## Anforderung

Die Anforderung war, dass ich eine Application via React-Native für Android und iOS entwickeln soll, die folgendes macht. Die Anwendung soll die aktuelle Temperatur aus der Seite https://openweathermap.org/ in „°C“ ermitteln. Dies soll in einem Screen angezeigt werden, falls dies nicht der Fall sein sollte, soll eine Fehlermeldung erfolgen.
Das Projekt soll zum Schluss auf Github hochgeladen werden.

## Probleme

Ich bekam die Challenge am 23.09.2019. Nun hatte ich vorerst Probleme mit der aktuelle React Native Version, sprich 0.61.1, sie gab mir, beim builden des Projektes, die folgende Fehlermeldung:

```
BUILD FAILED in 38s

error Failed to install the app. Make sure you have the Android development environment set up: https://facebook.github.io/react-native/docs/getting-started.html#android-development-environment. Run CLI with --verbose flag for more details.
Error: Command failed: gradlew.bat app:installDebug -PreactNativeDevServerPort=8081

FAILURE: Build failed with an exception.

* What went wrong:
Could not determine the dependencies of task ':app:preDebugBuild'.
> Could not resolve all task dependencies for configuration ':app:debugRuntimeClasspath'.
   > Could not resolve org.webkit:android-jsc:+.
     Required by:
         project :app
      > Failed to list versions for org.webkit:android-jsc.
         > Unable to load Maven meta-data from https://jitpack.io/org/webkit/android-jsc/maven-metadata.xml.
            > Could not get resource 'https://jitpack.io/org/webkit/android-jsc/maven-metadata.xml'.
               > Could not GET 'https://jitpack.io/org/webkit/android-jsc/maven-metadata.xml'. Received status code 522 from server: Origin Connection Time-out

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.

* Get more help at https://help.gradle.org

BUILD FAILED in 38s

    at checkExecSyncError (child_process.js:621:11)
    at execFileSync (child_process.js:639:15)
    at runOnAllDevices (C:\Users\Sivri\Desktop\APERTO\test2\node_modules\@react-native-community\cli-platform-android\build\commands\runAndroid\runOnAllDevices.js:74:39)
    at buildAndRun (C:\Users\Sivri\Desktop\APERTO\test2\node_modules\@react-native-community\cli-platform-android\build\commands\runAndroid\index.js:158:41)
    at C:\Users\Sivri\Desktop\APERTO\test2\node_modules\@react-native-community\cli-platform-android\build\commands\runAndroid\index.js:125:12
    at processTicksAndRejections (internal/process/task_queues.js:93:5)
    at async Command.handleAction (C:\Users\Sivri\Desktop\APERTO\test2\node_modules\@react-native-community\cli\build\cliEntry.js:160:7)
```

Somit entschied ich mich für die React-Native Version 0.59.9, da einige kleiner Projekte, die ich in meinem Praktikum angelegt hatte, mit dieser Version einwandfrei abspielten.

