

Jami Kulmala

# **COMP.CS.220 – MOBIILIOHJELMOINTI**

Oppimispäiväkirja

## Harjoitus 1

Apple Iphone 12 mini

<https://www.apple.com/fi/shop/buy-iphone/iphone-12>

<https://www.apple.com/eg/iphone-12/specs/>

Käyttöjärjestelmä uusin IOS 16.5

Tärkeimpinä ominaisuuksina: A14 Siru, 6-ydin prosessori, 4-ydin grafiikkasuoritin, kaksois 12mpx kamera ja Face ID.

Käyttöjärjestelmä käyttää Applen luomaa Swift-ohjelmointikieltä ja myös valtaosa IOS-sovelluksista on Swiftiä.

Omia IOS-aplikaatioita voi luoda Xcode IDE:ssä, joka on Applen oma kehitysympäristö. Xcode tukee Swiftiä, joka tulee siis olla hallussa halutessaan luoda oma sovellus Iphoneen. Iphone ei tue latauksia App Storen ulkopuolelta, joten käytännössä ainoa sovelluksen julkaisu vaihtoehto on App Store. 1

Iphone 12 mini tulee Applen omalla lightning liitännällä, usb-c:n sijaan. Ominaisuuksia, joita Applen laitteista löytyy, joita Androidista ei löydy ovat esimerkiksi: AirPlay, FaceTime, Subjektin erittely ympäristöstä kuvissa, raahaa ja tiputa ja akun kunnon seuraaminen. 2

Monet android käyttöjärjestelmää käyttävät yhtiöt lisäävät androidiin omia ominaisuuksiaan ja tekevät käyttöjärjestelmästä oman näköisensä ja tarpeisiinsa sopivan. Puhtaalla androidilla tarkoitetaan androidin versiota, joka on suoraan googlelta, eikä siinä ole muiden lisäämiä ominaisuuksia. Tämä mahdollistaa esimerkiksi päivitysten saamisen ensimmäisten joukoissa. Esimerkiksi Googlen omat puhelimet ovat tällaisia.

Oman sovelluksen saamiseksi App Storeen tulee kirjautua Applen developer ohjelmaan ja lisätä valmis sovellus Xcoden kautta tarkistettavaksi Applen toimesta. 3

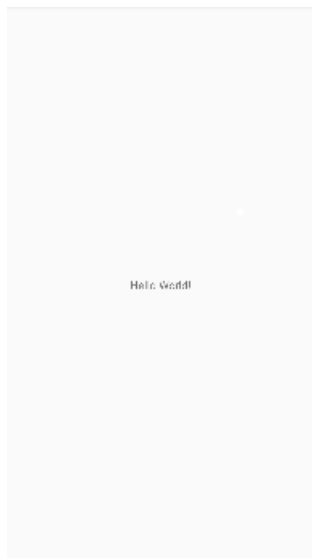
Swift on avoimen lähdekoodin kieli, joka on suunniteltu IOS sovellusten luomiseen, joten sillä pystyy pääsemään käsiksi pääasiassa kaikkiin IOS:in ominaisuuksiin.

## Harjoitus 3

GitLabin käyttö ja Android studion lataaminen sujui vaivatta. Hello World sovelluksen luominen onnistui helposti seuraamalla android developer dokumentaation java esimerkkiä. Minulla ei ole käytössä android laitetta, joten jouduin ottamaan käyttöön android emulatorin. Uuden Projektin luodessa Android studio luo automaattisesti erilaiset tarvittavat pakkaukset, joita voi myös muokkailla oman tarpeen mukaan.

Käytännössä yksinkertainen Android sovellus koostuu pakkauksien lisäksi Activity tiedostosta, jossa määritellään sovelluksen operaatiot ja tätä vastaavasta xml tiedostosta, johon kirjoitetaan sovelluksen layout. Tätä voidaan sitten laajentaa luomalla lisää layoutteja ja Activityja ja luomalla erilaisia luokkia ja rajapintoja ohjelmaan. Nämä sijoitetaan ohjelman main luokan alle, kun taas erilaiset testit sijoitetaan esimerkiksi testiluokkien alle. Erilaisia kirjastoja saa otettua käyttöön lisäämällä niitä sovelluksen riippuvuuksiin ja ne saa luokkiin import lauseilla.

