

Ilkka Hannula

PLA-32820 Mobiiliohjelmointi

Oppimispäiväkirja

SISÄLLYSLUETTELO

[1. Johdanto 1](#_Toc507762631)

[2. Harjoitukset 2](#_Toc507762632)

[2.1 Tutustuminen mobiililaitteeseen 2](#_Toc507762633)

[2.2 Git ja versionhallinta 2](#_Toc507762634)

[2.3 Ympäristön asennus ja testaus 2](#_Toc507762635)

[2.4 Perus käyttöliittymä 2](#_Toc507762636)

[2.5 Tietokantasovellus 3](#_Toc507762637)

[2.6 Firebase reaaliaikainen tietokanta 4](#_Toc507762638)

[2.7 Firebase autentikaatio 5](#_Toc507762639)

[3. MOOC-kurssi 6](#_Toc507762640)

[3.1 Lesson 1 – Create project 6](#_Toc507762641)

[3.2 Lesson 2 – Connect to the Internet 6](#_Toc507762642)

[3.3 Lesson 3 - RecyclerView 6](#_Toc507762643)

[3.4 Lesson 4 – Intents 7](#_Toc507762644)

[3.5 Lesson 5 – Lifecycles 7](#_Toc507762645)

[3.6 Lesson 6 – Preferences 8](#_Toc507762646)

[3.7 Lesson 7 – SQLite 9](#_Toc507762647)

[3.8 Lesson 8 – Content Providers 9](#_Toc507762648)

[3.9 Lesson 9 – More Content Providers 10](#_Toc507762649)

[3.10 Lesson 10 – Background Tasks 10](#_Toc507762650)

[3.11 Lesson 11 – Completing the UI 11](#_Toc507762651)

[3.12 Lesson 12 – Polishing the UI 11](#_Toc507762652)

[3.13 Sunshine App 12](#_Toc507762653)

[3.14 Mietteet MOOC-kurssin jälkeen 12](#_Toc507762654)

[4. harjoitustyö 14](#_Toc507762655)

[4.1 Alkutilanne 14](#_Toc507762656)

[5. Ajankäyttö 15](#_Toc507762657)

# Johdanto

Tämä dokumentti on mobiiliohjelmoinnin kurssia varten ja tässä käydään läpi oppimisprosessia, sen haasteita ja yleisesti kaikenlaisia mietteitä, mitä tehtäviä tehdessä, tietoa hakiessa tai vaikka sängyssä pyöriessä on tullut mieleeni. Kurssilla ohjelmoidaan Android-ympäristössä.  
  
Kun vaihtolukukausi oli jo pitkällä, kun tarkoitus oli käydä HOPSi läpi ja poimia kurssit kevätlukukaudelle. En voinut olla huomaamatta, että nyt oli tullut mahdolliseksi valita mobiiliohjelmoinnin kurssi, jota olin jo pari vuotta sitten TTY:llä kaivannut! Alun perin aloitin materiaalitekniikalla ja ensimmäinen pakollinen ohjelmoinnin kurssi käänsi opiskelujeni suunnan. Päätin vaihtaa automaatiotekniikalle ja ottaa ohjelmoinnin sivuaineeksi. Nykyään pääaineeni on automaation tietotekniikka.

Kun Johdatus ohjelmointiin oli käyty, kaipasin kovasti lisää ja mielellään mahdollisimman nopeasti. En kesän tullessa voinut kuitenkaan käydä mitään kurssia, joten rupesin omatoimisesti opiskelemaan Android Studiota, lähinnä YouTube videoiden avulla. Varovainen arvio on, että kesällä tuli koodailtua 100-150 tuntia. Tämä oli siis kesällä 2015 ja sen jälkeen en ole juuri Android Studiota käyttänyt, muuten kuin yhtä työnhakuun liittynyttä projektia tehdessä.

# Harjoitukset

Tässä kappaleessa kerron harjoituksista, niiden tekemisessä vastaan tulleista haasteista ja toivottavasti myös kattavasti ratkaisuista. Harjoitusten varsinaiset ratkaisut löytyvät versionhallinnasta (https://github.com/IlkkaHannula/Mobiiliohjelmointi).

## Tutustuminen mobiililaitteeseen

Itsellä ei enää ole Android puhelinta, mutta kurssin hengessä halusin valita kuitenkin jonkin sellaisen. Vanha puhelimeni ei palvellut minua kovinkaan hyvin, joten päädyin valitsemaan edullisen, mutta ainakin hieman kehutun puhelimen. Tehtävää tehdessä tuli huomattua, että kyseistä kapistusta kannattaisi harkita, jos olisi uutta puhelinta hankkimassa.

## Git ja versionhallinta

Vaikka hyvin tiedostan, että versionhallinta on tärkeä ja koodaamista huomattavasti helpottava apuväline, tulee sitä käytettyä liian harvoin. Tähän tuli kuitenkin poikkeus, kun lisäsin sinne surffattavia aaltoja varten koodaamani seurantatyökalun raakileen. Koska Gittiä on tullut käytettyä melko harvakseltaan, oli komennot tietysti unohtunut täysin. Pääsin kuitenkin helpon dokumentaation avulla taas perille ja nyt koodin päivittely tuntuu taas vaivattomalta, kun sai aloitettua.

Tähän voisin päivitellä, että nyt kun tuota omaa sovellusta on tullut koodailtua melko paljonkin, on myös Git päivittynyt ahkerasti ja komennot ei juuri tuota ongelmia. Kansiota ”Aaltovahti” voi käydä kurkkimassa, jos kiinnostaa.

## Ympäristön asennus ja testaus

Koska ympäristö oli minulle jo ennestään tuttu, ei tästä tehtävästä ole oikeastaan muuta kommentoitavaa kuin hieman integroidusta versionhallinnasta. Android Studion terminaalin kautta homma näytti toimivan oikein mallikkaasti ja koska repositoriolle oli jo aiemmin määritelty remotet, oli push-komentokin suoraan käytettävissä.

## Perus käyttöliittymä

Tehtävä nyt on periaatteessa pelkkää komponenttien lisäilyä ja kohdistamista sekä pienen logiikan toteuttamista Java koodin puolelle. Käyttöliittymän tekeminen graafisessa näkymässä on suhteellisen puuduttavaa, kun komponentit tuntuvat menevän sinne ihan, miten sattuu. Toisaalta tekstinäkymässä se on ehkä vielä epätoivoisempaa ja näin ollen päädyinkin tekemään pääosin graafisellapuolella ja muokkaamaan komponentteja hieman tarpeen vaatiessa tekstieditorin kautta.

