



Sovellusten ohjelmointi ja käytettävyys Oppimispäiväkirja

Antti Venetjoki

SISÄLLYS

[1 Viikkoharjoitukset 1 4](#_Toc181651392)

[1.1 Android -ympäristön asennus ja Hello World 4](#_Toc181651393)

[1.1.1 Android -ympäristön asennus 4](#_Toc181651394)

[1.1.2 Github-linkki 4](#_Toc181651395)

[1.1.3 Todiste ohjelman ajosta 4](#_Toc181651396)

[1.2 Jetpack Compose -tutustuminen 5](#_Toc181651397)

[1.2.1 Github-linkki 5](#_Toc181651398)

[1.2.2 Koodi ajettuna Android virtuaalikoneessa 5](#_Toc181651399)

[1.3 Kotlin essentials – osa 1 5](#_Toc181651400)

[2 Viikkoharjoitukset 2 6](#_Toc181651401)

[2.1 Valuuttamuuntimen käyttöliittymä 6](#_Toc181651402)

[2.1.1 Github-linkki 6](#_Toc181651403)

[2.1.2 Kuva käyttöliittymästä 6](#_Toc181651404)

[2.2 Sääsovelluksen käyttöliittymä 6](#_Toc181651405)

[2.2.1 Github-linkki 6](#_Toc181651406)

[2.2.2 Kuva käyttöliittymästä 7](#_Toc181651407)

[2.3 Scaffold 7](#_Toc181651408)

[2.3.1 Github-linkki 7](#_Toc181651409)

[2.3.2 Kuva käyttöliittymästä 7](#_Toc181651410)

[2.4 Kotlin harjoituksia osa 2 7](#_Toc181651411)

[3 Viikkoharjoitukset 3 8](#_Toc181651412)

[3.1 Lokalisointi 8](#_Toc181651413)

[3.1.1 Pohdinta 8](#_Toc181651414)

[3.1.2 github-linkki 8](#_Toc181651415)

[3.1.3 Kuvia ohjelman ajosta 8](#_Toc181651416)

[3.2 Teemat 9](#_Toc181651417)

[3.2.1 Pohdinta 9](#_Toc181651418)

[3.2.2 Github-linkki 9](#_Toc181651419)

[3.2.3 Kuvia ohjelman ajosta 9](#_Toc181651420)

[3.3 Sovelluksen tila ja toiminnallisuus 10](#_Toc181651421)

[3.3.1 Pohdinta 10](#_Toc181651422)

[3.3.2 Github-linkki 10](#_Toc181651423)

[3.3.3 Kuvia ohjelman ajosta 10](#_Toc181651424)

[4 Viikkoharjoitukset 4 11](#_Toc181651425)

[4.1 Navigointi 11](#_Toc181651426)

[4.1.1 Github-linkki 11](#_Toc181651427)

[4.1.2 Kuvia toiminnasta 11](#_Toc181651428)

[4.2 Bottom Tabs 12](#_Toc181651429)

[4.2.1 Github-linkki 12](#_Toc181651430)

[4.2.2 Kuvia toiminnasta 12](#_Toc181651431)

[4.3 Intent 12](#_Toc181651432)

[4.3.1 Pohdinta 12](#_Toc181651433)

[4.3.2 Github-linkki 12](#_Toc181651434)

[4.3.3 Kuva ohjelmasta 13](#_Toc181651435)

[5 Viikkoharjoitukset 5 14](#_Toc181651436)

[5.1 Dataluokat ja listojen toteuttaminen 14](#_Toc181651437)

[5.1.1 Pohdinta 14](#_Toc181651438)

[5.1.2 Kuva ohjelmasta 14](#_Toc181651439)

[5.1.3 github-linkki 15](#_Toc181651440)

[5.2 Detaljinäkymä 15](#_Toc181651441)

[5.2.1 Kuva näkymästä 15](#_Toc181651442)

[5.2.2 github-linkki 15](#_Toc181651443)

[6 Viikkoharjoitukset 6 16](#_Toc181651444)

