten aikaansaamiseksi. (Nordlund 2016, 12-13; Moyers 2016)

Blue			
500	#2196F3		
50	#E3F2FD		
100	#BBDEFB		
200	#90CAF9		
300	#64B5F6		
400	#42A5F5		
500	#2196F3		
600	#1E88E5	A100	#82B1FF
700	#1976D2	A200	#448AFF
800	#1565C0	A400	#2979FF
900	#0D47A1	A700	#2962FF

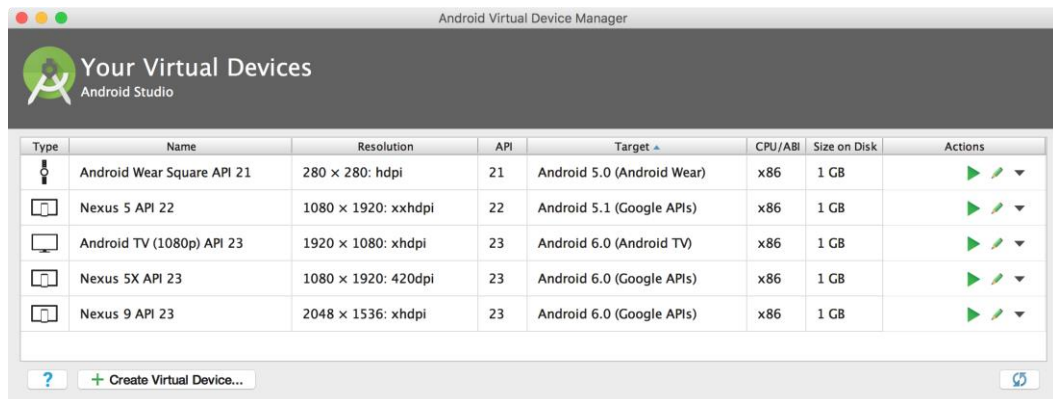
Kuva 8 Väripaletti esimerkki Material Design-sivustolla (Google 2017c)

Kuvassa 8 on esitelty esimerkki väripaletista Material Design-sivustolla, joka on sellaisenaan valmis käytäväksi sovelluksessa. Paletissa on numeroitu eri värisävyt selventämään niiden käyttötarkoitusta sovelluksen eri käyttöliittymä osioissa. Pääväri paletissa on numero viisisataa (500), sekä tummempia ja vaaleampia sävyjä kontrastien luomiseksi. Lisäksi kuvassa 8 on väripaletin oikealla puolella A-alkuisesti numeroidut toissijaiset värisävyt, jotka ovat tarvittaessa vaihtoehtoisia käytettäväksi sovelluksen käyttöliittymä osioissa, kuten esimerkiksi napeissa, tekstikentissä ja linkeissä. Tarkempia ohjeistuksia esimerkiksi värien käyttöön löytyy material.google.com -osoitteesta. (Nordlund 2016, Google 2017c)

Material Design ohjeistuksien tarkoituksena on luoda yhteneväisyyttä sovelluksen ja sen eri käyttöliittymä versioiden välille, mutta samalla auttaa kehittäjää itseään pysymään ajan tasalla uusista asioista, jotta tämä pystyy vastaamaan alati muuttuviin käyttäjien tarpeisiin. (Nordlund 2016; Moyers 2016)

3.10 Sovelluksen testaaminen

Android Studiosta löytyvät AVD (Android Virtual Device) ja Android-emulaattori, jotka yhdessä mahdollistavat sovelluksen testaamisen virtuaaliympäristössä. AVD on emulaattorin hallintatyökalu, jonka avulla voi muokata tietokoneella jo olemassa olevia virtuaalilaitteita tai määrittellä uusia Android-virtuaalilaitteita. Näitä määriteltyjä laitteita voi sitten simuloida Android-emulaattorissa. (Nordlund 2016, 15-16; Android Developers 2017i)



Kuva 9 Määriteltyjä virtuaalilaitteita AVD-hallintatyökalussa (Android Developers 2017i)

Kuvassa 9 on näkymä AVD-hallintatyökalusta, jossa näkyy siihen määriteltyjä Android-virtuaalilaitteita. Uutta virtuaalilaitetta määriteltessä on laitteelle valittava tyyppi, malli ja lopuksi haluttu Android-versio. Virtuaalilaitteet kannattaa määrittellä jokaiselle API-versiolle mitä halutaan, että sovellus voisi tukea, mikä tekee sovelluksen testaamisesta vaivatonta. AVD:n etuna fyysisen laitteen käyttämiseen testeissä on sen muokattavuus, saatavuus ja toimintavarmuus. (Android Developers 2017i; Krzyk 2016; Nordlund 2016, 15-16)



Kuva 10 Simuloitu virtuaalilaitte Android-emulaattorissa (Android Developers 2017h)

Kuvassa 10 on näkymä Android-emulaattorissa simuloidusta virtuaalilaitteesta ja hallintatyökaluista. Emulaattori käyttää virtuaalilaitteen simuloimiseen AVD löytyviä määrittelyjä ja se toimii ihan niin kuin fyysinen Android-laite. (Android Developers 2017h; Nordlund 2016)

4 Sovelluskehityssuunnitelma

Sovelluskehitysprojektissa päädyttiin käyttämään vesiputousmallin tyyppistä ratkaisua, joka sisältää vain osan vesiputousmallille tyypillisiä vaiheita, mutta se oli selkein ratkaisu käydä läpi Android-sovelluskehityksen vaiheita projektissa.

Vaatusmäärittely selkeyttää opinnäytetyöprojektia ja toimii apuvälineenä työn tekijälle. Laaditaan erillinen vaatimusmäärittäydokumentti, johon määritellään dokumentin tarkoitus ja kohderyhmä. Kuvataan projektia, asiakasta ja käyttötarkoitusta yleisesti, sekä sidosryhmiä. Määritellään sovelluksen toiminnalliset ja ei-toiminnalliset vaatimukset ja rajoitukset. Laaditaan käyttötapaukset tuotteen lisäys ja poisto toiminnallisuuksista. Kuvataan sovelluksen käyttävyttä ja toimintavarmuutta, sekä laaditaan sovelluksen tietokannasta tietokantakuvaudet. Määritellään rajoitukset projektin suunnittelulle ja toteuttamiselle.

Suunnittelu on tärkeä osa sovelluskehitysprojektia sillä sovelluksen toteutus ei onnistu ilman sitä ja kuten sanotaan, niin hyvin suunniteltu on puoliksi tehty. Sovellus suunnitellaan Googlen Material Design suunnitteluperiaatteiden mukaan, koska se tarjoaa aloittelevalle Android-sovelluskehittäjälle hyvät ohjeistukset sovelluksen suunnitteluun ja sillä varmistetaan sovelluksen yhdenmukaisuus laitteesta riippumatta.

