# 5.6.3 Asynchronität Wiederholung II

Wiederholungsheet

## Die Wetter-App (jetzt asynchron! 🥳)

Das Ziel dieser Aufgabe ist es, eine Wetter-App zu entwickeln. Diese soll das Wetter an einem Standort vorhersagen. Es wird JSON geparst werden und eine API abgefragt werden.

### Aufgabe 0 (Projekterstellung + Git)

Erstelle ein neues Projekt (`wetter_app_asynchron`). Lege Git und public GitHub Repository an und pushe das Projekt. Füge den Link zu dem GitHub-Projekt hier ein.

**Deine Antwort:**

https://github.com/RaphaelBirkl1991/app_akadiemie_weather_app_async.git

### Aufgabe 1 (Implementierung der UI)

Baue zuerst die UI aus dem Screenshot **grob** nach. Die Werte können direkt gesetzt werden.

## Aufgabe 2 (Verwendung von JSON)

Nun sollen die Werte aus einem JSON kommen.

```
const jsonString = """
{
    "latitude": 48.78,
    "longitude": 9.18,
    "current": {
        "time": "2024-01-12T11:45",
        "temperature_2m": -3.6,
        "apparent_temperature": -7.0,
        "is_day": 1,
        "precipitation": 12.00
    }
}
""";
```

Gehe hierbei folgendermaßen vor:

1. Kopiere den **JSON**-String in deine App.
2. Wandle das **StatelessWidget** in ein **StatefulWidget** um.
3. Erstelle Variablen im **State**, die die verschiedenen Werte enthalten werden (Temperatur, gefühlte Temperatur, ob es Tag oder Nacht ist, ...).
4. Passe die verschiedenen Widgets so an, dass die Variablen verwendet werden.
5. Schreibe eine Methode des **StatefulWidget**-State, die die Werte aus dem **JSON**-String holt und in die entsprechenden Variablen setzt.
6. Verwende die Methode im **onPressed** des Button.
7. Rufe **setState()** auf, damit die neuen Werte auch angezeigt werden.

*Hinweis: Es ist ok, wenn die UI nach dem App-Start erst mal "null" für die Werte anzeigt.*

## Aufgabe 3 (Holen der Daten per HTTP)

Als letzten Schritt sollen die Daten von einer **API** geholt werden, damit sie aktuell angezeigt werden können.

Du kannst **entweder** selber *momentane* Wetterdaten zusammenklicken unter dem folgenden Link (wichtig: “Current Weather” auswählen):

https://open-meteo.com/en/docs/

**Oder** die **vorbereitete** URL verwenden:

```
const url =
'https://api.open-meteo.com/v1/forecast?latitude=48.783333&longitude=9.183333&curre
nt=temperature_2m,apparent_temperature,is_day,precipitation&timezone=Europe%2FBerli
n&forecast_days=1';
```

Jetzt können die Wetter-Daten von der **API** geholt, dekodiert und angezeigt werden.
Gehe hierbei folgendermaßen vor:

1. Füge die URL in deine App ein.
2. Adde das Package **`http`**.
3. Schreibe eine Funktion, die **JSON**-Wetter-Daten per **get** von der API holt.
    a. Diese sollte vermutlich **await** verwenden, weil **get** ein Future zurückgibt.
    b. Außerdem muss sie dann entsprechend **async** werden.
    c. Sie sollte wohl ein **Future<String>** zurückgeben.
4. Verwende nun die eben geschriebene Funktion, um in der Methode, die das **JSON** in die Variablen packt, den **JSON-String** zu bekommen.
5. Verändere die Methode, in der der **JSON-String** dekodiert wird, nun folgendermaßen:
    a. Verwende die eben geschriebene Funktion, um an den **JSON-String** zu kommen.
    b. Dieser kann wie bisher dekodiert werden.
6. Die Methode muss dann vermutlich auch **async** werden etc. 😉

**Viel Erfolg bei der Bearbeitung der Aufgaben!** 🥳

## Bonusaufgaben (jetzt wird es richtig interessant 😀)

### Aufgabe 4 (Datenklasse)

Die Wetterinformationen sollen nun in einer eigenen Klasse gespeichert werden.

Erstelle zunächst eine Klasse *WeatherData*. Diese Klasse ist **kein** Widget! Sie soll nur die Daten enthalten.

#### Verwendung von WeatherData

Speichere nun die Daten, die von der API kommen, in einem Objekt der Klasse *WeatherData***.** Ändere die schon vorhandenen Widgets so ab, dass sie die Daten aus der *WeatherData*-Klasse verwenden anstatt den Variablen.

*Hinweis: Üblicherweise wird eine fromJson(...)-Methode in einer Klasse wie WeatherData verwendet, um ein neues Objekt von WeatherData zu erstellen.*

### Aufgabe 5 (WeatherRepository)

Jetzt soll ein *WeatherRepository* für das Holen des Wetters zuständig sein.

1. Erstelle eine Klasse *WeatherRepository*, in einer eigenen Datei.
2. Erstelle im *WeatherRepository* eine Methode, um das Wetter abzurufen.
    a. Hier kann der Code aus der bisherigen Funktion in das *WeatherRepository* wandern.
    b. Diese Methode gibt ein *WeatherData* zurück.
3. Erstelle ein Objekt der Klasse *WeatherRepository*, damit die Methode für das Holen des Wetters verwendet werden kann.
4. Verwende das Wetter aus dem *WeatherRepository* für die Anzeige des Wetters.

Jetzt sollte das Wetter aus dem *WeatherRepository* kommen und der Aufruf schön abstrahiert sein 🎉

### Aufgabe 6 (Mehr Wetterdaten)

Vielleicht ist dir bei Verwendung der Wetter-API aufgefallen, dass von der API mehr Daten zurückkommen als aktuell angezeigt werden. Spiele etwas mit der API herum und zeige dann die weiteren Daten ebenfalls in der App an. Ein Beispiel wäre die Zeit der Daten.

### Aufgabe 7 (Wettervorhersage)

Aktuell werden noch *aktuelle Wetterdaten* verwendet, anstatt *zukünftige Wetterdaten*. Spiele wieder mit der API herum und schaue, welche Wetterdaten du für *Wettervorhersagen* bekommen kannst.

Baue dann die Anzeige dieser Daten in die App ein.