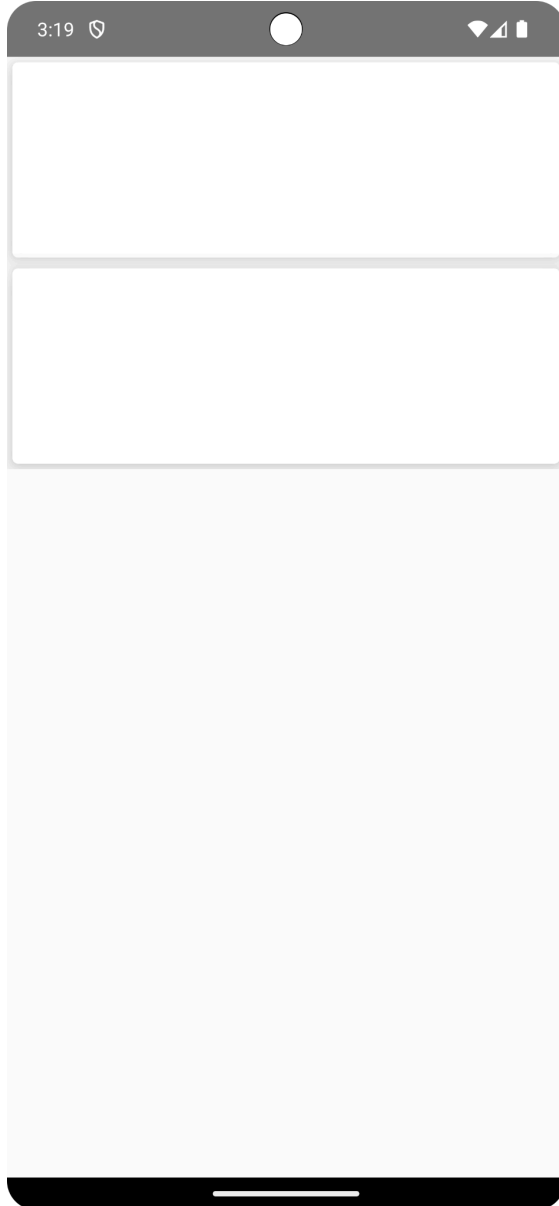


Practical Instructions

Practical 1 [20 minutes]

1. In Android Studio, navigate to “File” -> “Open”. In the file dialog, navigate to your local repository and then to the subdirectory /practical1/starter. The project should be compilable and runnable and will look like this (two empty cards).



2. Navigate to the `composeApp/src/commonMain/kotlin/com.jetbrains.weatherapp/App.kt` file. You need to fill in the details of three Composable functions (their bodies are currently marked as TODO).

- a. Flag - a Card to display the country's flag using the Image Loader library.



NB! Make sure you use `com.seiko.imageloader.rememberImagePainter` for **network** images, but you can use `org.jetbrains.compose.resources.painterResource` for **local** images.

- b. CountryNames - to display the country's common name and below it the official name.

Denmark
Kingdom of Denmark

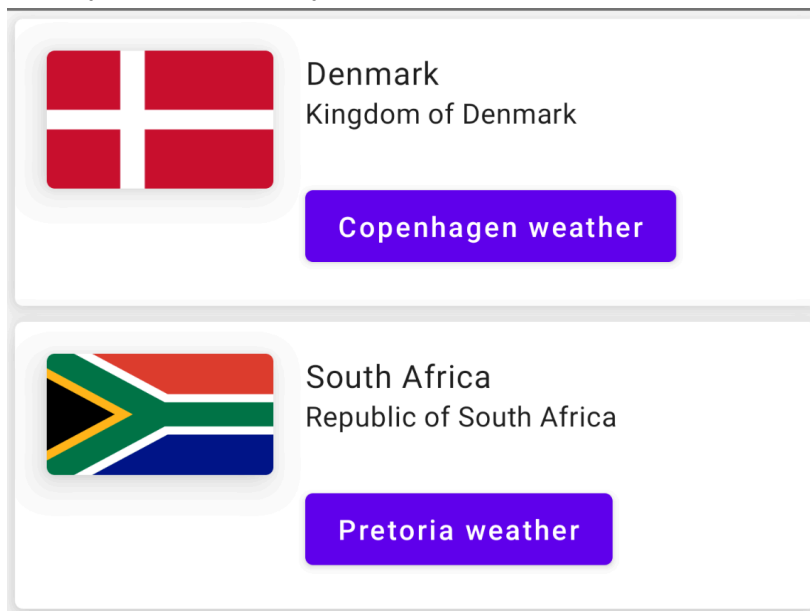
- c. WeatherButton - buttons to display the country's capital cities (there can be more than one).