Toteutus on se osa sovelluskehitysprojektia, jossa määrittelystä ja suunnittelusta saadaan tulosta aikaiseksi. Toteutetaan sovellus käyttämällä Android Studiota, koska se on virallinen Android-sovelluskehittämisen kehitysympäristö. Sovelluksen toteutus aloitetaan tuotteen lisäys ja tuotenäkymän toteutuksesta. Seuraavaksi toteutetaan tuotteen poisto ja sovelluksen splash-ikkuna. Lopuksi toteutetaan sovellukseen navigointipalkki, jossa on tuotelistan tyhjennys, järjestys ja suodatustoiminto. Toteutetaan sovellus toimimaan aluksi vain uusimmissa Android-versioissa, koska joidenkin toiminnallisuuksien toteuttaminen on hyvin hankalaa vanhemmissa versioissa ja uusien versioiden myötä on tullut uudistuksia, jotka helpottavat sovelluksen toiminnallisuuksien kehittämistä ja niihin löytyy paremmin ohjeistuksia. Sovellus toteutetaan suunniteltujen Material Design periaatteiden mukaisesti.

Testaus on sovelluksen toimivuuden kannalta tärkeä osa sovelluskehitysprojektia, koska sillä voidaan varmistaa yhteensopivuus tietystä Android-versiossa, havaita toteutusvirheet ja varmistaa sovelluksen laatu. Testataan sovellus Android Studiosta löytyvällä emulaattorilla ja fyysisellä Android-laitteella. Tehdään testausuunnitelma, koska se selkeyttää osataan testauksien tekemistä. Siinä käydään lyhyesti läpi testauksen kohde, tavoitteet ja rajataan testaus, sekä käytettävät testausmenetelmät. Laaditaan testitapaukset sovelluksen toiminnallisuuksien osalta.

5 Ostoslista-sovelluksen kehittäminen

Käsitellään sovelluskehitysprojektin aikana syntyneen ostoslista-sovelluksen kehittäminen. Ensin käydään läpi kuinka sovelluksen kehityksen vaiheet vaatimusmäärittely, suunnittelu, toteutus ja testaus projektissa hoidettiin. Lopuksi kerrotaan sovelluksen julkaisusta ja sen vaatimuksista yleisesti.

5.1 Vaatimusmäärittely

Sovelluksen vaatimusmäärittely laadittiin opinnäytetyön puitteissa kuvaamaan sovelluksen pääpiirteet ja sen tärkeimmät vaatimukset. Dokumentti toimi apuvälineenä projektin tekijälle ja selkeytti opinnäytetyöprojektia.

Projektin puitteissa suunniteltiin ja toteutettiin Ostoslista-sovellus. Käyttäjän oli sovelluksen avulla pystyttävä lisäämään, tarkastelemaan ja poistamaan listalla olevia tuotteita. Sovelluksen käyttäjiä olivat opinnäytetyöntekijä ja ketä tahansa, jolla oli tarve kyseiselle sovellukselle. Opinnäytetyöprojektilla ei ollut toimeksiantajaa, mutta projektin tavoitteena oli saada sovellus viimeistelyä siihen kuntoon, että sen voisi julkaista Google Play -sovelluskaupassa. Sovelluksen tarkoituksena oli tarjota käyttäjälle helppokäyttöinen Ostoslista-sovellus päivittäisten ruokaostoksien tekemiseen.

Taulukko 1. Sovelluksen sidosryhmät

Sidosryhmäluokka	Rooli
LOPPUKÄYTTÄJÄ	Käyttäjä – haluaa käyttää sovellusta

Sovelluksen loppukäyttäjiä olivat kaikki, jotka latasivat sovelluksen Google Play -sovelluskaupasta, kun se oli julkaistu. Sovellus suunniteltiin ja toteutettiin Android-käyttöjärjestelmälle ja sovelluksen API-versio määriteltiin niin, että sovelluksen olisi toimittava aluksi vain uusimmissa Android-versioissa. Sen oli toimittava luotettavasti 4.1 ja 6.0 Android-versioiden välillä.

Sovelluksen toiminnallisina vaatimuksina oli, että käyttäjän oli pystyttävä lisäämään ja poistamaan tuotteita sovelluksella. Rajoituksena sovelluksen toiminnallisille vaatimuksille määriteltiin, että projektin aikana sovellukseen ei toteutettaisi toiminnallisuutta, joka hyödyntäisi ruokakauppojen tietokantoja tuotteista ja niiden tiedoista. Sovelluksen käyttötapaukset laadittiin tuotteen lisäyksestä ja poistosta, niin että niistä kävi ilmi käyttötapausten oleellimmat asiat.

Taulukko 2. Käyttötapaus tuotteen lisäämisestä

Käyttötapaus:	Tuotteen lisääminen listalle
Yhteenveto:	Sovelluksen käyttäjä lisää tuotteen ostoslistalle
Toimijat:	Käyttäjä
Ehdot:	Käyttäjän on ennen tuotteen lisäämistä täytettävä tuotteen tiedot
Kuvaus:	Käyttäjä klikkaa sovelluksessa tuotteen lisäysnappia, täyttää tuotteen tiedot ja lisää tuotteen ostoslistalle
Poikkeukset:	Tuotteelle ei ole annettu tietoja tai ne ovat puutteellisia
Lopputulokset:	Tuote on onnistuneesti lisätty ostoslistalle

Taulukko 3. Käyttötapaus tuotteen poistamisesta

Käyttötapaus:	Tuotteen poistaminen listalta
Yhteenveto:	Sovelluksen käyttäjä poistaa tuotteen ostoslistalta
Toimijat:	Käyttäjä
Ehdot:	Käyttäjän on klikattava tuotteen poistonappia sen poistamiseksi
Kuvaus:	Käyttäjä valitsee tuotteen jonka haluaa poistaa ostoslistalta, klikkaa tuotteen poistonappia ja poistaa tuotteen listalta
Poikkeukset:	Ostoslistalla ei ole yhtään tuotetta
Lopputulokset:	Tuote on onnistuneesti poistettu ostoslistalta

Sovelluksen ei-toiminnallisina vaatimuksina oli, että sovelluksen käyttöjärjestelmänä toimi Android, ohjelmointikielinä Java ja XML, sekä projektin versionhallintana GitHub. Muita vaatimuksia sovelluksen osalta ei ollut. Vaatimukset sovelluksen ominaisuuksista asetettiin, jotta sovellus toimisi tarkoituksensa mukaisesti ja tehokkaasti, sekä olisi mahdollisimman käyttäjäystävällinen. Sovelluksen säilytettävistä tiedoista laadittiin tietokantakuvaukset, jotka ovat tämän työn liitteenä. Rajoituksena projektin tekemiselle määriteltiin, että projektille oli aikaa 400 tuntia ja se oli suunniteltava ja toteutettava tämän ajan puitteissa.