[6.1 REST-toiminnallisuuden toteuttaminen Android-sovelluksissa 16](#_Toc181651445)

[6.1.1 Suorat HTTP-pyynnöt HttpURLConnection- ja OkHttpClient-luokilla 16](#_Toc181651446)

[6.1.2 Volley-kirjasto REST-pyyntöjen toteutukseen 17](#_Toc181651447)

[6.1.3 Retrofit ja sen suosion syyt Android-kehityksessä 18](#_Toc181651448)

[6.2 JSON-tiedon konvertointi Kotlin data-luokiksi 18](#_Toc181651449)

[6.3 Tehtävälista-sovellus ja tietojen haku palvelimelta 19](#_Toc181651450)

[6.3.1 Github-linkki 19](#_Toc181651451)

[6.3.2 Kuva ohjelmasta 19](#_Toc181651452)

[6.4 REST-pohjainen sääsovellus 19](#_Toc181651453)

[6.4.1 Pohdinta 19](#_Toc181651454)

[6.4.2 Github-linkki 19](#_Toc181651455)

[6.4.3 Kuva ohjelmasta 20](#_Toc181651456)

[Käytetyt lähteet 21](#_Toc181651457)

# Viikkoharjoitukset 1

## Android -ympäristön asennus ja Hello World

### Android -ympäristön asennus

Asensin version Android Studiosta, joka sisälsi myös Android SDK

ja tarvittavat työkalut. Asennusohjelma ohjasi automaattisesti SDK

asennukseen. Huomasin, että prosessi asensi myös virtuaaliset Android-laitteet (AVD), joita käytetään sovellusten testaamiseen ilman fyysistä laitetta. Valitsin AVD Managerista laitteeksi Pixel 8

ja Android 12 -version, jossa on API-versio 31. Virtuaalikoneen käynnistämisessä huomasin, että laite voi olla hidas, mikä johtui osittain koneen resurssien rajoitteista.

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/blob/main/HelloName/app/src/main/java/com/example/helloname/MainActivity.kt)

### Todiste ohjelman ajosta

A screen shot of a cell phone

Description automatically generated

## Jetpack Compose -tutustuminen

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/blob/main/JetpackComposeStarter/app/src/main/java/MainActivity.kt)

### Koodi ajettuna Android virtuaalikoneessa

A screen shot of a cell phone

Description automatically generated

## Kotlin essentials – osa 1

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/blob/main/KotlinEssentials.kt)

# Viikkoharjoitukset 2

## Valuuttamuuntimen käyttöliittymä

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/valuuttamuunnin/app/src/main/java/com/example/valuuttamuunnin)

### Kuva käyttöliittymästä

A screen shot of a cell phone

Description automatically generated

## Sääsovelluksen käyttöliittymä

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/WeatherApp/app/src/main/java/com/example/weatherapp)

### Kuva käyttöliittymästä

A screen shot of a cell phone

Description automatically generated

## Scaffold

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/MyScaffoldExample/app/src/main/java/com/example/myscaffoldexample)

### Kuva käyttöliittymästä

A screen shot of a cell phone

Description automatically generated

## Kotlin harjoituksia osa 2

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/blob/main/KotlinEssentials.kt)

# Viikkoharjoitukset 3

## Lokalisointi

### Pohdinta

Lokalisointi resurssitiedostoissa koodin sijaan on hyvä käytäntö monista syistä. Koodissa ei ole kovakoodattuja merkkijonoja, mikä tekee koodista siistimpää ja helpommin luettavaa. Resursseja voidaan muokata ilman, että kosketaan itse logiikkaan, mikä vähentää regressiovirheiden mahdollisuutta. Sovellus voi näyttää automaattisesti oikean käännöksen käyttäjän laitteen kieliasetusten mukaan, mikä parantaa käyttäjäkokemusta.