Tehtävän luonteen takia oli sekä käyttöliittymänäkymässä melko paljon toistoa ja vastaavasti myös koodilogiikassa. En keskittynyt tekemään hienoa toistotonta koodia, koska ajattelin tehtävän pihvin enemmänkin olevan perus juttujen hanskaaminen. Tehtävän aikana tuli vastaan pieniä ongelmia, muttei mitään isompaa. Google on kaveri ja syntaksi alkaa muistua paremmin, kun koodaa enemmän. Eiköhän tästä kohta vielä innostu.

## Tietokantasovellus

Esimerkkisovellus on varmaan oppimisen kannalta aivan mainio, mutta päädyin kuitenkin tekemään sovelluksen, mikä on hieman haastavampi. Pari vuotta sitten tuli tehtyä työnhakua varten Android Studiolla sovellus, mihin sai tallennettua kavereiden tilinumeroita ja niitä sitten pystyi myös muokkaamaan ja poistamaan. Tässä sovelluksessa oli kuitenkin vain käytössä nimet ja tilinumerot tietokannasta, mutta käyttöliittymä oli varsin hiottu ja olihan se myös hauska ruveta tutkimaan mitä aikaisemmin olikaan tehty.

Aluksi meni noin tunti tutustua vanhaan koodiin ja käydä se melko huolellisesti läpi. Samalla sain ajatuksen sovelluksesta, johon pystyy tallentamaan kavereiden merkkipäiviä. Näin ollen käytössä olisi kentät nimelle, syntymäpäivälle ja nimipäivälle, sekä tietenkin puhelinnumerolle, jotta onnittelemaan pääsee nopeasti. Tähän voisi myös yrittää järjestää tietoja tulevien merkkipäivien mukaan. Joitain muitakin lisäominaisuuksia tuli mieleen.

Vanhan ohjelman rakenne vaikutti hyvältä, mitä nyt pieniä muutoksia koodiin tuli heti tehtyä. Koodissa toteutus on tietysti aina jaettu kahteen, eli siis jotain näkyy käyttäjälle ja jotain on tietokannassa. Yleensä nämä myös vastaavat toisiaan, mutta joissain tapauksissa, kuten muokatessa oli järkevä tehdä ensin näkymään lisäys ja poisto ja sen jälkeen tietokantaan muokkaus. Näin järjestys saatiin säilymään oikeana. Jälkikäteen mietittynä tältä olisi vältytty ja toteutuskin olisi varmaan parempi, mikäli alun perin olisi ajateltu, että näkymässä olevat palikat ovat aina olioita, jotka vastaavat tietokannan yhtä riviä. Näin ollen oliolle olisi voinut kutsua muokkaa metodia, joka sitten olisi välittynyt näkymän elementille ja tietokannan alkiolle. Sitten olisi pitänyt vielä järjestää alkio oikealle paikalle.

Harjoituksessa tuli hyvin esille se, että vaikka tietokannan lisääminen itsessään ei ole kovinkaan hankalaa ja asian osaavalle se on todella suoraviivaista, olisi silti hyvä pysähtyä hetkeksi miettimään millä tavalla tietokanta linkittyy sovellukseen. Näin voisi päätyä toimivampiin ratkaisuihin tietokannan käsittelyssä ja välttää purkkaa, kun jotain pientä pitää virittää sovelluksen lisäksi. Muita suurempia ongelmia ei toteutuksessa sitten oikeastaan ollutkaan. Googlailtua toki taas tuli, mutta niin tulee aina kun koodaa.

Koska SQLite tietokanta ei tuo kuin paria tietotyyppiä, on tiedon konvertoinnista ja oikeellisuudesta käyttäjän huolehdittava itse. Tähän harjoitukseen en jaksanut tehdä kummempia oikeellisuustarkastuksia päivämäärille ja käytin suosiolla vain merkkijonoja tyyppiä tietokannassa.

## Firebase reaaliaikainen tietokanta

Ensisilmäykseltä tietokanta vaikuttaa suhteellisen helpolta vaihtaa sovellukseen. Tämä tietysti siitä syystä, että tietokanta on yksinkertainen ja sisältääkin vain 4 eri saraketta, joista kaikki edustaa merkkijonoja. Jsonin käsittelyn ei siis pitäisi olla ainakaan hankalaa.

Firebasen saa nykyiseltään liitettyä projektiin melko helposti suoraan Android Studiosta, tämä kuitenkin rikkoi aikaisemman toteutuksen, kun taas oli ladattava uusia SDK tooleja ja ilmeisesti se muutti asetuksia. Onneksi Google osasi taas auttaa. Google myös tarjosi tietokannalleen melko hyvää dokumentaatiota ja joku muukin oli sitä ennen käyttänyt.

Valitettavasti ei tietokannan vaihtaminen osoittautunut alkuunkaan helposti ja kun viimein olin saanut tehtyä koodin muutokset en oikeastaan päässyt ollenkaan kokeilemaan tietokantaa. Alussa vaikeuksia oli syntaksin kanssa ja ilmeisesti suojausasetuksetkin olivat menneet päin mönkään. Meni vaikka kuinka kauan ennen kuin sain lisättyä mitään dataa tietokantaan.

Kun tietokantaan sai lisättyä jotain, helpottui yrittäminen hieman. Webbikäyttöliittymästä kun pääsi kivasti näkemään lisätyn datan rakenteen. Vastaavasti myös Android Studion debuggerissa pääsi tutkailemaan muuttujia, mikä helpotti huomattavasti virheiden löytämistä. Esimerkkinä muokkauksessa jäi jostain syystä nimi muuttamatta, mutta virheen sain paikannettua hyvin sekä debuggerista, että webbikälistä. Toisaalta muokkauksen monistumiseen liittynyt ongelma oli ei ollutkaan tietokannassa ja ongelman sain ratkaistua hetkessä, kun huomasin eron debuggerin ja tietokannan välillä.

Tietokannan reaaliaikaisuuden vuoksi olisi ollut varmasti fiksumpaa käyttää RecycledViews ideologiaa, mutta koska aiemman harjoituksen toteutus ei perustunut siihen en viitsinyt tässä ruveta vaihtamaan. Ideana kuitenkin se, että tietokannassa jokin tapahtuma vaikuttaa heti sovelluksen näkymään on mielenkiintoinen. Riippuen tietokannasta ja ketkä sitä pystyy päivittämään, on joissain tapauksissa tärkeää käyttää RecycledViewseja. Tässä tehtävässä kuitenkin oma toteutustapani näytti toimivan erittäin hyvin, kun kokeilin vaihtaa kannan arvoja suoraan netistä. Jos kuitenkin sama kanta on useammalla käytössä ja muokattavissa, olisi mahdollista päätyä tilanteeseen, jossa sovellus menee rikki tai jopa kaatuu. Realistista olisi, että nappien näkyvyys voisi rikkoontua, mikäli toisella laitteella poistetaan viimeisiä tietoja samaan aikaan.