5.2 Suunnittelu

Sovelluksen ulkonäkö suunniteltiin Googlen Material Design periaatteiden mukaisesti ja sovelluksen suunnittelussa painotettiin ennen kaikkea sitä, että sovellus olisi mahdollisimman selkeä ja helppokäyttöinen. Tämä saatiin aikaan minimoimalla sovelluksessa tapahtuvat, niin sanotut turhat klikkaukset ja näyttämällä käyttäjälle vain se osa käyttöliittymää kuin milläkin hetkellä on tarve. Tämän toivon tekevän sovelluksen käyttämisestä nopeaa ja lisäävän sitä mahdollisuutta, että käyttäjät jatkaisivat sovelluksen käyttöä. So-

vellus suunniteltiin toimivan aluksi vain uusimmissa Android-versiossa ja mahdollisissa jatkokehitysvaiheessa tuotaisi tuki myös vanhempien Android-versioiden laitteille.



Kuva 11 Sovelluksen käyttöliittymämalleja

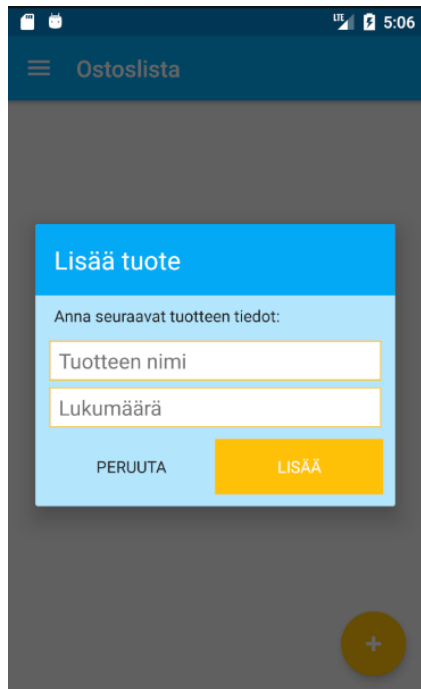
Kuvassa 11 on esitelty sovelluksen ensimmäisiä käyttöliittymä malleja, jotka on tehty käyttämällä Adobe Photoshop -ohjelmaa. Android Studiosta löytyy myös käyttöliittymän mallinnustyökalut, joiden avulla mallien laadinta olisi ollut huomattavasti helpompaa, mutta koska tämä selvisi vasta jälkeinpäin, niin ei niitä käytetty sovelluksen käyttöliittymä mallien suunnittelussa.

5.3 Toteutus

Sovellus toteutettiin Material Design suunnitteluperiaatteiden ja vaatimusmäärityksiensä mukaisesti. Toteutuksessa käytettiin kehitysympäristönä Android Studiota. Sovelluksesta oli aluksi kaksi eri versiota, koska sillä haluttiin testata erityyppien ratkaisujen toteutusta samaan aikaan. Ensimmäisessä versiossa oli addProductActivity tuotteen lisäämiselle ja ostoslista näytettiin MainActivity:n sisällä. Toisessa versiossa oli sovelluksen toiminnallisuudet toteutettu kaikki tapahtuvaksi MainActivity:n sisällä. Näistä päätyttiin lopulta jatkaamaan versiota, jossa kaikki sovelluksen toiminnallisuudet tapahtuivat MainActivity:n sisällä, koska se tuntui sopivammalta tavalta ja vastasi enemmän suunniteltua ja määriteltyä.

Sovelluksen toteutus aloitettiin tuotteen lisäys toiminnosta ja tuotelistan näkymän toteutuksesta. Tuotetietojen säilyttämistä varten käytettiin SQLite-tietokantaa. Sitä varten luotiin DBHelper ja DBContract Java-tiedostot. Näistä ensimmäisessä tiedossa käytettiin hyväksi SQLiteOpenHelper-luokan onCreate-metodia, mihin tuli tietokannan luontilauseet.

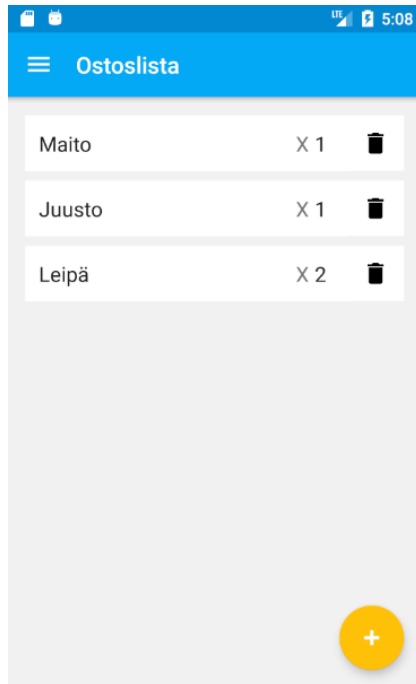
Toisessa tiedostossa oli määritelty tietokannan sarakkeet, sekä niiden tietotyypit, joita käytettiin DBHelper-tiedostossa. Sovellus luo DBHelper-tiedoston avulla SQLite-tietokannan ja se tallentuu oletuksena sitä käyttävän laitteen muistiin.



Kuva 12 Tuotteen lisääminen

Kuvassa 12 on esitelty tuotteen lisäys-ikkuna sovelluksessa. Tuotteen lisäys-ikkunan toiminnallisuuden toteutuksessa käytettiin hyväksi AlertDialog-luokkaa, jolla ikkuna saatiin aukeamaan saman näkymän sisällä. Tuotteen lisäys nappina on FloatingActionButton ja jotta AlertDialog suostuu aukeamaan sitä kautta, niin tuotteen lisäyksen koodi oli siirrettävä erilliseen metodiin. Kun tätä nappia painetaan, niin se kutsuu showDialogShow-metodia ja luo AlertDialog-olion ja näkymän.

Näkymälle metodi hakee tuotetietokenttien, nappien ja AlertDialogin tyylin erillisestä tyyli-tiedostosta ja asettaa näkymän AlertDialogille ja luo sen lopuksi. Tuotteen lisäys-ikkunassa Lisää-nappia kuuntelee napin setOnClickListener, jossa tuotetietokenttien tiedot tallennetaan String-tietotyypin muuttujiin. Jos muuttujat ovat tyhjiä, niin sovellus antaa siitä ilmoituksen käyttäjälle ja näyttää AlertDialogin uudestaan. Kun tiedot ovat kunnossa, niin tuote lisätään tietokantaan.