### github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/WeatherAppLocal)

### Kuvia ohjelman ajosta

A screen shot of a cell phone

Description automatically generatedA screen shot of a phone

Description automatically generated

## Teemat

### Pohdinta

Muutokset ulkoasuun voidaan tehdä helposti ja keskitetysti. Jos haluat muuttaa esimerkiksi kaikkien painikkeiden värin, voit tehdä sen teemasta käsin ilman, että sinun täytyy käydä läpi kaikkia käyttöliittymäkomponentteja erikseen. Teemat varmistavat, että koko sovelluksen ulkoasu on yhtenäinen. Väripaletti, fontit ja muut visuaaliset elementit pysyvät johdonmukaisina, mikä parantaa käyttäjäkokemusta. Teemat mahdollistavat tumman ja vaalean tilan tai jopa dynaamisten värien käytön Androidin materiaalidesignin mukaisesti. Sovellus voi mukautua käyttäjän järjestelmäasetuksiin tai antaa käyttäjälle mahdollisuuden valita eri teemoja itse.

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/WeatherAppTheme)

### Kuvia ohjelman ajosta

A screen shot of a cell phone

Description automatically generatedA screen shot of a phone

Description automatically generated

## Sovelluksen tila ja toiminnallisuus

### Pohdinta

Käyttöliittymän tilalla tarkoitetaan sovelluksen käyttöliittymän eri tiloja, joita voidaan käyttää käyttäjän interaktioiden seuraamiseen ja hallintaan. Näitä tiloja voi olla esimerkiksi käyttäjän syöttämät tiedot, sovelluksen näkymät tai eri elementtien tilat, kuten lomakekenttien sisällöt. Käyttöliittymän tilan hallinta on keskeistä, jotta sovelluksen eri osat voivat reagoida oikein käyttäjän tekemisiin.

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/FuelCalculator)

### Kuvia ohjelman ajosta

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

# Viikkoharjoitukset 4

## Navigointi

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/Navigation)

### Kuvia toiminnasta

A screen shot of a phone

Description automatically generatedA screen shot of a cell phone

Description automatically generatedA screen shot of a phone

Description automatically generated

## Bottom Tabs

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/BottomTabs)

### Kuvia toiminnasta

A black rectangular device with a black screen

Description automatically generatedA black rectangular device with a black screen

Description automatically generatedA black rectangular device with a black screen

Description automatically generated

## Intent

### Pohdinta

Androidissa Common Intents tarkoittaa järjestelmän tarjoamia valmiita intenttejä, joita voidaan käyttää yleisiin tehtäviin sovellusten välillä. Intentti on olio, jota käytetään viestimään eri komponenttien (aktiviteettien, palveluiden jne.) välillä. Common Intents tarjoavat valmiita toimintoja, joilla sovellukset voivat käynnistää Androidin sisäänrakennettuja toimintoja tai siirtyä toisen sovelluksen tiettyyn toimintaan.

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/CommonIntent)

### Kuva ohjelmasta

A screenshot of a phone

Description automatically generated

# Viikkoharjoitukset 5

## Dataluokat ja listojen toteuttaminen

### Pohdinta

Kotlinin dataluokat (data classes) on suunniteltu tietorakenteiksi, ja ne generoivat automaattisesti tärkeät metodit kuten equals(), hashCode(), toString(), ja copy(). Javan dataluokissa nämä metodit täytyy kirjoittaa itse. Kotlinin dataluokat tukevat helppoa datan kopiointia ja toimivat sujuvasti data-binding-kirjastojen kanssa, kun taas Javassa vastaava käytettävyys vaatii lisäkoodia.

Column on yksinkertainen pystysuuntainen kontti, joka soveltuu pienille, kiinteille määrille elementtejä, sillä se renderöi kaikki elementit kerralla. LazyColumn on tarkoitettu suurille listanäkymille, ja se luo vain näkyvissä olevat elementit (lazy-loading), optimoiden suorituskyvyn ja muistinkäytön. Dynaamisissa listoissa LazyColumn on tehokkaampi vaihtoehto.