Emulator vaati toimiakseen virtualization teknologian käyttöönottoa, joka tuotti aluksi hieman ongelmia. Lopulta emulator alkoi toimimaan laittamalla virtualization päälle ja Uma buffer size automaattiseksi. Tietokone ei suostunut käynnistymään ilman uma buffer size asetusta, jos virtualization oli päällä, joten sen muuttaminen oli käänteentekevää ja kaikki alkoi toimimaan.

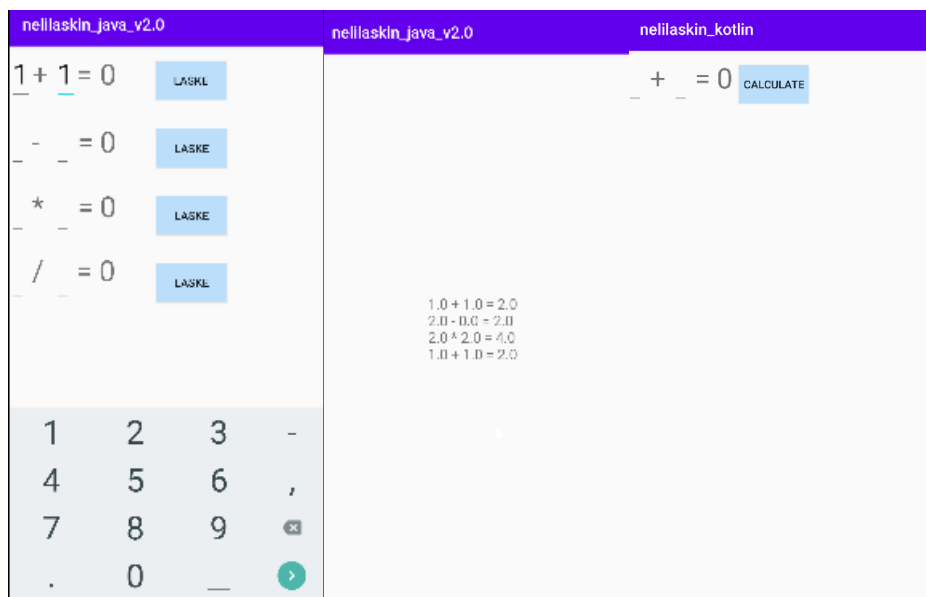


## Harjoitus 4 ja 5

Tein harjoituksen 4 Kotlinilla ja harjoituksen 5 Javalla, sillä se on itselleni tutumpi ja harjoitus 5 oli sen verran laajempi kuin harjoitus 4.

Loin aluksi Yksinkertaisen plus lasku näkymän, johon sitten lisäsin constraintit chatgpt:llä. Sitten loin Kotlinilla eventlistenerin laskupainiketta varten, joka laskee plus laskun kahden annetun numeron välillä ja näyttää sen = merkin jälkeen. Chatgpt tarjosi aluksi Relative layoutia xml tiedostoon, joka vaikutti kätevämmältä kuin constraint layout. Relative layoutia todennäköisesti käytänkin jatkossa, mutta tämä harjoitus on tehty constraintilla.

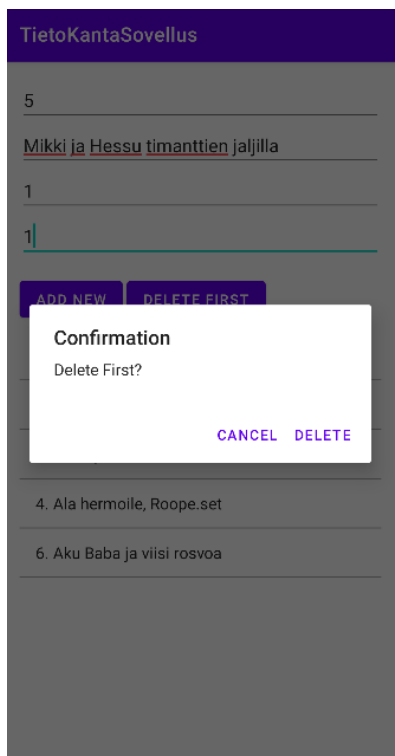
Sitten tein vastaavat näkymät muille laskuille harjoitusta 5 varten ja loin chatgpt:llä tiedostoon kirjoittamis ja lukutoiminnot siirryttäessä uuten activityyn. Java oli jo entuudestaan tuttua ja moni operaatio oli hyvin samanlainen MainActivity tiedostossa kuin tietokoneellekin kirjoitetuissa ohjelmissa. Suurin eroavaisuus työpöytä ohjelmointiin onkin layout tiedostojen käyttäminen ja ohjelman ajaminen.