Kuva 13 Ostoslista-näkymä

Kuvassa 13 on esitelty sovellukseen lisättyjen tuotteiden näkyminen ostoslistanäkymässä. Tuotteiden näyttämiseen käytettiin hyväksi ListView-luokkaa. Ostoslista näkymä käyttää omaa tyylitiedostoa. Näkymää varten luotiin, sekä Product ja ListColumnAdapter Java-tiedostot. Product-luokka toimii apuluokkana ListColumnAdapter-luokalle. ListColumnAdapter-luokan avulla yhdelle tuoteriville listalla saadaan tuotua useampi sarakke tuotetietoja.

Tuotelistan päivitystä varten tehtiin updateUI-metodi, joka ajetaan aina sovelluksen käynnistyessä tai kun listaan on lisätty tai siitä on poistettu tuotteita. Metodi päivittää tuotelistan käymällä ensin tietokannan läpi, tallentaa haun tulokset kursoriin, käy kursorin läpi, asettaa luomalleen Product-oliolle kursorin ensimmäiseen ja toiseen paikkaan tallennetut arvot ja lisää Product-olion Productlist ArrayListille. Tämän jälkeen se asettaa ArrayList-olioon tallennetut tiedot uuteen ListColumnAdapter-olioon, hakee mProductListView näkymän ja asettaa adapterin näkymälle.



Kuva 14 Tuotteen poistaminen

Kuvassa 14 on esitelty tuotteen poistaminen sovelluksessa. Tuotteen poisto ikkunan toteutuksessa käytettiin myös AlertDialog-luokkaa, mutta lisäksi ja poistolla molemmilla on omat tyylitiedostonsa. Sovelluksessa on deleteProduct-metodi, joka ajetaan kun tuotteen poistonappia on painettu. Se luo AlertDialog-olion ja näkymän.

Näkymälle se hakee nappien ja AlertDialogin tyylin erillisestä tyylitiedostosta ja asettaa näkymän AlertDialogille ja luo sen lopuksi. Tuotteen poisto nappia kuuntelee napin setOnClickListener, joka hakee tuotelista näkymän tuotteen, tallentaa sen nimen String muuttujaan ja poistaa kyseisen nimisen tuotteen tietokannasta.

Sovelluksen splash-ikkunaa varten luotiin SplashActivity, joka ajetaan ennen MainActivity:n ajamista. Sovelluksen käynnistyessä SplashActivity luo Intent-olion, minkä avulla siirtyään MainActivity:n puolelle. Ennen siirtymistä se kuitenkin näyttää sovelluksen splash-ikkunan laitteen näytöllä, jonka se saa erillisestä tyylitiedoston kautta.

5.4 Testaus

Sovellus testattiin laaditun testaussuunnitelman ja testitapauksien mukaisesti käyttämällä Android-emulaattoria ja fyysistä Android-laitetta. Projektin alkuvaiheessa sovelluksen testaamiseen käytettiin Android Studiosta löytyvää emulaattoria. Uutta virtuaalilaitetta ei tarvinnut heti luoda, koska asennuksen yhteydessä Android Studio loi valmiiksi yhden virtuaalilaitteen, joka oli ilman suurempaa säätämistä heti käyttövalmiina. Android Studio loi oletuksena laitteen nimeltä Nexus 5 API 23. Laitteen nimessä loppuosa API 23 kertoi, että kyseessä oli Android 6.0 versiota vastaava virtuaalilaitte, joten sitä voitiin hyvin käyttää sovelluksen testaamiseen projektissa, koska sovellus suunniteltiin jo alun perin toimivan aluksi vain uusimmissa Android-versioissa.

Sovellusta testattiin projektin aikana vain tällä yhdellä virtuaalilaitteella, mutta myöhemässä vaiheessa myös fyysisellä Android laitteella. Testauksessa käytettiin Huawei Honor 7 laitetta, jossa oli Android 6.0 version käyttöjärjestelmä. Laite ei ollut suoraan käyttövalmis, vaan sen asetuksista jouduttiin kytkemään kehittäjätila ja usb debuggaus päälle, ennen kuin se saatiin näkymään Android Studiossa.

Molempien laitteiden käyttämisessä testauksessa oli omat hyvät ja huonot puolensa. Emulaattori oli omalta osaltaan monipuolisempi, koska virtuaalilaitteita pystyttiin luomaan tarpeen mukaan eri Android-versioilla, eikä se vaatinut lataamista ja oli muutenkin aika luotettava testauksessa. Virtuaalilaitetta vaivasivat kuitenkin ajoittaiset kaatumiset, sekä hitaus käynnistymisessä.

Fyysisen laitteen käyttämisen etuna oli, että nähtiin miltä sovellus siinä näyttää ja sen nopeus emulaattoriin verrattuna, koska laitteen ollessa jo käynnissä ei sen käynnistymiseen mennyt ylimääräistä aikaa. Android Studion tarvitsi asentaa vain sovelluksen APK paketti laitteeseen, jonka jälkeen sovellus oli testattavissa. Se ei kuitenkaan ollut yhtä monipuolinen kuin emulaattori, koska sitä voitiin käyttää testaamaan sovellusta vain laitteen Android-versiossa. Useamman Android-version testaaminen olisi vaatinut eri Android-versiota käyttävien laitteiden hankkimista ja tuonut lisäkustannuksia sovelluskehitysprojektille.

Sovelluskehitysprojektin koodin testaamiseen käytettiin Android Studiosta löytyvää Logcat monitorointi ominaisuutta. Sen avulla pystyttiin monesti selvittämään missä kohtaa sovelluksen koodissa oli ongelma, kun voitiin tarkistaa että mihin asti sovellus vielä toimi. Logcatin käyttämistä varten on ensin esiteltävä uusi muuttuja aktiviteetissa ja kutsuttava tätä muuttujaa kohdassa jota halutaan tarkistaa.

```
private static final String TAG = "MainActivity";
```