## Firebase autentikaatio

Sitten seuraavaksi vuorossa oli tapella sovellukseen jonkin lainen autentikaatio ja sen piti taas olla suoraviivaista. Ongelmia tuli kuitenkin taas jo ennen kuin pääsi edes aloitamaan, sillä edessä oli taas uusia SDK:iden asenteluja, Googlailua ja pään seinään hakkaamista, ennen kuin edes sai tarvittavat osaset importoitua ja aloitettua.

Periaatteessa itse toteutus onnistuu helpolla seuraamalla Moodlen ohjeeseen linkattua MOOC-kurssia. Tätä se ei kuitenkaan todellakaan ole, itse Android Studio näyttää kyntensä, kun mikään ei toimi ja koko ajan tulee ongelmaa ongelman perään. Hieman aina meni eteenpäin, mutta sitten piti taas Googlata uutta virheilmoitusta. Mitään oppimistakaan tässä ei oikeestaan tapahtunut, kun lopulta tuli seuratttua vaan tyhmänä mahdollisimman tarkasti ohjeita.

No lopulta sain tuon ihmeen kaupalla toimimaan ja hyvä niin. Ei kyllä tällä hetkellä oikeastaan hirveästi edes kiinnosta mistä se oli kiinni. Toivottavasti vaan en joudu uudestaan tekemään samaa myöhemmin tai jos joudun niin hommat onnistuvat.

## GPS-sijaintia hyödyntävä sovellus

Tässä harjoituksessa menin aiempiin eroten hieman matalammalta kohtaa aitaa. Itse harjoituksen pihvihän on GPS:n hyödyntäminen ja Googlelta karttapalvelun lainaaminen omaa sovellustaan varten. Lisäksi tuli myös kerrattua oikeuksia ja mitä kaikkea pitää muistaa, kun haluaa isompia valmiita kokonaisuuksia käyttöön.

Harjoitusta varten oli hyvät ohjeet Gitlabissa ja niitä seuraamalla pääsin helposti eteenpäin. Tosin parissa kohtaa joutui miettimään hieman enemmän, kun meinasi jäädä jumiin. Tämä oli ensimmäinen kerta, kun tulin harjoituksiin oikeasti tekemään (motivaation kera) ja helpotti kyllä paljon, kun sai kysellä. Jo ääneen pohtiminen auttaa kummasti!

Harkan toteutuksesta ei juuri ole sanottavaa. Itse GPS:n käyttäminen ja Google Mapsin ohjaaminen on melkoisen suoraviivaista, mutta perehtymistä vaativaa. Nyt on sitten hyvä pohja, jos jatkossa tulee tarvetta toteuttaa jotain tämän tyylistä tai haastavampaa.

# MOOC-kurssi

Tässä kappaleessa kerron valitsemastani MOOC-kurssista, jonka käyn kurssin suoritusta varten. Valitsin opettajien suosituksesta Googlen kurssin ”Developing Android Apps”. Koska suurin osa aiheista on minulle luultavasti jo ennalta tuttuja, käyn materiaalit läpi, mutta en aio tehdä tai valehdella tehneeni tehtäviä, joista en usko olevan oppimisen kannalta apua. Ennen kurssin aloittamista halusin palauttaa Android ohjelmoinnin mieleeni ja tein harjoitustehtävät 1-4, sekä aloitin tekemään tehtäviä 5-6.

## Lesson 1 – Create project

Käytiin perusasioita läpi, ei oikeastaan mitään uutta. Kurssi vaikuttaa järkevästi toteutetulta ja sääsovelluksen toteutus koko kurssin aikana on hauska idea. Pitää katsoa myöhemmin, josko sen toteuttaisi. Tältä kierrokselta riitti kuitenkin vain videoiden ja materiaalin läpi käynti.

## Lesson 2 – Connect to the Internet

Tässä tulikin jo vähän enemmän asiaa, eikä kaikki ollut ihan perusjuttuja. Urlin muodostaminen tuntui hölmöltä, mutta varmaan kätevä mikäli oikeasti tulisi muodostettua paljon erilaisia. Tuli myös sellainen fiilis, että tässä käsitellään tekijän mielestä parhaita tapoja ja aionkin suhtautua seuraavien kierrosten opetussisältöön. Varmasti opetettavat tavat on hyviä, mutta tuskin yksiselitteisesti parhaita. Pitää myös muistaa, että aina ei kannata ihan kaikkea hioa viimeiseensä varsinkin, kun kyse on opettelusta.

## Lesson 3 - RecyclerView

Tässä vaiheessa alkoi tulla jo hyödyllisen oloista materiaalia, joten virittelin tehtävät kuntoon ja tein niitä hieman samalla. Kuitenkin niin, että taas itsestään selvyyksiin en viitsinyt käyttää aikaa. RecycledView vaikuttaa sellaiselta, mikä olisi hyvä osata, mikäli mielii jotain scrollausnäkymiä rakentelemaan. Harjoituksissa 5&6 tätä voisi hyödyntää, mutta pitänee tsekata jo toteutettu versio, että onko siellä hitaita kyselyitä xml-puolelta.

Nyt sitten iski myös ongelmia, johtuen tehtävien valmiista rakenteesta. Koska minulla on ollut asennettuna Android Studio jo reilun vuoden ja aiemmin käyttämäni SDK:t ovat vanhoja oli edessä osasten päivittäminen, että saan tehtävien rungot kääntymään. Onneksi tämä meni suhteellisen vaivatta SDK:den osalta ja kun sai Android Studion päivitettyä, säilyi asetukset ja emulaattorit ennallaan.

Itsellä on aikaisemmin ollut vähän päänsärkyä toastien kanssa, kun ne on odotellut toisiaan ja näin ollen jotkin toiminnallisuudet on ollut tosi ärsyttäviä nopeasti tehtynä. Nyt tuli sitten siihenkin vastaus, että miten tämän voisi hyvin kiertää!

Itse kierroksen harjoitukset oli mielestäni fiksusti toteutettu ja pystyin ne tekemään lähes pelkästään TODO-kommenttien avulla. Hieman ehkä näissä on se vaara, että tulee vain tehtyä eikä juuri mietittyä.