## Harjoitus 6 – 8

Tietokannan käyttö androidissa vaati hieman opettelua ja sisäistämistä, mutta itse tietokannan luominen onnistui lopulta helposti chatgpt:llä, jolla sai luotua helposti tarvittavat luokat malliksi tietokannan käyttöön. Käytännössä Room:illa tehty Sqlite sovellus tarvitsee Luokan joka kuvaa tietokantaan lisättävää taulua, siihen liittyvän database ja operations luokan ja viewmodel ja repository luokat, jotka yhdistävät databasen itse sovellukseen.

Kun tietokanta ja siihen liittyvät luokat oli luotu oli helppo jatkaa tehtävää tekemällä activity\_main.xml ja MainActivity tiedostot ja tehdä käyttöliittymästä vaadittava. Tein esimerkin mukaisen taskari sovelluksen, joka lajittelee tietokannan taskarit taskarin numeron mukaan näyttää ne listana sovelluksessa. Tietokantaan voi lisätä ja poistaa taskareita ja poistaminen vaatii hyväksynnän ponnahdusikkunassa. Lajittelu oli helppoa muuttamalla luokan compare metodia ja Sekä taskarin numeron ja nimen sai helposti näkymään listviewissä muuttamalla luokan toString metodia ja käyttämällä androidin simple\_list\_layout\_1 layoutia.



## Harjoitus 9 ja 10

Firebase palveluiden käyttöönotto onnistui helposti seuraamalla Googlen dokumentaatiota. Käytännössä sovellus tuli yhdistää firebaseen lisäämällä riippuvuudet ja erillinen googleservices json tiedosto. Tämän jälkeen firestoren ja authenticatin sai käyttöön lisäämällä niihin riippuvuudet.

Tein saman teemaisen sovelluksen kuin edellisessä taskaritehtivässä. Käytännössä copypastesin Luokan taskarille ja activity\_main.xml tiedoston ja muokkasin MainActivity tiedostoa käyttämään firebasea Room:in sijasta. Roomia ei siis firebase toteutuksessa tarvittu. Käyttöliittymä on samanlainen kuin edellisessä harjoituksessa, mutta siihen on alkuun lisätty kirjautumisikkuna. Mitään database operaatio luokkia ei tarvittu toteutuksessa niinkuin edellisessä.

Käytän email password todentamista firebasesta niin, että sovellukseen pääsee sisään antamalla tiedot **guest@gmail.com** ja **guestpassword** ja kun sovellukseen on kirjautunut on valtuutettu poistamaan ja lisäämään artikkeleita tietokantaan. Tämän määrittely onnistui firebaseassa rules kohdassa. Kirjautumisikkunaa varten piti luoda oma activity ja layout tiedosto ja lisätä tästä tieto AndroidManifest tiedostoon. Sovellus näyttää ensin kirjautumisikkunan ja sitten onnistuneen kirjautumisen jälkeen pääsovelluksen.

FirestoreSovellus	FirestoreSovellus
guest@gmail.com	Numero
*****	Nimi
	Painos
	Hankinta
LOGIN	ADD NEW DELETE FIRST
	1. Mikki Kiipelissa
	3. Mikki ja viidakon vaarat

```
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
↑ z x c v b n m ✕
?123 , . ✓

rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Allow read access to all documents
    match /{document=**} {
      allow read: if true;
    }

    // Allow write and delete access to the entries collection for authenticated users
    match /entries/{entry} {
      allow read: if true;
      allow write, delete: if request.auth != null;
    }
  }
}
```

## Harjoitus 11

ChatGpt osasi suoraan kertoa mitkä luvat tulee lisätä Android manifest tiedostoon tulee lisätä, jotta pääsee käsiksi laitteen sensoreihin, joita tehtävässä tarvitsee. Sen jälkeen käyttämällä SensorEventListener rajapintaa se osasi hakea tarvittavat arvot.