Copenhagen weather

Troubleshooting

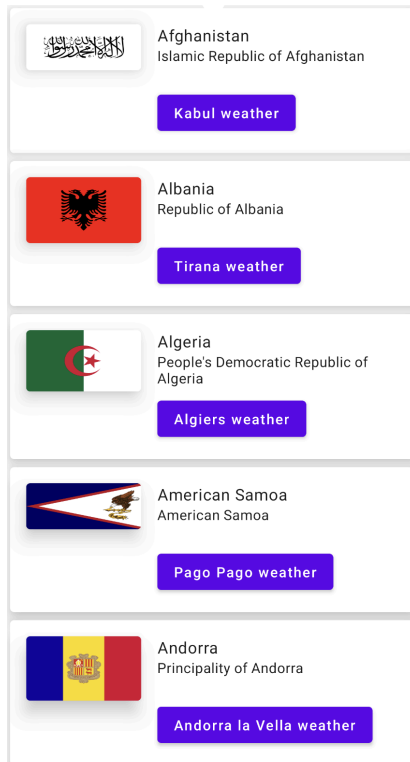
- If your flags aren't showing up (especially on Android), please check that your emulator is connected to the Internet.

3. When you are finished, your application will look like this:



Practical 2 [20 mins]

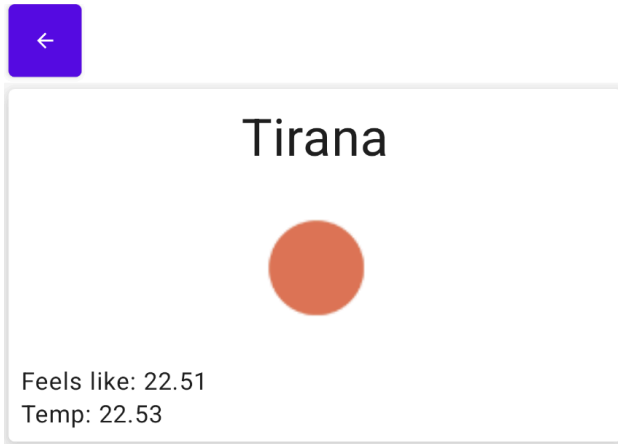
1. Navigate to “File” -> “Open”. In the file dialog, navigate to your local repository and then to the subdirectory /practical2/starter.
2. **NB: Make sure you have configured your WEATHER_API_KEY in your project’s local.properties. It was part of the Pre-steps instructions. If “Config” is missing, try a Gradle sync.**
3. The project should be compilable and runnable and should look like this:



4. Navigate to the `composeApp/src/commonMain/kotlin/com.jetbrains.weatherapp/network/WeatherApi.kt` file. You need to fill in the two TODOs:
 - a. Setting up the `httpClient`.
 - b. Getting the weather data from the webservice using the http client. The URL is:
`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${long}&appid=${Config.WeatherApiKey}&units=metric`
Where `lat` is the capital’s latitude, `long` is the capital’s longitude.
 - c. `Config.WeatherApiKey` will automatically be populated for you from your `local.properties`. The full name is `com.myapplication.Config`.

NB: If you get stuck, look at the `CountryApi.kt` file (without the mock server part).

5. Navigate to the `App.kt` file, and find the TODO in the `WeatherCard` Composable. You need to call the `WeatherApi` class’ `getWeather` suspend function here. When you are finished, you should be able to click on a capital city’s weather button and see it’s current weather, for example:



Practical 3 [20 minutes]

1. Navigate to “File” -> “Open”. In the file dialog, navigate to your local repository and then to the subdirectory /practical3/starter. **The project should not compile, because there is an expect function with missing actual implementations. Also remember to copy the WEATHER_API_KEY to the new local.properties file.**
2. Navigate to composeApp/src/commonMain/kotlin/com.jetbrains.weatherapp/location/CountryCodeService.kt. Notice that you need to provide an implementation for the getCountryCode function and CountryCodeService on each platform.
3. Navigate to the composeApp/src/androidMain/kotlin/com.jetbrains.weatherapp directory and create a new location directory and a file called AndroidCountryCodeService.kt, and provide the implementations there. To get the country code on Android, you can use:

```
import java.util.Locale
Locale.getDefault().country
```

NB: It's very important that the actual function is in the same package as the expect function.

4. Navigate to the composeApp/src/desktopMain/kotlin/com.jetbrains.weatherapp directory and create a new location directory and a file called DesktopCountryCodeService.kt, and provide the implementations there. To get the country code on Desktop, you can use:

```
import java.util.Locale
Locale.getDefault().country
```

5. Navigate to the composeApp/src/iosMain/kotlin/com.jetbrains.weatherapp directory and create a new location directory and a file called iosCountryCodeService.kt, and provide the implementations there. To get the country code on iOS, you can use

```
import platform.Foundation.*
NSLocale.currentLocale().objectForKey(NSLocaleCountryCode).toString()
```

6. By default the locale for Android will be the US, so that card will be displayed at the top of the countries list. To change this:

- a. Open “Settings” and search for “System Languages”.
 - b. You may need to first add another language, let’s say UK English.
 - c. Drag UK English above any other languages in the list.
 - d. Close and restart our app. The United Kingdom will then appear at the top of the countries list.
7. By default the locale for iOS will also be the US. To change this:
- a. Open “Settings” -> “General” -> “Language & Region”.
 - b. Click on “Region” and select a new region, let’s say United Kingdom.
 - c. Click on the “Change to United Kingdom” button. The phone will restart.
 - d. Open the app. The United Kingdom will then appear at the top of the countries list. This is an example of what it could look like.