## Lesson 4 – Intents

Tässä kierroksessa alku oli niin ikään tärkeää, mutta myös erittäin simppeliä ja tylsää. Omat tarpeet näkymästä toiseen siirtymisessä olen aiemmin täyttänyt eri tavoilla, ehkä kuitenkin käsitellyillä periaatteilla tulee jatkossa toimittua. En nyt kuitenkaan vaivautunut tekemään tehtäviä, koska ne vaikuttivat hieman turhan simppeliltä TODO-kommenttien seuraamiselta.

Loppuosa kierroksesta olikin pääasiassa visailua ja muuta sisältöä ei tullut juuri lisää. Tyydyin nyt vain tehtävien ratkaisujen tsekkaamiseen, kun harjoitus 7&8 tekemiseen menikin luvattoman paljon aikaa. Palataan asiaan tämänkin kierroksen opetussisällön kanssa sitten kun näitä pääsee kokeilemaan oikeaa sovellusta toteuttaessa niin jää varmasti paremmin mieleenkin.

## Lesson 5 – Lifecycles

Jatketaan siitä mihin edellisellä kierroksella jäätiin ja jo ensimmäisistä videoista voikin arvata mihinkä ollaan menossa. Kun ollaan koodaamassa softaa Androidille, on tärkeä tiedostaa, että ehkei se sovellus ole käyttäjällä koko ajan ajossa vaan se voi unohtua taustalle, jos vaikka kaveri lähettää mielenkiintoisen viestin. Vastaavasti unohtuneeseen sovellukseen voidaan yllättäen palata tai joku kesken oleva sovellus voidaan tappaa. Kokeneelle kehittäjälle tämmöiset on ihan pala kakkua. Täytyy kuitenkin sanoa, ettei meikäläiselle ainakaan tullut nämä mieleen ennen kuin rupesin yrityksen ja erehdyksen takia näitä tarkemmin miettimään ja tutkimaan.

Näihin jo aiemmin huomiota kiinnittäneenä ei kuitenkaan tuntunut mielekkäältä käyttää aikaa alun tehtävien hakkaamiseen läpi, mutta seurasin ne silti läpi. Mikäli sovelluksen tulee sallia puhelimen kääntö pitää tilan säilyttämiseen kiinnittää huomiota, mutta toisaalta on myös vaihtoehto kieltää kääntäminen mikä on järkevä toimintamalli useissa sovelluksissa. Kierroksen opetus sisältö on mielestäni siis kategoriaa hyvä tiedostaa, mutta ei välttämätöntä osata tehdä.

Koska satuin tekemään Firebase tehtävät juuri ennen tätä kierrosta, tuli väkisinkin mieleen, että siellähän me toteutettiin ratkaisuja, jotka voisivat olla toimivia myös Async taskien tilalle. Tähän tietysti edellytyksenä sellainen, että Firebase tietokantaa voitaisiin hyödyntää rajapintahaun sijasta. Tämä varmasti joissain tapauksissa toimii, jos rajapinnasta haettava tieto on kevyttä ja se on mahdollista sekä järkevää siirtää Firebaseen. Huomiona, että tallainen voisi hyvinkin toimia sääsovelluksessa, jossa haetaan hieman historiallista sää dataa.

Tein sitten videon ”perässä” tuon jo toisella kierroksella nähdyn toyappin parantelun ja täytyy kyllä sanoa, että tuota en yritä enää toiste. Videolla selkeästi koodaaminen tapahtuu eri järjestyksessä ja joitakin vaiheita jättäen välistä, kun vertaa tehtävässä mainittuihin TODO-kommentteihin. Tästä johtuen tehtävän viimeisteleminen oli todella ikävää, kun emulaattoristakaan ei tule tarkkaa virhettä ulos. Myös Android Studiolla samaan aikaan harjoitus ja ratkaisu projektien aukipitäminen, sekä emulaattoreiden ajaminen ei tunnu toimivan kovin saumattomasti, kun projektit tuntuvat aukeavan emulaattorin, miten sattuu, jos ollenkaan. Ehkä Googlen olisi syytä kehittää vähän ajettavan projektin tilan näyttämistä yms. jolloin tiloista oppiminen olisi heidän alustallaan uskottavaa. Erityisesti login tulosteiden näkeminen olisi sovelluskehittämisessä erittäin tärkeää.

## Lesson 6 – Preferences

Settings menua lisätessä tuli uutena parent ja child activityt, joiden avulla vältetään turhat päänäkymän uudelleen lataamiset. Hyvä huomio, jonka tarpeeseen olisi ainakin tullut törmättyä jossain vaiheessa. Tässä myös esiteltiin toinen tapa siirtyä activityjen välillä, joten aiemmin esitelty ”Intents” kuulostaa entistä harvemmin käytettävältä.

Seuraavaksi ”PreferenceFragments” esitellään erittäin hyvin, jolloin saa kuvan millaisia asetusvalikkoja sen avulla voi toteuttaa, sekä hieman esimakua, että kuinka helposti. Joskin tässä yhteydessä neuvotaan käyttämään xml-koodiin määriteltyjä merkkijonoja kaikkialla. Varmaankin ihan hyvä ajatus, mutta etenkään ketterässä ohjelmistokehityksessä ei tuo ole mielestäni käytännöllistä. Kun koko ajan vähän muutellaan koodia, on muutokset helpompi tehdä nopeasti yhteen paikkaan, sen sijaan, että määrittelee ensin vakion, jonka avulla muutoksen voi tehdä.

Monet mobiilisovellukset pohjautuvat vahvasti tapahtumapohjaisuuteen ja usein tiettyyn toimintoon olisi myös mukava reagoida heti, eikä esimerkiksi pienellä viiveellä. Halutaan siis saada herätä, kun jotain tapahtuu, eikä turvautua mekanismiin, mikä tietyin väliajoin käy kyselemässä muuttujien tiloja. Jo tässä vaiheessa MOOC-kurssia on ehditty esitellä monta erilaista kuuntelijaa. Itse en niinkään välitä yksittäisistä esittelyistä, kun nuo tiedot löytyvät melko helpolla netistä. Sen sijaan tässä kierroksessa tuli taas hyvää kertausta toteutuslogiikasta kuuntelijoiden suhteen.

Päädyin siihen, että teen sääsovelluksen lopuksi kaikkien kierrosten jälkeen ja kirjaan tästä täysin oman kappaleeseensa. Sovelluksessa käsitellään monia tärkeitä aiheita kootusti, joten se on hyvä testi lopuksi, että kuinka paljon kierroksista on oikeasti jäänyt käteen.