### Kuva ohjelmasta

A screen shot of a phone

Description automatically generated

### github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/Restaurant)

## Detaljinäkymä

### Kuva näkymästä

A screen shot of a phone

Description automatically generated

### github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/Restaurant)

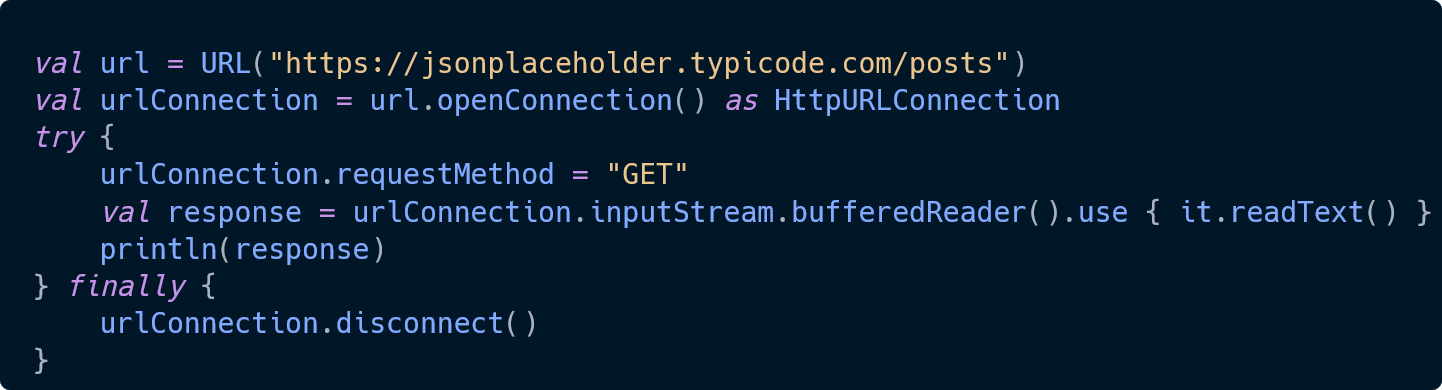
# Viikkoharjoitukset 6

## REST-toiminnallisuuden toteuttaminen Android-sovelluksissa

REST-toiminnallisuutta voidaan toteuttaa Android-sovelluksissa hyödyntämällä erilaisia lähestymistapoja ja kirjastoja verkkopyyntöjen tekemiseen. Kotlin-kielellä REST-pyyntöjen tekemiseen on useita vaihtoehtoja, kuten **HttpURLConnection**, **OkHttpClient**, **Volley** ja **Retrofit**. Jokaisella vaihtoehdolla on omat etunsa ja käyttötarkoituksensa.

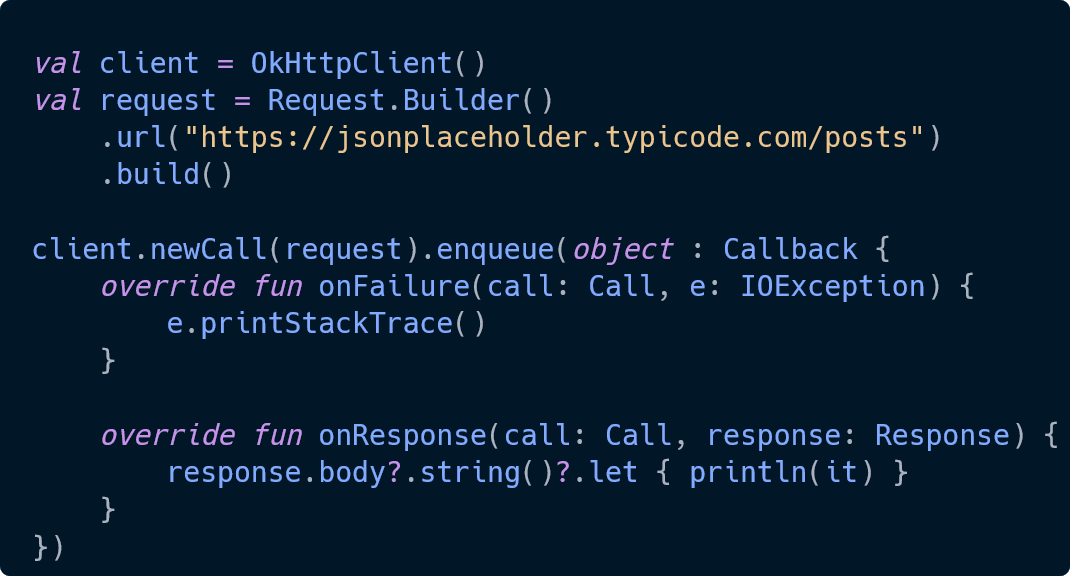
### Suorat HTTP-pyynnöt HttpURLConnection- ja OkHttpClient-luokilla