Kun arvot saatiin tallennettua oli helppo luoda yksinkertaiset textview elementit layout tiedostoon. Käytännössä Android Emulatorilla toteutus jäi vähän tylsäksi, sillä se ei täysin simuloi android laitetta, jossa kyseiset sensorit ovat sidoksissa itse laitteeseen.

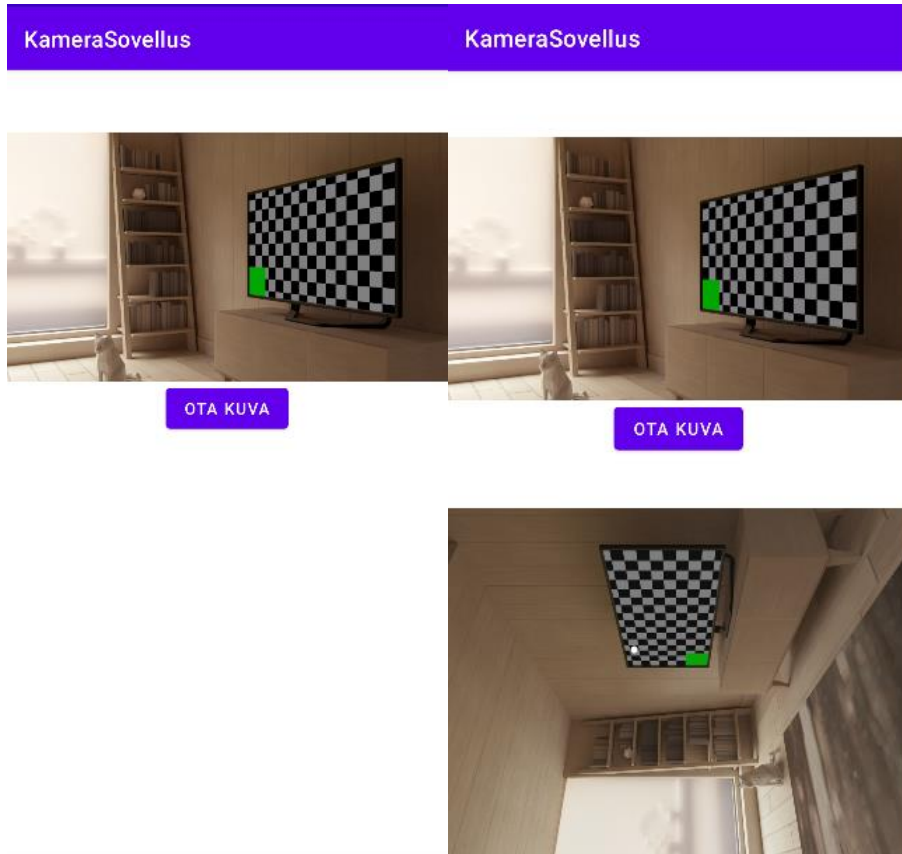
Valo sensorin käytännössä esittää laitteen ympärillä olevan valon määrän, mikä on hyödyllistä esimerkiksi automaattisessa kirkkauden säätelyssä. Läheisyys sensorin tunnistaa onko laitteen läheisyydessä jokin artikkeli ja tätä voidaan käyttää esimerkiksi automaattisessa näytön sulkemisessa. Orientaatio anturi kertoo puhelimen kierron arvot, joka on hyödyllistä esimerkiksi erilaisissa peleissä, joissa pelaaminen perustuu puhelimen kääntelyyn.

Android emulatorissa käyttämäni Google Pixel 5 ei tue ilmanpaineen, ilman kosteuden tai lämmön mittaamista, eikä myöskään oma iphone 12 minini.

SensoriData
Light Sensor: Proximity Sensor: Orientation: -0.0, 0.0, -0.0

## Harjoitus 12

Käytössäni on emulator Pixel 5 Api 30. Android2 tai Android1 kirjastoilla toteutus ei onnistunut, mutta AndroidX:llä sen sai toimimaan. Jostain syystä kuva kiertyi 90-astetta, enkä saanut sitä korjattua chatgpt:lläkään.





## Harjoitus 13 ja 14

Mitkään tarvittavista sensoreista ei ole saatavilla emulatorissa, mutta kiihtyvyys anturiin on luotu interaktiivinen toteutus, jota voi käyttää askelmittarin toteutuksessa. Kysyin chatgpt:ltä onko sensorit saatavilla ja testastin yksinkertaisesti pystyykö emulator löytämään niitä.