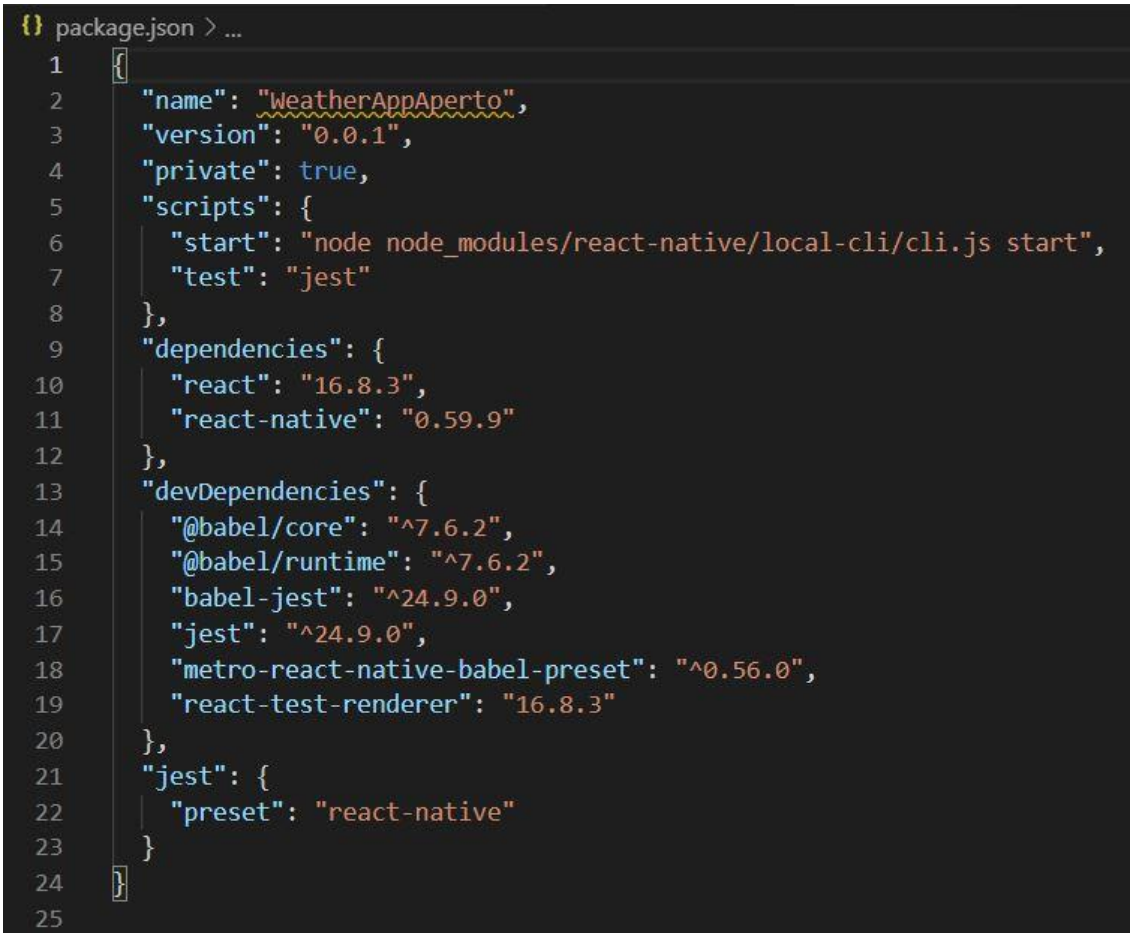
Problemlösung: Hierzu schreib ich „`react-native init WeatherAppAperto –version react-native@0.59.9`“ in die Konsole.

## Mein System

Da ich auf einem Windows Rechner entwickle kann ich leider nur die Android Version erstellen. Durch die Eingabe „react-native info“ in der Konsole innerhalb des Projektes kommt folgende Ausgabe:

```
info
  React Native Environment Info:
    System:
      OS: Windows 10
      CPU: (4) x64 Intel(R) Core(TM) i5-4210H CPU @ 2.90GHz
      Memory: 2.03 GB / 7.91 GB
    Binaries:
      Yarn: 1.17.3 - C:\Program Files (x86)\Yarn\bin\yarn.CMD
      npm: 6.10.3 - C:\Program Files\nodejs\npm.CMD
    IDEs:
      Android Studio: Version  3.5.0.0 AI-191.8026.42.35.5791312
```

Mein package.json sieht folgendermaßen aus:



```
{} package.json > ...
1   {
2     "name": "WeatherAppAperto",
3     "version": "0.0.1",
4     "private": true,
5     "scripts": {
6       "start": "node node_modules/react-native/local-cli/cli.js start",
7       "test": "jest"
8     },
9     "dependencies": {
10      "react": "16.8.3",
11      "react-native": "0.59.9"
12    },
13    "devDependencies": {
14      "@babel/core": "^7.6.2",
15      "@babel/runtime": "^7.6.2",
16      "babel-jest": "^24.9.0",
17      "jest": "^24.9.0",
18      "metro-react-native-babel-preset": "^0.56.0",
19      "react-test-renderer": "16.8.3"
20    },
21    "jest": {
22      "preset": "react-native"
23    }
24  }
25
```

Android Emulator:
Android Studio.
Pixel 2 API 28 mit einer Android Version 9.0

| Type | Name | Play Store | Resolution | API | Target | CPU/ABI | Size on Disk |
|---|---|---|---|---|---|---|---|
| | Pixel 2 API 28 | | 1080 × 1920: 420dpi | 28 | Android 9.0 (Google Play) | x86 | 16 GB |


