Forecast

The Forecast app is a simple weather app that shows the weather forecast for different places in Denmark. The app will show a list of location and the user can click on one of the locations to get detailed information about the weather forecast for the given location.

Forecast uses Yahoo's weather service as its dataprovider (https://developer.yahoo.com/yql/console/). The Overview screens shows a list of locations in Denmark. Every location is identified based on "woeid". The details screen uses the "woeid" to load the forecast for the given location.

Your task

Currently the app only contains the Overview screen. Your task is to add the Details screen, that is shown when the user taps on one of the locations from the Overview screen.
Setup the Details screen, so it shows a spinner while the data is being loaded. Additionally you can be add error handling, in case loading the data fails.

You are allowed to make changes to any parts of the code, that is already there. However, we do hope, you will strive to follow the architectural patterns applied in the current code.

Resources

If you are not familiar with *Clean Architecture*, there are a lot of resources around to check out. You can start with a general overview here:
https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

To keep the project simple, we have kept the use of third party libraries to a minimum. However, we do use a *Retrofit* for http requests coupled with *GSON* for json parsing. You can read more about how it works here:
http://square.github.io/retrofit/

Weather details data

The endpoint to get weather details based on a woeid:

```
https://query.yahooapis.com/v1/public/yql?q=select units, item from weather.forecast
where woeid=[WOEID]&format=json
```

Example:

```
https://query.yahooapis.com/v1/public/yql?q=select units, item from weather.forecast
where woeid=28362580&format=json
```

The json response will consist of a Query object, and it will look like this:

```
query: {
   count:    Int
   results:  <Result-object>
}

result: {
   channnel: <Channel-object>
}

channel: {
   item:     <Item-object>
}

item: {
   forecast: [<Forecast-object>]
}

forecast: {
   date:     String
   text:     String
}
```