## Lesson 7 – SQLite

Tässä vaiheessa on tullut jo tehtyä harjoitus koskien SQLiteä, eli kierrokselta en odota juuri oppivani siitä uutta, mutta ehkä vahvistavani vanhaa osaamista. Ehkä kuitenkin kierros kattaa myös muita asioita, joista voi sitten olla enemmän hyötyä.

Yksi näistä muista asioista on yksikkötestit, joista Androidin osalta minulla ei vielä ole kokemusta. Ensimmäisen tehtävän toteutuksen tarkastelussa käytetään yksikkötestausta varmistamaan, että tietokannan rakentaminen on ymmärretty oikein. Tätä ei ole juuri avattu videoilla, mutta koodin puolelta on helppo kurkata, että testien rakenne vastaa muiden ympäristöjen yksikkötestausta. Testeistä kuitenkin näkee, että ne on tehty tässä yhteydessä neuvomaan missä virhe on, eikä oikeasti testaamaan. Seuraavaan tehtävään löytyykin sitten testit, jotka oikeasti testaavat tuotettua koodia.

Kun tehtävissä päästään eteenpäin, tulee hyvää kertausta RecycledViewseista ja adaptereista. Koska kyseessä on tärkeä asia, on kiva nähdä lisää esimerkkejä suunnittelumallin toiminnasta. Ja nyt siihen vielä yhdistetään itemTouchHelperin käyttö. Elementti on siitä syystä erittäin kiinnostava, että sen avulla listasta saa helposti esimerkiksi poistettua osia vetämällä sivulle. Tämä onnistuu tekemättä itse mitään animaation eteen ja näin ollen muutamalla rivillä koodia on mahdollista toteuttaa totutulla tavalla käytettävä käyttöliittymän osa, joka voisi olla ihan oikeassakin sovelluksessa.

## Lesson 8 – Content Providers

Noniin, nyt tuleekin täysin uutta asiaa ja vielä tärkeää sellaista. Ainakin itseäni ärsyttäisi käyttää jotain sovellusta, joka tallettaa jo puhelimesta löytyvän datan kaltaista dataa omaan tietovarastoonsa. Usein se myös tarkoittaa, että sovellus käyttää ehkä vain omaa tietovarastoaan, mutta ainakin sitouttaa käyttäjää typerästi sovellukseen, koska tähän kerättyyn tietovarastoon ei ole pääsyä muualta.

Ikävä kyllä mielenkiinto kierroksen aiheeseen saatiin hyvin karkoitettua typerällä esimerkillä. Voi ehkä olla, että helppoa ja järkevää esimerkkiä on vaikea löytää, mutta tallainen esimerkki on mielestäni vain kämänen toteutus kiertää ”middleManClass” ja tietokannan käyttö. Keksityn Content Providerin ansiosta nimittäin esitellään ”paras” käytäntö aika kaukaa haetusti ja teeskennellään, että se oikeasti olisi paras.

Okei, ehkä tuossa esimerkissä on sittenkin hieman järkeä. Mikäli nyt ymmärsin oikein, voisi olla tapaus, jossa on luotu Content Provider, jonkin sovelluksen käyttöön. Tämä sovellus ei kuitenkaan miellytä kaikkia ja kilpaileva firma tekee sovelluksesta oman toteutuksensa, joka miellyttää vanhaa käyttäjää. Nyt jos tilanteessa on voitu käyttää Content Providereita, on mahdollista, että ensimmäisen sovelluksen käyttäjä siirtyy käyttämään uutta kilpailijan sovellusta ja kaikki data säilyy. Tuo käy järkeen, mutta veikkaanpa, ettei tallaisia tapauksia ole paljoa, joissa jonkin firman luoma Content Privider vakiinnuttaa asemansa.

## Lesson 9 – More Content Providers

Tässä vaiheessa lienee varmaan jo selvää, mitä mieltä olen oman Content Providerin tekemisestä. Ei välttämättä mitään kaikista hyödyllisintä osata… Voi tosin olla, että olen väärässä ja nyt on jäänyt tärkeä juttu oppimasta, kun käyn aiheen pelkästään tutkailemalla tehtävät läpi. Taustalla on hieno ajatus, että kuka vain voi tehdä Content Providerin, mutta haluaisin kyllä kuulla hyvän esimerkin, milloin sellainen oikeasti kannattaa tehdä? Oma paras arvaus on, jos tehdään useampi sovellus tukeutumaan samaan tietokantaan ja toteutus saattaa muuttua.

Hieman lisää URIsta ja tällä kertaa vähän yleisemmin kuin aiemmin. Myös avataan ajatusta Content Providerin jakamisesta käytettäviin ”näkymiin” mikä toisaalta koskee myös käyttäjää, eikä vain tekijää. Myös URIMatcher tuli uutena asiana ja on mielenkiintoinen, sillä sen käyttötapaukset ylettyvät varmasti myös Android kehityksen ulkopuolelle.

Testaaminen on kyllä tärkeä taito, mutta myös vaikea. Oikeasti koskaan ei anneta hyviä testejä, jotka auttavat korjaamaan koodissa pienetkin virheet. Sitten taas, jos ja kun testit tehdään itse, on niiden tekeminen toimiviksi äärimmäisen vaikeaa. Saattaa olla, että kun luulee testin toimivan, onkin testannut jotain ihan muuta ja ikään kuin luonut testit vastaamaan toteutettua koodia.

## Lesson 10 – Background Tasks

Tässä on ehkä mielenkiintoisin aihe, mitä tällä kurssilla on tullut vastaan tai on tulossa. Kun mietin sovelluksia, joita voisin itse haluta omaan puhelimeen, on niistä suurin osa juuri taustalla pyöriviä. Mielellään myös sellaisia, jotka hoitavat hommansa, vaikka unohtaisi niiden olemassa olon. Näistä todellakin haluan oppia niin paljon, että hyvän idean tullessa voisin sovelluksen toteuttaa!

Toinen erittäin kiinnosta aihe liittyen taustalla pyöriviin tehtäviin on ilmoitukset. Kuinka ollakaan aika paljon noista sovelluksista, mitä voisin haluta, perustuu myös ilmoituksiin. Jos jotain tehdään taustalla niin, että jotain mielenkiintoista tapahtuu, on siitä ehkä hyvä myös viestiä käyttäjälle. Tämän jälkeen ehkä sovelluksen käyttöliittymä alkaisi kiinnostaa tai haluaisin sen ohjaavan minut nettiselaimeen tai vastaavaa. Kohta opin, että miten.