Kuva 15 Logcat muuttuja

Kuvassa 15 on esitelty Logcatin käyttämiseen vaadittu String-tietotyyppin muuttuja, joka on esiteltävä aktiviteetin alussa ennen Logcat viestiä. Muuttujan arvo on aina sen aktiviteetin nimi, jossa sitä käytetään.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    Log.d(TAG, "Oncreate: käynnistyy");  
}
```

Kuva 16 Logcat viesti metodissa

Kuvassa 16 on esitelty logcat debuggaus viestin käyttö sovelluksen onCreate-metodissa. Kun sovellus ajetaan seuraavan kerran, niin onCreate-metodi on metodeista ensimmäinen, joka suoritetaan sovelluksen käynnistyessä. Viesti tulee näkyviin Android Studiossa Android Monitorin puolella olevalle Logcat välilehdelle.



Kuva 17 Logcat viestin näkyminen

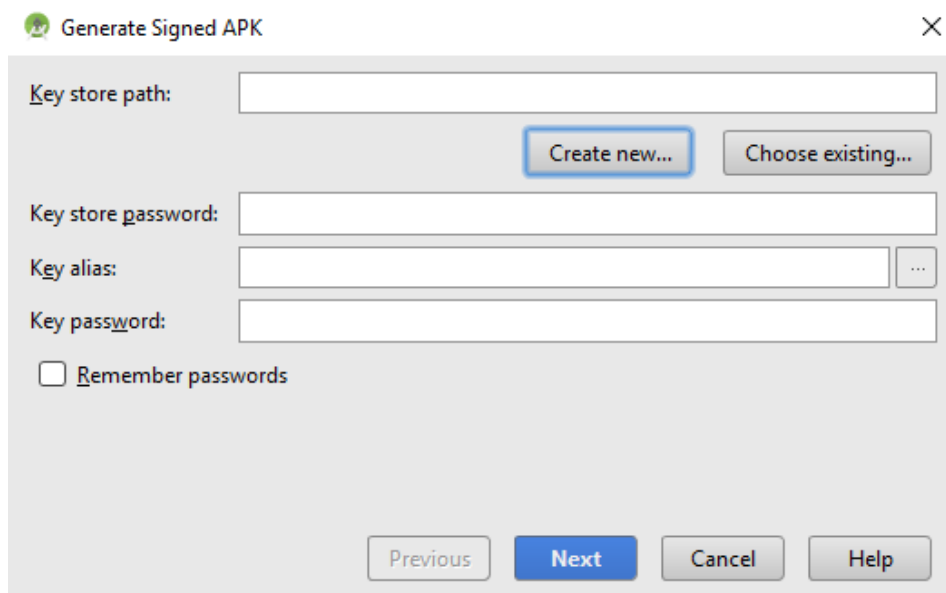
Kuvassa 17 on esitelty miten debuggaus viesti näkyy Logcat välilehdellä Android Monitorissa, jos sovellus on suorittanut onCreate-metodin ongelmitta. Viestin lisäksi siinä näkyy myös tarkat tiedot debuggauksen suoritusajankohdasta, sekä projektin nimi.

5.5 Sovelluksen julkaisu

Sovelluksen julkaisu tapahtuu Google Play -sovelluskaupassa ja sitä varten tarvitsee ensin luoda Google-tili, jos ei sellaista ole tai voi käyttää jo olemassa olevaa tiliä. Molemmissa tapauksissa on kuitenkin kehittäjän rekisteröitävä tili viralliseksi sovelluskehittäjän tiliksi, ennen kuin sovelluksia voi julkaista Google Play -sovelluskaupassa. Tilin rekisteröinti sovelluskehittäjätiliksi tapahtuu osoitteessa <https://play.google.com/apps/publish/signup/>.

Rekisteröinnin yhteydessä Google-tili yhdistetään osaksi kehittäjäkonsolia ja sitä varten on suoritettava 25 dollarin kertaluontoinen maksu, jonka jälkeen voi rajoituksetta julkaista sovelluksia. (Nordlund 2016; Nurmi 2015)

Sovelluskehittäjäksi rekisteröitymisen jälkeen on sovelluksesta luotava julkaisua varten APK-tiedosto. Sovelluksen APK-tiedoston versionumerointi tapahtuu Android Studiossa build.gradle-tiedostossa. Versiointi saadaan tiedossa määriteltä versionCode ja versionName kohdissa. Näistä kahdesta muuttujasta versionName voi määrittää niin kuin näkee parhaaksi, mutta versionCode tulee olla määritetty, niin että jokaisen uuden julkaistun sovelluksen version myötä sen arvo kasvaa aina yhdellä numerolla. (Nordlund 2016)



Kuva 18 APK-tiedoston ja salausavaimen luonti

Kuvassa 18 on esitelty allekirjoitetun APK-tiedoston luominen Android Studiossa. APK-tiedostoa luotaessa on tiedostolle samassa yhteydessä luotava salausavain, siltä varalta, että jos kehittäjän sähköposti tai salasana joutuu jostain syystä väärin käsiin, niin kukaan ei pääse julkaistuille sovelluksille tekemään vahinkoa, vaikka pääsisi kirjautumaan tilille. (Nordlund 2016)

APK-tiedoston luomisen jälkeen siirrytään takaisin kehittäjäkonsoliin ja sieltä jatketaan klikkaamalla luo sovellus-nappia. Järjestelmä pyytää seuraavaksi antamaan sovellukselle nimen ja valitsemaan sille kieli. Tästä jatketaan eteenpäin luomalla sovellus. Seuraava vaihe on sovelluksen tietosivu, jossa järjestelmä pyytää antamaan sovelluksen tietoja, mitkä tulevat näkyviin sovelluksen sivulle sovelluskaupassa. Tuotetietojen osalta on annettava sovelluksen lyhyt ja täysi kuvaus. Graafisen sisällön osalta on annettava vähintään kaksi kuvankaappausta sovelluksesta, yksi korkean resoluution kuvake ja ominai-

suuskuva. Luokittelun osalta on valittava sovelluksen tyyppi, arvot ja sisällön ikärajoitus. Sisällön ikärajoitus vaatii vastaamaan jokaisen kehittäjäkonsoliin julkaistun sovelluksen osalta luokittelukyselyyn ja pyytämään sisällönlukitusta. Kehittäjän yhteystietojen osalta on annettava sähköpostiosoite. Viimeisessä kohdassa pyydetään antamaan tietosuojakäytännön URL-osoite, mutta sitä ei ole pakko antaa heti, vaan voi valita että lähettää osoitteen myöhemmin.

Tämän jälkeen mennään kohtaan sovellusjulkaisut, jossa järjestelmä pyytää valitsemaan mihin vaiheeseen APK-tiedosto lisätään. Vaihtoehtoina on tuotanto, beta tai alfa. Kun vaihe on valittu ja allekirjoitettu APK-tiedosto lisätty, niin julkaisulle on vielä annettava nimi, jonka jälkeen se tarkistetaan ja tulee näkyviin kehittäjäkonsoliin.

Lopuksi on mentävä kohtaan hinnoittelu ja jakelu, jossa järjestelmä pyytää määrittämään missä maissa sovellus on saatavilla ja sisältääkö se mainoksia. Lupa asioiden osalta on hyväksyttävä sisältösäännöt ja yhdysvaltojen viestintälait. Kun kaikki tiedot on täytetty, niin sovellus on lähetettävä käsiteltäväksi. Kun käsittely on valmis, niin sovellus julkaistaan Google Play -sovelluskaupassa.

6 Pohdinta

Valitsin tämän työn aiheen mielenkiinnosta Android-sovelluskehitystä kohtaan, halusin laajentaa osaamistani ohjelmistokehittäjänä, sekä kehittää Java ohjelmointi taitojani ja oppia Android-sovelluskehityksen perusteet projektin aikana. Android sovelluskehityksestä ei ollut oikeastaan minkäänlaista kokemusta, eikä Java-ohjelmointi ollut koulussa koska se vahvin ohjelmointikieli. Siksi käytin ennen varsinaisen opinnäytetyöprosessin alkua aikaa sovelluskehitykseen tutustumiseen erilaisten harjoitusten avulla ja ne auttoivat osaltaan paremmin alkuun projektissa, kun ymmärsi jo joitain perusasioita Android-sovelluskehityksestä. Päädyin tekemään Ostoslista-sovelluksen tämän työn aikana, koska tarkoituksena ei niin sanotusti ollut lähteä keksimään pyörää uudestaan, vaan kehittää sovellus, josta olisi hyötyä projektin jälkeen, niin työn tekijälle kuin myös sovelluksen tuleville käyttäjille. Kyseisen sovelluksen kehitys tarjosi mielestäni riittävästi haasteita.