HttpURLConnection: Tämä on yksi Androidin perustyökaluista verkkopyyntöjen suorittamiseen. HttpURLConnection on yksinkertainen ja kevyt tapa toteuttaa perus-HTTP-pyynnöt, mutta sillä on rajoituksia, kuten monimutkaisuus virheenkäsittelyssä ja JSON-datan hallinnassa.



Kuva 1 Esimerkki HttpURLConnection-pyynnöstä

**OkHttpClient:** Tämä on Googlen suosittelema vaihtoehto, joka tarjoaa useita edistyneitä ominaisuuksia, kuten asynkroniset pyynnöt ja tehokkaamman välimuistin hallinnan. OkHttpClient on luotettavampi ja joustavampi kuin HttpURLConnection, ja se on laajalti käytetty Android-kehityksessä.



Kuva 2 Esimerkki OkHttpClient-pyynnöstä

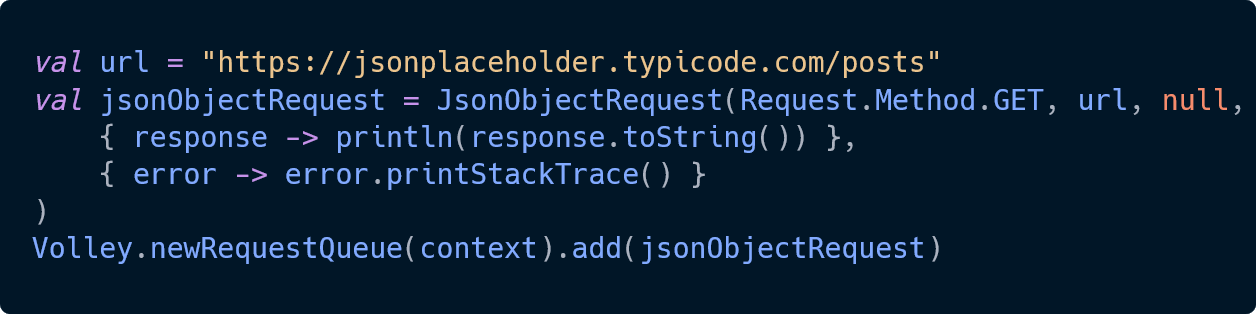
**Sovelluksen suorituskyky pääsäikeessä:** Jos HTTP-pyyntöjä tehdään suoraan pääsäikeessä ilman erillisiä säikeitä tai asynkronista käsittelyä, se voi hidastaa käyttöliittymän toimintaa merkittävästi ja jopa johtaa sovelluksen "ei vastaa" -tilaan (ANR, Application Not Responding). Sekä HttpURLConnection että OkHttpClient mahdollistavat pyynnöt erillisessä säikeessä, mutta OkHttpClient tukee myös asynkronisia pyyntöjä, jotka ovat tehokkaampia ja vähemmän kuormittavia käyttöliittymän kannalta.

### Volley-kirjasto REST-pyyntöjen toteutukseen

**Milloin käyttää Volleya:** Volley on Googlen kehittämä kirjasto, joka on hyvä vaihtoehto verkkopyyntöjen toteutukseen, erityisesti kun tarvitaan yksinkertainen ja kevyt ratkaisu ilman monimutkaisia konfiguraatioita. Se sopii pieniin ja keskisuuriin sovelluksiin, joissa on paljon verkkopyyntöjä ja mahdollisesti dynaamista tietoa.

