

Ilkka Hannula

PLA-32820 Mobiiliohjelmointi

Oppimispäiväkirja

SISÄLLYSLUETTELO

[1. Johdanto 1](#_Toc506761406)

[2. Harjoitukset 2](#_Toc506761407)

[2.1 Tutustuminen mobiililaitteeseen 2](#_Toc506761408)

[2.2 Git ja versionhallinta 2](#_Toc506761409)

[2.3 Ympäristön asennus ja testaus 2](#_Toc506761410)

[2.4 Perus käyttöliittymä 2](#_Toc506761411)

[3. harjoitustyö 4](#_Toc506761412)

# 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 onnistuu.

# 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

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

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

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.

# harjoitustyö

Harjoitustyön aiheeksi ajattelin sovelluksen, jonka avulla voisi helpottaa simppelien yhtälöiden ratkaisemista. Sovelluksen ideana olisi, että käyttäjä voisi ensin syöttää jonkun helpon yhtälön ja sen jälkeen alkaa siirrellä termejä. Samalla kun käyttäjä siirtää termejä sovellus auttaisi matematiikan kanssa ja keräisi logia siirroista.

## Alkutilanne

Kyseessä on sovellus, jota olen useampaan otteeseen lähtenyt koodaamaan, mutta toistaiseksi en ole päätynyt itseäni tyydyttävään lopputulokseen. Toiminnallisuutta on todella vaikea ruveta rajaamaan, sillä jokaisesta yhtälön mahdollisessa liikkeestä on mahdollista tehdä jotenkin vielä vaikeampi, jos ja kun osoittajia ja nimittäjiä onkin enemmän kuin yhdet molemmilla puolilla. Rekursiosta voisi olla apua tämän ongelman ratkaisemiseksi, mutta toistaiseksi en ole siinä vielä onnistunut. Tässä harjoitustyössä tavoittelen, että saan aikaiseksi version, jolla pystyy ratkaisemaan näppärästi yksinkertaisia yhtälöitä, mutta joka voi mennä solmuun liian monimutkaisia yrittäessä.

Ensitöikseni rupesinkin katsomaan läpi eri versioita sovelluksesta, joita olen toteuttanut. Suurin versio on todella ruma raakile, jossa toiminnallisuudet on pisimmällä, mutta koodi on mahdotonta ylläpitää. Ajattelinkin käyttää sitä lähinnä referenssinä ja ottaa mallia toteutettujen toimintojen toiminnasta, kurkkaamatta vähääkään koodiin. Työnhakua varten olin joskus aloittanut kirjoittamaan toista versiota sovelluksesta, missä keskityin koodin ylläpidettävyyteen. Taidan ruveta työstämään harjoitustyötä kyseisen projektin pohjalta.