Projektin alussa käytettiin muutama viikko aikaa projektin suunnittelulle. Tämän koin osittain haastavaksi, koska en ollut hirveästi tehnyt esimerkiksi projektin dokumentointi puolta koulun Ohjelmistokehitys ja Softala-kursseilla. Projektin dokumentointi pohjien laadintaan käytin oppimaani ja aikaisemmilta kursseilta tallessa ollutta materiaalia, josta oli paljon apua. Projektinhallintaan käytettiin osittain sovellettua Scrum-menetelmää, koska se oli tutuin projektinhallinnan menetelmä. Sovelletulla tarkoitan sitä, että siitä oli karsittu pois päiväpalaverit ja muut tämän projektin osalta epäolennaiset asiat.

Projektin backlog tehtävät merkkasin sitä mukaan kun niitä tuli mieleen. Joitain tehtäviä merkkasin alustavasti jo seuraaville viikoille, mutta en tarkasti, koska en yksinkertaisesti tiennyt vielä mitä tulen tarkalleen ottaen seuraavalla viikolla tekemään. Sovelluksen käyttööntymän ja muun ilmeen suunnitteluun käytettiin myös paljon aikaa suunnitteluviikoilla. Projektin vaatimusmäärittäminen jäi osittain keskeneräiseksi, koska sen tekemiseen ei varattu tarpeeksi aikaa, mutta siinä saatiin määriteltä ainakin osa projektin vaatimusmäärittäyksestä ja se selkeytti osaltaan työn tekemistä. Työn suunnitteluvaihe meni tätä lukuun ottamatta mielestäni kaikin puolin hyvin, eikä vastaan tullut suurempia ongelmia.

Työn tavoitteena oli kehittää ja julkaista Ostoslista-sovellus, jolla käyttäjä pystyy lisäämään ja poistamaan tuotteita listalla. Sovellukseen oli lisäksi tarkoitus tulla navigointipalkki, jonka kautta käyttäjä pystyisi tyhjentämään ja järjestämään listalla olevia tuotteita, sekä suodattamaan tietoa piilottamalla esimerkiksi tuotteiden kappalemäärä tiedot. Projektiin käytettävä aika oli rajallinen ja tuotteen lisäys ja poisto toiminnallisuuksien ja muiden asioiden kanssa meni sen verran aikaa, että navigointipalkki oli jätettävä pois sovelluksen lopullisesta versiosta. Sovelluksen julkaisuversioon tuli mukaan vain tuotteen lisäys ja

poisto toiminnallisuudet. Tässä olisin mielestäni voinut pystyä parempaan, jos lähtötasoni projektiin olisi ollut erilainen ja työn toiminnalliseen osaan olisi pystynyt keskittymään enemmän. Työn alkupuolella pystyin keskittymään enemmän sovelluksen tekemiseen, mutta sen loppupuolella keskittyminen oli melkein pä yksin työn kirjallisessa osassa. Projektilla ei ollut toimeksiantajaa, mikä kanssa varmasti vaikutti sovelluksen lopputulokseen, kun ainoa palaute sovelluksesta tuli joko työn ohjaajalta tai kavereilta. Tästä johtuen josain kohtaa tuntui kuin olisin ensisijaisesti tehnyt sovellusta vain itselleni, vaikka se oli tarkoitus julkaista projektin päätteeksi Google Play -sovelluskaupassa.

Sovelluksen testaamiseen suunniteltiin alun perin käytettäväksi paljon enemmän aikaa kuin mitä lopulta siihen käytettiin. Testaamiselle oli varattu yksi kokonainen viikko projektista, milloin oli tarkoitus keskittyä yksinomaan testaamiseen. Sovellusta testattiin kyllä jatkuvasti kehittäessä joko emulaattorilla tai fyysisellä Android-laitteella, mutta tämä suunniteltu isompi testaus sessio jäi toteutumatta. Tähän oli myös syynä työn kirjalliseen osaan keskittyminen työn loppupuolella. Testaamisesta oli iso apu koko työn ajan ja se muutti joitakin sovelluksen toteutustapoja hiukan erilaiseksi, kun jokin ratkaisu ei tuntunutkaan niin hyvältä kuin aluksi oli ajatellut.

Projektin lopputulos vastasi suurimmilta osin sitä mitä olin itse projektin alkuvaiheessa ajatellut ja siitä jäi kokonaisuutena hyvä maku suuhun. Projektin vaiheissa olisi ollut jonkin verran parannettavaa ja asioita olisi voinut tehdä hiukan eritavalla. Opinnäytetyöprosessista jäi kokonaisuudessaan ihan positiivinen kuva. Työn aikana en käynyt kuin muutaman kerran kyselemässä apuja kirjallisen työn osalta opinnäytetyöpajalla, mutta muuten pärjäsi hyvin kaikilla sähköisillä ohjeistuksilla. Työ esiteltiin seminaarissa ja sekin meni hyvin, mutta jostain syystä odotin isompaa yleisöä paikalle. Projektin ohjauksia ohjaajan kanssa pidettiin aika vapaamuotoisesti silloin kun niille oli tarvetta ja tämä oli hyvä juttu. Mitä tulee itse sovellukseen, niin olen sen lopputulokseen tyytyväinen, kun sain kehitettyä sovelluksen josta on iloa ja hyötyä jatkossa. Sovellus ei käytännössä ole koskaan täysin valmis, vaan siihen on aina mahdollista lisätä jotakin uutta. Tulen varmasti korjaamaan sovelluksessa olevat puutteet ja jatkamaan sovelluksen kehittämistä tämän projektin jälkeen.

6.1 Jatkokehitysideat

Projektin jatkokehitystä ajatellen lähtisin ensimmäiseksi liikkeelle projektin suunnitteluvaiheesta. Kaikki projektin dokumentaatio tehtäisiin asian kuuluvalla tavalla ja sen tekemiseen varattaisi tarpeeksi aikaa. Suunnittelussa pidettäisi muutenkin enemmän huolta siitä että projektin tehtäville ja vaiheille olisi varattu mielellään vähän liikaa kun liian vähän aikaa. Toteutusvaiheessa panostettaisi siihen, että olisi lähtökohtaisesti parempi osaaminen Java-ohjelmoinnissa, mikä helpottaisi työn jatkokehitystä huomattavasti. Sovellus toteutet-