Käytännössä sovellusta voi testata asettamalla sensorin herkkyyden pieneksi, jolloin luku alkaa rullaamaan, sillä itse liikkuminen ei ollut emulatorilla mahdollista.

Askelimittari sovelluksen tein siis kiihtyvyysensorilla ja apuna käytin lähteenä vain chatgpt:tä.

OminaisuuksienTarkastelu

Askelmittari

ALOITA

LOPETA

TYPE\_STEP\_COUNTER sensor ei saatavilla  
TYPE\_STEP\_DETECTOR ei saatavilla  
ACTIVITY\_RECOGNITION ei saatavilla

Askel Määrä: 0

TALLENNNA

---

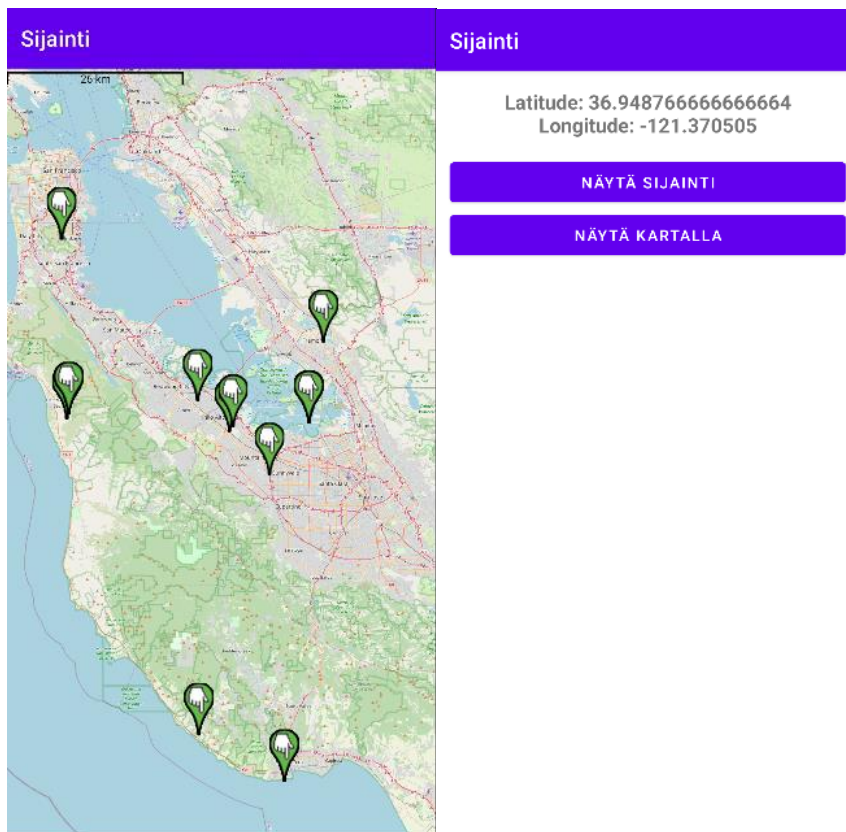
## Harjoitus 15 ja 16

Harjoitus on testattu emulatorilla, ja vakio sijaintina oli kalifornia, joten testailin sovellusta asettelemalla erilaisia sijainteja San fransiscon ja San Josen alueilla emulatorin extended controls Location valikossa.

Tarvittavat luvat karttaan ja sijaintiin löytyivät helposti ja siitä oli helppo aloittaa sovelluksen toteutus. Sovellus käytännössä Näytä Sijainti painiketta painamalla aloittaa toimintansa pyytämällä sijaintia. Sen jälkeen LocationListener rajapinnan onLocationChanged metodia käyttämällä päivittää ja tallentaa sijainnin aina sijainnin muuttuessa.

Sitten näytä kartalla nappia painamalla pääsee katselemaan tallennettuja pisteitä openstreetmap kartalla.

Sovelluksessa on käytetty apuna vain GitHubin kartta materiaalia ja chatgpt:tä.



## Harjoitus 17-19

Valuuttalaskin on toteutettu tekemällä GET request Volley-kirjastolla tehtävässä annettuun xml sivuun ja sen jälkeen muuttamalla responsesta saadut valuutta kurssit mapiksi, josta sitten haetaan tarvittavat kurssit nappia painamalla ja muutetaan annettu summa dollareiksi, punniksi ja ruotsin kruunuiksi.