**Keskeiset ominaisuudet:**

* **Välimuisti:** Volleyllä on sisäänrakennettu välimuisti, joka mahdollistaa aiempien pyyntöjen tallentamisen. Tämä parantaa suorituskykyä ja vähentää verkkokuormitusta.
* **JSON-tuki:** Volley tarjoaa sisäänrakennetun tuen JSON-objekteille, kuten JsonObjectRequest ja JsonArrayRequest, mikä helpottaa JSON-datan hakemista ja käsittelyä.
* **Asynkroninen pyyntöjono:** Volley hallinnoi automaattisesti pyyntöjonoa ja suorittaa verkkopyynnöt taustasäikeessä. Tämä varmistaa, ettei käyttöliittymän pääsäie kuormitu.



Kuva 3 Esimerkki Volleyn käytöstä JSON-pyynnön tekemisessä

### Retrofit ja sen suosion syyt Android-kehityksessä

**Miksi Retrofit on suosituin REST-kirjasto:** Retrofit on yksi suosituimmista ja tehokkaimmista kirjastoista REST-toiminnallisuuden toteutukseen Android-sovelluksissa, erityisesti Kotlinin ja Jetpack Composen kanssa. Retrofitiä pidetään erittäin luotettavana ja kattavana, sillä se integroituu helposti Gson- tai Moshi-kirjastojen kanssa, jotka auttavat JSON-datan automaattisessa deserialisoinnissa suoraan Kotlin-luokkiin.

**Retrofitiin liittyviä etuja:**

* **Helppo JSON-käsittely:** Retrofit tukee automaattista konversiota JSON-datasta Kotlin-objekteihin, mikä vähentää koodin määrää ja virheiden mahdollisuutta.
* **Tuki erillisille HTTP-metodeille:** Retrofit tukee suoraan GET, POST, PUT, DELETE ja muita HTTP-metodeja annotaatiolla.
* **Asynkroninen käsittely ja coroutine-tuki:** Retrofit integroituu hyvin Kotlinin coroutinen kanssa, mahdollistaen tehokkaan asynkronisen käsittelyn ja pääsäikeen keventämisen.



Kuva 4 Esimerkki Retrofitin käytöstä

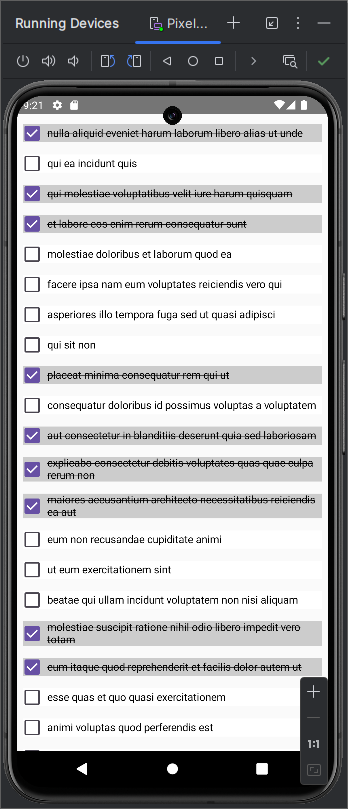
## JSON-tiedon konvertointi Kotlin data-luokiksi

## Tehtävälista-sovellus ja tietojen haku palvelimelta

### Github-linkki

[github.com](https://github.com/KippaK/5G00EV05-3004-Sovellusten-ohjelmointi-ja-kaytettavyys/tree/main/TodoList)

### Kuva ohjelmasta



## REST-pohjainen sääsovellus

### Pohdinta

Sovellus on todettu toimivaksi ja sen jälkeen api-avain on poistettu tiedostoista. Tämän voisi korjata esimerkiksi .env tiedostolla, mutta sitä en tähän nyt alkanut toteuttamaan

### Github-linkki

### Kuva ohjelmasta



Käytetyt lähteet

<https://kotlinlang.org/docs/>

<https://developer.android.com/guide>

<https://developer.android.com/studio/debug/>

<https://material-foundation.github.io/material-theme-builder/>

<https://carbon.now.sh>

<https://home.openweathermap.org>