Seuraavana tulee vielä ratkaisu siihen, että miten noita taustalla toimivia tehtäviä saadaan aikataulutettua ja ehdotettua. Mikä ei sinänsä yllätä, sillä harvoin taustalla tapahtuvat asiat ovat vain yhtäjaksoista laskemista. Kulku toteutukselle tuntuu olevan melko suoraviivainen, vaikkakin siinä on aika paljon asiaa. Tuskin jää täysin mieleen, mutta näistä tehtävistä saa hyvät esimerkkitoteutukset omia sovelluksia varten. Itse koodihan ei ole juurikaan erikoista, mutta sitä pitää muistaa tuottaa useaan paikkaan. Kun tehtävää tekee niin ei saa kunnolla kokonaiskuvasta kiinni, sillä TODO-kommentit jo itsessään ohjaavat aina oikeaan paikkaan. Näin ollen ainakin itsellä jäi tuo miettiminen, että mitä ja minne vähän turhan vähälle.

## Lesson 11 – Completing the UI

Hieman jo ehdinkin kummastella, kuinka vähän tässä kurssissa keskitytään käyttöliittymän tekemiseen. Käyttöliittymät kun kuitenkin on mobiilisovelluksissa aina se mikä pistää silmään ja mistä yleensä myös koko ensivaikutelma muodostuu. Käyttöliittymien tekemisen ainakin itse olen oppinut vaan hakkaamalla niitä. Tietysti joskus on katseltua paljonkin opetusvideoita, mutta eipä se perässä tekeminen opeta läheskään niin paljoa, kuin oikeasti itse miettiminen, kokeileminen, epäonnistuminen ja lopulta onnistuminen.

Data Binding tuli uutena asiana ja aluksi se herätti mielenkiinnon. Se esiteltiin mahtavana asiana, mutta sen hyödyt jäivät kertomatta. Lopulta siitä jäi sellainen kuva, että se ei ehkä kuitenkaan juuri vähennä koodin määrää tai työtä arvojen sijoittamisessa tai päivittämisessä. Ehkä sääsovelluksessa tämän toteuttaminen avaa aihetta paremmin, kun pääsee sitä käyttämään oikeassa tapauksessa.

Aiemmin on tullut törmättyä ja vähän kokeiltuakin sovelluksen lokalisointiin, mutta esteettömyys on täysin uusi termi. Ehkä hyväkin niin, sillä ei tässä vielä mitään killerisovellusta ole tehty, joten ehkä on hyvä keskittyä olennaisiin asioihin. Realistisempaa olisi tarve kääntää sovellus eri kielille, mutta homma on sen verran simppeli, että se sujuu helposti ensin palauttelemalla asiat mieleen ja sitten tekemällä. Toisaalta taas oikealta vasemmalle kirjoitettaville kielille kääntäminen tuskin on realistista.

Jotain järkevääkin tuli vielä, kun esiteltiin landscape näkymälle erilaisen järjestelyn tekemä ja käyttöliittymän pilkkominen ja rakentaminen eri xml-tiedostoista. Hyödyllisiä juttuja, joskin hieman turhauttavaa opetella harjoituksissa, koska nämä hiomiset on yllättävän aikaa vieviä.

## Lesson 12 – Polishing the UI

Noniin nyt tuli vakuuttavaa motivaatiopuhetta, siitä mitä varten käyttöliittymä hiotaan ja mitä se käyttäjäkokemus olikaan. Toivottavasti kierros on sitten myös sen mukainen. Vielä välikevennyksenä se, että mielestäni useat Androidin videollakin nähdyt omat tai Googlen sovellukset ovat hävyttömän rumia, nyt sormet ristiin, ettei opita sellaisia tekemään.

Itse en henkilökohtaisesti ole kovin suuri fani alussa esiteltävälle tavalle valita sovelluksen värit. En myöskään tykkää hirveästi terävistä kulmista napeissa tai täysin tyhjästä taustasta. Käyttöliittymän hienous on toki mielipide kysymys, mutta tuntuu hassulta saada ohjeita, kuinka tehdä hyvä käyttöliittymä, kun lähtökohtana tuo ”hyvä” on juuri sellainen mistä en itse tykkää. Toisaalta oppiihan tässä pieniä knoppeja elementtien muokkaamiseen, jotka tosin löytyisivät Googlen avullakin sitten kun on tarve.

Tyylit ja teemat menevät aika suoraviivaisesti ja samoin kuin esim. webbikehityksessä on totuttu tekemään. Eri näyttöjen handlaamisesta oli mulla aiempaa kokemusta, mutta ilmeisesti onkin olemassa kätevämpi tapa! Vaikka tätä ei mainittu niin ratkaisee tuo myös sen ongelman, että ainakin itse miellän puhelimen näkymän normaalisti pystyyn ja tabletin sivuttain. Kun mietitään pienintä leveyttä, ei tätä tarvitse ottaa huomioon ja erilaisella landscape näkymällä saa kaikki mahdolliset kompinaatiot otettua huomioon. Mikäli jokin sovellus tulisi joskus julkaistua Play-kauppaan näistä olisi varmasti hyötyä!

## Sunshine App

Kävin aluksi kurssin sisällön läpi ja rupesin sitten vasta tekemään kurssin ”projektina” toimivaa sääsovellusta, minkä ainakin pitäisi kattaa koko kurssin sisältö. Ajatuksena olisi, että näin oppisin parhaiten, saisin vähän kertausta ja harjoituksen laatukin olisi varmaan muita tehtäviä parempi.

Tässäkin projektissa on tehtävät jaoteltu kierrosten mukaan, mikä on ihan hyvä, niin tietää heti suunnilleen tehtävän aihepiirin. Nyt tehtävät vaikuttavat hieman laadukkaammilta, joskin muutama niistä on erillisiä, eikä ne siis tule oikeasti sovelluksen. Mutta toki tukevat oppimista ja osa näistä on myös ihan perusteltuja välivaiheita koodaamisen kannalta.

Mitä pidemmälle tehtävissä meni, sitä paremmin niistä saattoi jotain oppia. Vastaavasti tehtävien taso kasvoi ja niitä alkoi olla hankalampi tehdä. Useasti aihe olisi ollut helpompi sisäistää, jos olisi ollut hieman enemmän tietoa TODO-kommenttien kanssa. Tässä nimittäin kävi aina välillä niin, että ei oikeastaan ollut hajua, miten jokin syntaksi menee ja piti katsoa tehtävään vastaus, missä sitten olikin 5-10 riviä koodia.