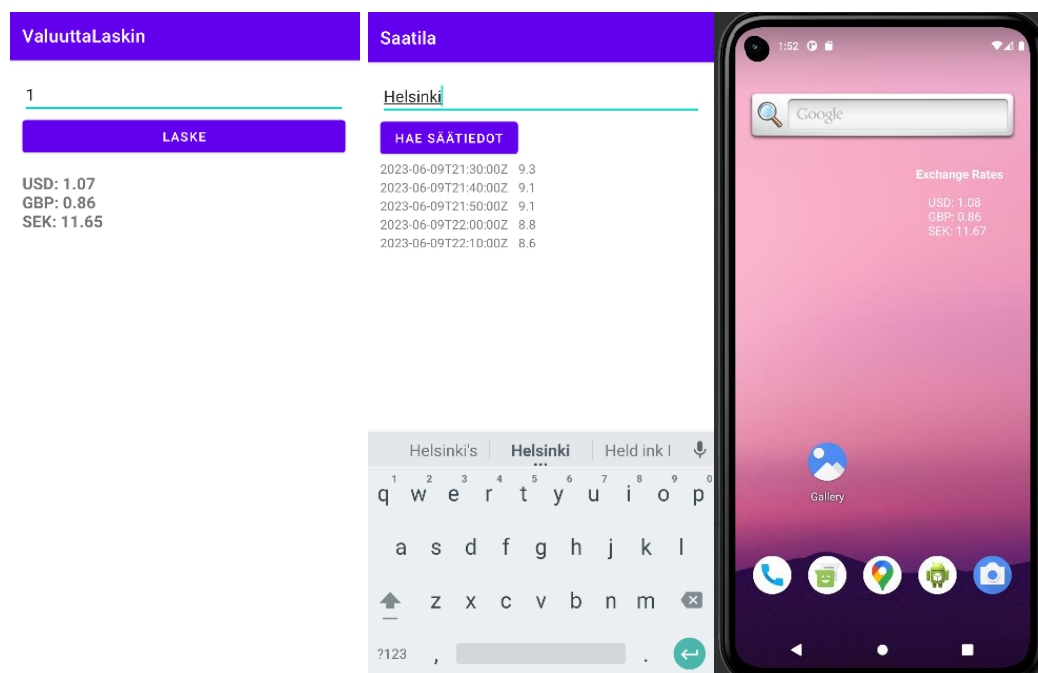
Säätilä sovellus on toteutettu käytännössä samalla tekniikalla kuin valuuttalaskin, mutta se hakee ainoastaan tarvittavat viisi lämpötilaa valitulle kaupungille eikä kaikkia dataa niinkuin valuuttalaskin, jossa response oli niin pieni, että tämä oli vielä perusteltavaa. Request ei mennä aluksi toimia, mutta chatGpt osasi kertoa, että sovellus tarvitsee erilliset turvallisuus määrittelyt jos se tekee pyyntöjä muuhun kuin https osoitteeseen.

Widget on luotu hyödyntämällä valuuttalaskimen toteutusta ja luomalla widget uuteen projektiin. Toteutuksissa on hyödynnetty chatgpt:tä.

```
xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config cleartextTrafficPermitted="true">
    <domain includeSubdomains="true">opendata.fmi.fi</domain>
  </domain-config>
</network-security-config>
```

Then, in your app's manifest file ('AndroidManifest.xml'), add the 'android:networkSecurityConfig' attribute to the 'application' element:

```
xml
<application
  ...
  android:networkSecurityConfig="@xml/network_security_config"
  ...
</application>
```



## Harjoitus 20

Mielestäni kurssi toimi kokonaisuutena hyvin. Työmäärä oli sopiva, harjoitustehävät eivät olleet liian vaikeita eikä liian helppoja ja niihin ei jäänyt jumittamaan. Kurssilla tarjottu materiaali oli riittävää ja harjoitustehtävät + opimispäiväkirja toimi hyvin.

Kurssi pysyi hyvin asiassa ja opinkin perussovellusten toteuttamisen ja työvälineiden käytön androidiin mitä haulsinkin. Jonkin näköinen katsaus los ohjelmointiin olisi voinut olla mielenkiintoinen, nyt se jää käytännössä oman harastuneisuuden varaan.