taisi toimimaan myös vanhemmissa Android-versioissa. Testausvaiheen tekisin myös toisella tavalla. Sovelluksesta voisi järjestää pienimuotoisia käytettävyytestestauksia ainakin muutaman kerran projektin aikana, jonka kautta saataisi mahdollisesti lisää ideoita käyttäjiltä kuinka sovellusta voisi parantaa. Testaussuunnitelmaa ja testitapauksia lähtisin miettimään laajemmin. Testausvälineenä emulaattori soveltui hyvin sovelluksen testaamiseen, mutta hankkisin silti muutamia eri Android-version fyysisiä laitteita testauksia varten, koska laitteilla on aina suorituskyvyllisiä eroja mitä ei emulaattorin avulla nähdä.

Sovelluksen jatkokehityksen osalta lähtisin ensin toteuttamaan navigointipalkin toiminnallisuudet kuntoon, jotka jouduttiin jättämään pois sovelluksen lopullisesta versiosta. Lisäisin sovellukseen tämän jälkeen tuotelista näkymään toiminnallisuuden, että käyttäjä voi merkata yksittäisen tuotteen kerätyksi, eikä sitä tarvitsisi vaan poistaa. Samalla voisi toteuttaa ominaisuuden, jonka avulla käyttäjä voisi valikoida useampia tuotteita mitkä voisi poistaa yhdellä kertaa. Tämän jälkeen laajentaisin sovelluksen tietokantaa lisäämällä siihen tuoteryhmä taulun. Sovelluksessa olisi tuotteella myös tuoteryhmä ja listalla olevat tuotteet voisi järjestää myös tuoteryhmän mukaiseen järjestykseen.

Lähtisin seuraavaksi tekemään parannuksia tuotteen lisäys ja poisto ominaisuuksiin sovelluksessa. Tuotteen lisäys ominaisuuteen lisäisin tarkistuksen, joka käy läpi ostoslistalla jo olevien tuotteiden nimet ja antaa käyttäjälle ilmoituksen jos tämä yrittää lisätä samannimistä tuotetta ostoslistalle. Poisto ominaisuuteen lisäisin mahdollisuuden pyyhkäistä sivulle tuotteen päällä poistaakseen tuotteen ostoslistasta tai piilottaisin tuotteen poistonapin pyyhkäisyn taakse, mikä vapauttaisi tilaa uusille tuotetiedoille tuoterivillä. Tämän jälkeen lähtisin suunnittelemaan ja toteuttamaan sovellukselle uutta ulkoasua. Sovelluksen ulkoasun muutosta varten voisi kerätä ideoita esimerkiksi käytettävyytestestauksista.

Seuraavaksi toteuttaisin sovellukseen ominaisuuden, joka hyödyntäisi kauppojen hintatietoja tuotteiden hinnoista, niin että sovellus pystyisi laskemaan ruokaostoksien kokonahinnan sitä mukaan kun tuotteita lisää ostoslistalle. Tämän jälkeen keskittyisin tekemään parannuksia sovellukseen aina sitä mukaan kun uusia ideoita tulee, mutta uusien isompien toiminnallisuuksien lisäämistä sovellukseen harkittaisi hyvin tarkkaan, ettei sovelluksesta tulisi käyttäjille liian monimutkainen käyttää.

Lähteet

Android Developers 2017a. Platform Architecture. Luettavissa:

<https://developer.android.com/guide/platform/index.html>

Luettu: 15.3.2017

Android Developers 2017b. Application Fundamentals. Luettavissa:

<https://developer.android.com/guide/components/fundamentals.html>

Luettu: 7.3.2017

Android Developers 2017c. Activity. Luettavissa:

<https://developer.android.com/reference/android/app/Activity.html>

Luettu: 6.3.2017

Android Developers 2017d. The Activity Lifecycle. Luettavissa:

<https://developer.android.com/guide/components/activities/activity-lifecycle.html>

Luettu: 22.2.2017

Android Developers 2017e. Platform Versions. Luettavissa:

<https://developer.android.com/about/dashboards/index.html#Platform>

Luettu: 4.4.2017

Android Developers 2017f. What is API Level?. Luettavissa:

<https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels>

Luettu: 10.4.2017

Android Developers 2017g. Specify API Level Requirements. Luettavissa:

<https://developer.android.com/studio/publish/versioning.html>

Luettu: 11.4.2017

Android Developers 2017h. Run Apps on the Android Emulator. Luettavissa:

<https://developer.android.com/studio/run/emulator.html> Luettu: 3.5.2017

Android Developers 2017i. Create and Manage Virtual Devices. Luettavissa:

<https://developer.android.com/studio/run/managing-avds.html#about> Luettu: 3.5.2017

Android Developers 2017j. Download Android Studio and SDK Tools. Luettavissa:

<https://developer.android.com/studio/index.html> Luettu: 21.3.2017

Android Developers 2017k. Saving Data in SQL Databases. Luettavissa:
<https://developer.android.com/training/basics/data-storage/databases.html>
Luettu: 19.5.2017

Android 2017. The Android Story. Luettavissa:
<https://www.android.com/history/#/marshmallow> Luettu: 21.2.2017

Atlassian 2017. What is Git. Luettavissa:
<https://www.atlassian.com/git/tutorials/what-is-git> Luettu: 21.3.2017

Brachmann, S. 2014. A Brief history of Google's Android Operating System. Luettavissa:
<http://www.ipwatchdog.com/2014/11/26/a-brief-history-of-googles-android-operating-system/id=52285/> Luettu: 20.2.2017.

Eclipse 2017a. About the Eclipse Foundation.
Luettavissa: <https://eclipse.org/org/> Luettu: 20.3.2017

Eclipse 2017b. Eclipse Andmore. Luettavissa:
<https://projects.eclipse.org/projects/tools.andmore> Luettu: 21.3.2017

Eason, J. 2016. Support ended for Eclipse Android Developer Tools. Luettavissa:
<https://android-developers.googleblog.com/2016/11/support-ended-for-eclipse-android.html> Luettu: 20.3.2017

Finley, K. 2012. What exactly is GitHub anyway?. Luettavissa:
<https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/> Luettu: 3.4.2017

Furlan, A. 2016. The Big List of Android Development Tools. Luettavissa:
<http://www.businessofapps.com/list-android-development-tools/> Luettu: 21.3.2017

Git 2017a. Getting Started – About Version Control. Luettavissa:
<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control> Luettu: 21.3.2017

Git 2017b. Getting Started – Git Basics. Luettavissa:
<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>
Luettu: 22.3.2017

Git 2017c. GUI Clients – Git. Luettavissa:

<https://git-scm.com/downloads/guis> Luettu: 5.4.2017

GitHub Education 2017. Student Developer Pack. Luettavissa:

<https://education.github.com/pack> Luettu: 4.4.2017

GitHub Guides 2017. Getting your project on GitHub. Luettavissa:

<https://guides.github.com/introduction/getting-your-project-on-github/>

Luettu: 5.4.2017

Google 2017. Material Design – Introduction. Luettavissa:

<https://material.io/guidelines/material-design/introduction.html#> Luettu: 25.4.2017.

Google 2017b. Material Design – Layout – Metrics & keylines. Luettavissa:

<https://material.io/guidelines/layout/metrics-keylines.html> Luettu: 2.5.2017

Google 2017c. Material Design – Style – Color. Luettavissa:

<https://material.io/guidelines/style/color.html#color-color-palette> Luettu: 2.5.2017

Halkola, J. 2014. Android-käyttöjärjestelmällä toteutettu liikuntapäiväkirjasovellus. Luettavissa:

http://theseus.fi/bitstream/handle/10024/83838/insinorityo_halkola_VALMIS_theseus.pdf?sequence=1 Luettu: 13.2.2017.

Heller, M. 2016. Choosing your Java IDE. Luettavissa:

<http://www.javaworld.com/article/3114167/development-tools/choosing-your-java-ide.html>

Luettu: 20.3.2017

Hildenbrand, J. 2015. What is Android? Luettavissa: <http://www.androidcentral.com/what-android> Luettu: 20.2.2017.

Hildenbrand, J. 2016. Inside the different Android versions. Luettavissa:

<http://www.androidcentral.com/android-versions> Luettu: 20.2.2017

Hill, S. 2015. How a yearly release cycle could improve the Android experience. Luettavissa: <http://www.androidauthority.com/google-android-yearly-release-cycle-benefits-607459/>

Luettu: 21.2.2017

JetBrains 2017a. Meet IntelliJ IDEA. Luettavissa:

<https://www.jetbrains.com/help/idea/2016.3/meet-intellij-idea.html>

Luettu: 21.3.2017

JetBrains 2017b. Android Support Overview. Luettavissa:

<https://www.jetbrains.com/help/idea/2016.3/android-support-overview.html>

Luettu: 21.3.2017

JetBrains 2017c. Choose your edition. Luettavissa:

https://www.jetbrains.com/idea/features/editions_comparison_matrix.html

Luettu: 21.3.2017

JetBrains 2017d. Prerequisites for Android Development. Luettavissa:

<https://www.jetbrains.com/help/idea/2016.3/prerequisites-for-android-development.html>

Luettu: 21.3.2017

Konradsson, T. 2015. ART and Dalvik performance compared. Luettavissa:

<http://www8.cs.umu.se/education/examina/Rapporter/TobiasKonradsson.pdf>

Luettu: 24.2.2017

Krzyk, K. 2016. Managing Android Virtual Devices during test session. Luettavissa:

<https://medium.com/azimolabs/managing-android-virtual-devices-during-test-session-98a403acffc2> Luettu: 3.5.2017

Meier, R. 2010. Professional Android 2 Application Development. Wiley Publishing, Inc.

Moyers, S. 2016. The Beginner's Guide to Google's Material Design. Luettavissa:

<https://speckyboy.com/beginners-guide-material-design/> Luettu: 25.4.2017.

Nordlund, M. 2016. Ostoslista Android-sovelluksen kehittäminen. Luettavissa:

http://theseus.fi/bitstream/handle/10024/111306/Nordlund_Miikka.pdf?sequence=1

Luettu: 13.2.2017.

Nurmi, O. 2015. Android-pelin kehitys ja julkaisu. Luettavissa:

http://theseus.fi/bitstream/handle/10024/100951/Opinnaytetyo_Oskari_Nurmi_2015_HETI.pdf?sequence=1 Luettu: 12.5.2017

Olevsky, I. 2013. Why Version Control Is Critical To Your Success. Luettavissa:
<http://www.codeservedcold.com/version-control-importance/> Luettu: 22.3.2017

Schmidt, C. 2016. What is Android? Here is a complete guide for beginners. Luettavissa:
<https://www.androidpit.com/what-is-android> Luettu: 20.2.2017.

Sims, G. 2016. I want to develop Android apps – What languages should I learn.
Luettavissa: <http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008/> Luettu: 16.3.2017

Sinicki, A. 2016. 10 completely different IDEs and methods for making Android apps.
Luettavissa: <http://www.androidauthority.com/10-completely-different-ides-and-methods-for-making-android-apps-701584/> Luettu: 20.3.2017

Stackoverflow 2016. What is difference between DVM and ART. Luettavissa:
<http://stackoverflow.com/questions/31957568/what-is-difference-between-dvm-and-art-why-dvm-has-been-officially-replaced-wi> Luettu: 24.2.2017

SQLite 2017a. Distinctive Features of SQLite. Luettavissa:
<http://www.sqlite.org/different.html> Luettu: 19.5.2017

SQLite 2017b. About SQLite. Luettavissa:
<https://www.sqlite.org/about.html> Luettu:19.5.2017

Tower 2017. Learn Version Control with Git. Luettavissa:
<https://www.git-tower.com/learn/git/ebook/en/command-line/basics/why-use-version-control> Luettu: 21.3.2017

Tutorialpoint 2017. Android architecture. Luettavissa:
https://www.tutorialspoint.com/android/android_architecture.htm Luettu: 21.2.2017

Vogel, L. 2016a. Activity life cycle. Luettavissa:
<http://www.vogella.com/tutorials/AndroidLifeCycle/article.html> Luettu: 6.3.2017

Vogel, L. 2016b. Broadcast receiver. Luettavissa:
<http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html>
Luettu: 16.3.2017

Liitteet

Taulukko 1. Product-taulun sarakkeiden määrittely

Taulun nimi: Product			
Sarake	Tietotyyppi	Pakollisuus	Kuvaus
id	INT	Kyllä	Yksilöi tuotteen, luodaan automaattisesti, tuotteen id (pääavain)
name	String	Kyllä	Tuotteen nimi
quantity	String	Kyllä	Tuotteen lukumäärä