Sovelluksen tekemisestä oli kyllä hyötyä, mutta asioita on vaikea painaa mieleen niin, että voisi tehdä vastaavia toiseen projektiin ilman mitään referenssiä. Oppimisen lisäksi tästä projektista saikin hyvän referenssiprojektin, josta voi jatkossa ottaa mallia.

## Mietteet MOOC-kurssin jälkeen

Verrattuna Firebasea koskevaan MOOC-kurssiin jäi tästä kokonaisuudesta hieman pettynyt fiilis. Aiheita lopulta käytiin kyllä mukavasti läpi ja näiden pohjalta on helppo visioida ensin sovellus jonka pystyisi toteuttamaan ja sitten käydä töihin, mutta jotain jäi hampaankoloon. Vaikka asiat tuli esiteltyä lopulta lähes poikkeuksetta hyvin, ei mielestäni toteutus ollut kovinkaan johdonmukainen ja itse olisin odottanut parempaa. Välillä tuntui, että kun aihe esiteltiin ja käytiin läpi, oli se silmiä avaava, mutta punaisen langan unohtanut esitys. Vastaavasti myöhemmin, kun aihetta sivuttiin, toisen aiheeen mukana olikin tuo punainen lanka jostain kummasta löytynyt ja aihe oli mahdollista sisäistää paremmin kuin ensimmäisellä kerralla. Nyt en myöskään puhu hyvästä kertauksesta vaan päinvastoin.

Pienellä uudelleen järjestelyllä ja paria aihetta paremmin miettimällä olisi toteutuksesta saanut loistavan hyvän sijaan. Kurssi sisälsi hyvin perusasiat ja antoi ymmärryksen ja osaamisen käyttää niistä helppoja juttuja juuri miettimättä. Vaikeampien asioiden suhteen on edessä varmasti paljon Googlaamista, mutta nyt pitäisi olla hyvin selvillä, että mitä on mahdollista toteuttaa ja miten se kannattaa tehdä.

# harjoitustyö

Aluksi ajatuksena oli jatkaa jotain vanhaa projektia, mutta päädyin kuitenkin vaihtamaan aihetta kokonaan uuteen. Näin harjoitustyölle löytyy paremmin motivaatiota ja muutenkin puhtaalta pöydältä on aina mukavampi aloittaa. Työn ollessa täysin uuden kehittämistä on helpompi valita suoraan kunnolla hyvät tekniikat ja koittaa välttää turhaa toistoa.

## Idea ja posteri

Usein sääennusteissa saattaisi olla jotain mielenkiintoista tulossa, mutta toisaalta kukapa niitä jaksaa tarkastella? Tätä varten onkin hyvä ajatus koodata sovellus, joka tekee sen puolestasi ja ilmoittaa kun esimerkiksi sopivat olosuhteet surffaukseen voisivat olla tulossa. Sanottakoon vielä, että olen myös toteuttanut nettisovelluksen, jonka avulla pystyn tarkastelemaan ennustedatoja kivasti visuaalisessa muodossa.

Tarkoituksena on siis tehdä sovellus mikä toimii taustalla puhelimessa sekä hakee ja tutkii sää ennusteita. Sovellus sitten lähettää ilmoituksia sitä mukaan, kun käyttäjän olisi syytä tarkastella niitä itse. Sovelluksessa pääpaino on siis laskennassa ja käyttöliittymän osuus on vähäinen.

Posteria tehdessä idea selkeytyi vielä entisestään ja posteri oli haastava tehdä, kun ei pääse paikalle esittelemään sitä. Siispä posterin pyrin tekemään niin, että jo sen lukemalla saa pienen käsityksen, että minkälaista koodia työhön voisi olla tulossa. Näin ollen posteriin tuli koottua ylös työssä käytettäviä tekniikoita ja hieman myös yleistä toteutusta idea tasolla. Posteri löytyy myös Gitistä.

## Työn aloitus

Ainakin omalta osalta voin sanoa, että koodaamisen aloittaminen on aina vaikeinta. Sitten kun on saanut homman aluilleen, on helppo jatkaa tekemistä ja varsinkin jos hommat sujuvat välillä saa käsittämättömän paljon aikaan. Nyt ei ollut kuitenkaan siitä vaaraa.

Koska sovelluksessa ei ole varsinaista käyttöliittymää tai ainakaan tuo käyttöliittymä ei ole keskeisessä roolissa oli hankala aloittaa senkään tekemisestä. Tein kuitenkin testailua varten käyttöliittymän, johon laitoin napin ja tekstikentän. Näillä oli tarkoitus päästä testaamaan ennusteiden hakemista netistä.

Samaa dataa tulee käytettyä myös nettisovelluksessa, joten rajapinnat ei onneksi vaatineet sen enempää tutustumista. Valitsin sovellukseen käytettäväksi Tanskan säälaitoksen REST-rajapinnan, josta tulee kätevästi suoraan JSON muotoista dataa. Toinen rajapinta, jota käytän, on auringon nousu- ja laskuaikoja varten, koska en halua, että sovellus lähettää ilmoituksia, kun surffattavaa olisi pimeän aikaan.

Logiikan tekeminen data hakua ja käsittelyä varten, sekä Javan käyttäminen (Pythonin ja Javascriptin) sijaan tuotti hieman päänvaivaa, mutta ei kuitenkaan ylitsepääsemättömästi. Isoimmaksi ongelmaksi alkuvaiheessa osoittautui kuitenkin taas Android Studio ja sen emulaattori. Jostain syystä emulaattorin kanssa oli todella paljon ongelmia, enkä saanut, sillä missään vaiheessa testailtua luomustani. Täytyy siis toivoa, että mitään isompia ongelmia ei koodissani ole tai jos on, niin pystyn selvittämään ne vielä emulaattorin avulla.

Kun sitten vihdoin sain emulaattorin toimimaan (lataamalla uuden SDK:n ja käyttämällä kokonaan uutta emulaattoria). Alkoi pienien virheiden metsästys. Tässä ei kyllä helpota yhtään se, että jokaista testi ajoa joutuu emulaattorin takia odottamaan jonkin aikaa. Toisaalta uusi emulaattori vaikutti toimivan hieman paremmin verrattuna aiempiin.