Teoriaa enemmän kiinnosti enemmän itse sovellusten toteuttaminen ja tähän kuluikin enemmän aikaa kuin luentoihin ja materiaaliin tutustumiseen. Aineiston sijasta lähinnä Googlettelin apuja ja käytin ChatGpt:tä lähes jokaisessa harjoituksessa.

Harjoitukset 13 ja 14 tuntuivat ehkä turhimmilta, mihin voi vaikuttaa se että käytössäni oli vain android emulator ja muuteskin ne käytännössä opettivat vain miten pyytää lupa android manifest tiedostossa ja sitten käyttää androidin metodeja sensoritiedon saamiseen, joka onnistuu nykyään lähes yhdellä ChatGpt kysymyksellä. Mielenkiintoisimpia harjoituksista olivat varmasti harjoitukset 9 ja 10 sillä pilviplaneluiden käyttö sovelluksessa on hyvin relevanttia tänä päivänä.

## Harjoitustyö

### Toiminta, ominaisuudet ja rakenne

Projektin aiheena on muistutus sovellus johon käyttäjä voi kirjautua ja asettaa muistutuksia erilaisilla tiedoilla.

Sovelluksen käynnistyessä tulee esiin. Alkunäyttö, josta sovellukseen pääsee kirjautumaan tai rekisteröitymään. Riippuen valitusta tavasta, tulee esiin login tai register näkymä, joista pääsee sisään itse pääikkunaan. Näkymien välillä voi navigoida painikkeiden alla olevien tekstien kautta. Kirjautuminen toteutetaan Firebase authenticaten avulla.

Sovelluksen siirtyessä pääikkunaan se hakee Firestore tietokannasta käyttäjän tallennetut muistutukset. Muistutukset näytetään listviewinä keskellä näyttöä, niin että siitä näkyy vain otsikko ja muistutusta painamalla näkee sen kaikitiedot.

+ painiketta painamalla tulee esiin kentät muistutuksen otsikolle, kuvaukselle, ajalle, ja sijainnille. Ne voi jättää joko tyhjäksi tai täyttää kaikki, muistutus luodaan tietokantaan kaikesta huolimatta. Add Note painikkeen painamisen jälkeen listview päivitetään uudella muistutuksella. Muistutuksen voi merkata aktiiviseksi ja poistaa sen otsikon viereisistä painikkeista. Kaikki muutokset siirtyvät tietokantaan ja listview päivittyy sen mukaan.

Ajan pystyy asettamaan joko näppäimistöllä tai painamalla tekstikenttää tulee Androidin Time picker esiin josta käyttäjä voi myös valita halutun ja sitten asettaa ajan tekstikenttään Add Time painikkeesta. Myös sijainnin käyttäjä pystyy asettamaan joko manuaalisesti tai valitsemalla Open Street Map API:n kartasta, joka tulee esiin painamalla tekstikenttää. Painamalla karttaa tallentuu piste, joka sitten välitetään Google geocoder API:lle, joka muuttaa pisteen fyysiseksi paikan nimeksi.

Kuvista puuttuu logout painike oikeasta yläkulmasta, jonka lisäksi ohjelmaan ai-  
van lopuksi, sillä sen olin unohtanut. Logout painike poistaa autentikoinnin ja palauttaa käyttäjän alkuäkymään.



- ## Kuvat

<div>Reminders</div> <div>REGISTER</div> <div>LOGIN</div>	<div>Reminders</div> <div>guest@user.com</div> <div>*****</div> <div>LOGIN</div> <div>New Here? Register Now!</div>
---	---

## Reminders

Valid Email

Password

Confirm Password

REGISTER

Already A User? Log In!

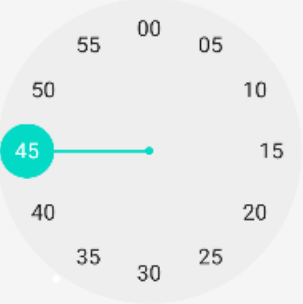

## Reminders

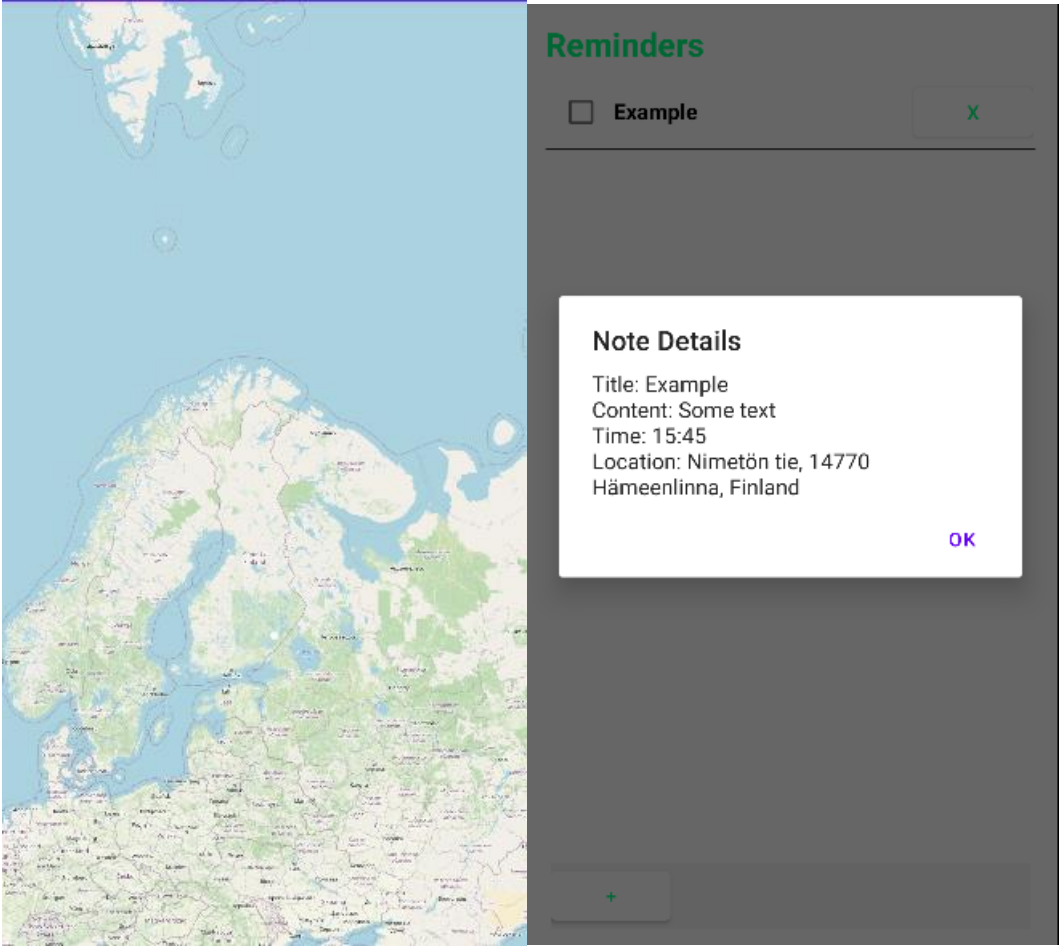


## Reminders

Note Title
Note Content
Note Time
Note Location
ADD NOTE

## Reminders

ADD TIME	X
3:45 AM PM	
	
	



## Ajankäyttö

Luennot 0, 1 ja 2	1h
Harjoitus 1	2h
Harjoitus 3	3h
Luento 3 ja 4	1h
Harjoitus 4 ja 5	4h
Harjoitus 6, 7 ja 8	6h
Luennot 4 ja 5	1h
Harjoitus 9 ja 10	6h
Luennot 6, 6.5 ja 7	1h
Harjoitus 11	1h
Harjoitus 12	3h
Harjoitus 13 ja 14	4h
Luento 8	1h
Harjoitus 15 ja 16	5h
Harjoitus 17, 18 ja 19	11h
Harjoitustyö	50h
Harjoitus 20	30min

## **Lähdeluettelo**

1. <https://devmountain.com/blog/how-to-create-an-ios-app/#:~:text=Before%20creating%20an%20iOS%20app,to%20respond%20to%20user%20interaction.>
2. <https://www.makeuseof.com/best-ios-features-missing-from-android/>
3. <https://codewithchris.com/submit-your-app-to-the-app-store/>
4. Chatgpt - <https://chat.openai.com/>