## Ennusteiden haun ajastaminen

Päädyin käyttämään tässä FirebaseJobServicea, jota käytettiin myös valitsemallani Mooc-kurssilla. Kurssilla toki kehuttiin palvelua ja myös muualtakin netistä löytyy vakuutteluja, että tätä kannattaisi ehdottomasti käyttää muiden vaihtoehtojen sijaan. Tässä minun sovelluksessa olisi itseasiassa nuo muut ehkä jopa parempia, mutta käytetään nyt tätä tuttua, kerran tämä olisi hyvä osata.

Heti alussa tuli mieleen vanha havainto harjoitusten ajalta. Harjoituksia tehdessä sivuutin havainnon miettimättä sitä pidempään, mutta nyt tässä vaiheessa tuo rupesi vaivaamaan. Nimittäin tässä tekniikassahan ei suoranaisesti päästä ajastamaan, että jotain tapahtuu 15 minuutin välein tai vaikka aina kun kello on tasan jotakin. Tapahtuman laukaisijana, kun käyttää tuota suoritusikkunaa, voi lopputulos olla mielenkiintoinen. Omassa sovelluksessani tällä ei ole niin suurta väliä, mutta mielestäni 15 minuutin välein ilmoittava nesteytyssovellus hieman menettää merkitystään, jos se on vain sinnepäin.

Tuon edellä mainitsemani suoritusikkunan toteutumista on myös hankala seurata, sillä kun asetin välin nollaan ja kahteenkymmeneen, oli odotukset taskien välillä kuitenkin jotain 2-3 minuuttia. Tähän kun lisää vielä typot ja muut ongelmat koodissa on sulavaa ohjelmaa vaikea saavuttaa.

## Ennusteista ilmoittaminen

Varsinaisesti mikään osa tässä harjoitustyössä ei ole hirvittävän hankala, mutta kaikkeen tuntuu kuitenkin uppoavan melko paljonkin aikaa. Tähän poikkeuksena ei olleet notifikaatiot. Isoimmaksi ongelmaksi paikantui lopulta käsite ”context”, josta täytyy sanoa sen verran, että en juuri tiedä mitä se sitten lopulta tarkoittaakaan. Koska sovellusta on tietenkin hyvä jakaa useampaan pienempään luokkaan tämä tarkoittaa, että tuota contextia pitää osata pallotella läpi sovelluksen, mikäli haluaa pelata varman päälle.

Katsoin sekä MOOC-kurssin oppaasta, että parista muustakin tutoriaalista ja siellä context oltiin saatu haettua ajonaikaisesti, tämän yrittäminen tuotti kuitenkin omassa koodissani ongelmia ja ennen kuin lisäsin notifikaatiolle testaus-napin ei nämä ongelmat jostain syystä tulleet edes näkyviin. Vaikka kyseisille virhetilanteille ei ollut edes toteutettuna virheenkäsittelyä ei virhe tullut ikinä ilmoille, mutta ei kyllä myöskään notifikaatiokaan.

Onneksi sitten notifikaatioiden lähettämisen rajoittaminen onnistui melkoisen näppärästi. Koska Android tarjoaa shared preferenced käyttöön voi suosiolla jättää pienen SQL-tietokannan toteuttamisen suoraan ajatuksen tasolle, mitäpä sitä suotta tekisikään tietokantaa, jossa on taulu tai pari muutamaa arvoa varten. Uskomatonta myös, että shared preferencejen käyttäminen onnistui mukavan saumattomasti ja notifikaatioiden rajaaminen onnistui lähes hetkessä.

Toisaalta samalla puuhastellessa tämän kanssa antoi Android Studio oman työpanoksensa, kun emulaattori meni sillä tavalla solmuun, ettei netti toiminut. Tätä oli yllättävän vaikea paikantaa, vaikka virheilmoitus antoi suuntaa. Lopulta kun kokeilin emulaattorissa Chrome-selaimen toimintaa, tajusin, että vika oli nettiyhteydessä. Tätä en kuitenkaan saanut korjattu muulla tavalla, kuin tekemällä uuden emulaattorin, jolla lähtökohtaisesti asetukset olivat kunnossa.

## Viimeinen rutistus

Vaikka olinkin jo käyttänyt aika paljon harkkatyöhön aikaa, oli palautuksen lähestyessä vielä sellainen fiilis, että eihän tämmöistä kehtaa palauttaa. Onneksi kuitenkin hiomalla pikku seikkoja saa yleisilmettä oikein hyväksi. Runko ja kaikki tuntui olevan oikein hyvin hallussa ja tavoite, jonka takia sovellusta lähdin tekemään, täyttyi oikein hyvin.

Lopussa tuli hieman leikittyä broadcastReceiverien kanssa ja lopulta myös sain hoidettua boottauksessa käynnistämisen niin, ettei käyttöliittymä jää täysin tarpeettomana näkyviin. Lopun pitkästä urakasta ei ole juurikaan muuta sanottavaa, sillä kaikki ongelmat ratkesivat nopeasti stackoverflown avulla ja sovelluksen laatu parani oikein silmissä.

Toivottavasti sovellusta arvioitaessa huomioidaan sovelluksen tarve ja kiinnitetään huomiota myös notifikaatioiden avulla tapahtuvaan ”kommunikointiin”, sillä sovelluksen varsinainen käyttöliittymä jäi tarkoituksella todella vaatimattomaksi. Jos minä sovellusta tulen käyttämään, en aio kovinkaan usein tai kauaa viettää aikaa tuon käyttöliittymän äärellä. Käyttöohje gitissa posterin kanssa samassa dokumentissa.

# Ajankäyttö

Koska kurssin suorittaminen on ollut mahdollista etätyöskentelynä, on tästä tietenkin tullut otettua hyöty irti. Vastaavasti tämä on tarkoittanut, ettei opiskelu ole ollut aina kovinkaan johdonmukaista ja välillä on tullut käytyä useita asioita läpi rinnakkain ja joskus jopa keskityttyä osittain johonkin muuhun asiaan. Tämä vaikeuttaa ajankäytön arviointia, mutta tähän on koottu jonkinlainen arvio, kauanko eri osiin on mennyt.

|  |  |
| --- | --- |
| Ennen kurssia tapahtunut Android Studion opiskelu ja sovellusten tekeminen | 100+ tuntia |
| Harjoitukset 1-4. | 8 tuntia |
| Harjoitukset 5&6 | 6 tuntia (&15+ tuntia aiemmin) |
| Harjoitukset 7&8 | 12 tuntia |
| Harjoitukset 9&10 | 4 tuntia |
| MOOC- materiaalit ja harjoitukset | 40 tuntia (1-10. kierroksilta) |
| Harjoitustyö | 30 tuntia |
| Oppimispäiväkirja | 10+ tuntia |