## Vorgehensweise

Da ich die Api „OpenWeatherMap“ nicht kann erkundigte ich mich vorerst darüber. Ich ließ mir ein ApiKey zu schicken und probierte aus, Daten zu erfassen, vorerst manuell.
Ich entschied mich für die Version mit der Latitude und Longitude. Somit entstand folgender Link für mich:

http://api.openweathermap.org/data/2.5/weather?lat=52.517&lon=13.3883&APPID=473c27e28a114c10026d078611455a86
Hierbei bekam ich als Ausgabe ständig ein Wert zwischen 280 und 290. Ich habe John vom Aperto Team diesbezüglich angeschrieben, um nachzufragen, ob mit der Api alles okay ist.
Nach der Versendung des Mails fiel mir auf, dass ich die Umformung „`&units=metric`" vergessen hatte hinten einzufügen. Das lag meines achtens wohl dran, weil ich seit geraumer Zeit mir ständig die Link angeguckt habe und ansatzweise auswendig konnte.
Netterweise hat mir nach kurzer Zeit auch ein weiteres Teammitglied eine Mail geschrieben, dass die Formatierung fehlte, jedoch war ich erstmal froh den Fehler selber gefunden zu haben.

Da ich mit der Latitude und Longitude arbeiten will, muss ich erstmal meiner Anwendung beibringen, wie sie auf die Standortfreigabe zugreift. Hierzu fügte ich folgenden Code:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

in den Unterordner „*android\app\src\main\AndroidManifest.xml*" ein.
Durch „`react-native run-andorid`" bekam ich das Fenster für die Erlaubnis der Standortfreigabe.

Als nächstes musste ich die API in die App mit einbinden. Hierzu weiste ich die jeweiligen Werte als `state` hinzu und ließ die in der FetchWeather Methode auslesen und einbinden. Mein ApiKey packte ich in eine externe .js  Datei und importierte dies in die App.js. Somit hatte ich die Möglichkeit die Werte beliebig zu ändern ohne die Link für die Api anzufassen.
Da die Link als json Format ist haben ich die genaue Temperatur innerhalb der FetchWeather Methode mit der Aussage „`json.main.temp`" definiert und mein zuvor fest gelegte Variable „`temprature`" überschrieben.

In der `componentDidMount()` Methode habe ich die zuvor erstellte Api Link, sprich das fetching, erneut eingebunden und mit der Standortfreigabe durch „`navigator.geolocation.getCurrentPosition`" verknüpft.

Laut Vorgabe sollte eine Benachrichtigung auftauche, falls keine Daten erfasst werden konnten. Hierbei dachte ich an einen Popup (Alert) Fenster, welches nach einer gewissen Zeit aufspringt und mir eine Information gibt.
Gelöst habe ich dies innerhalb der „`render()`" Methode. Hierzu erstellte ich eine Methode namens „`timer()`" und setzte dem eine „`if`" Abfrage.

Da der aktuelle Wert von „`isLoading`" innerhalb der „`render()`" Methode „`false`" ist sollte meine „`if`" Abfrage folgendes überprüfen.

Wenn „`isLoading`" den gleichen Wert und Typ hat, setzt er nach 5 Sekunden ein Alert auf, falls nicht, lädt er „`isLoading`".

Getestet habe ich dies innerhalb des Views als eine ganz normale Text Ausgabe.

Die Anwendung läuft nun „nackt" auf dem Emulator.

## Design

Das `StyleSheet` packte ich in eine externe „.js" Datei in dem Unterordner Style. Dort schrieb ich alles Style Komponenten für alle Vorhandenen „.js" Dateien.

Zugriff auf die jeweiligen Style Komponente erlangte ich durch einen Import in der jeweiligen „.js“ Datei.

```
var styles = require('./source/components/Style/style')
```

Das aufrufen ist hierbei gewöhnlich durch den Aufforderung „styles.xx“.
Grund für diese Variante ist, dass es meines Achtens sauberer aussieht.

Die „`render()`“ Methode enthält ein „`const`“ Deklaration, die den Wert von `isLoading` von der `state` übernimmt.

In die „`return()`“ Methode fing ich vorerst an die App zu gestallten.
Die Ausgabe der Temperatur erfolgte in der Mitte des Views, drüber kam eine Überschrift und ein kleines Image. Die jeweiligen styles wurden in der „Style.js“ definiert.

Für eine besondere Schriftart importierte ich die „.ttf“ Fonts in den Unterordner „*android\app\src\main*“ hier erstellte ich ein neues Verzeichnis namens „*assets*“ und kopierte die „.ttf“ Fonts rein. Durch den Befehl „`fontFamily`“ in den jeweiligen styles, lässt sich das gewünscht Font anzeigen.

Das Icon in der App ladete ich in den „images“ Verzeichnis, welches sich in dem „*assets*“ Ordner befindet. „*source\components\assets\images*“
Das Bild ließ ich durch die Aufforderung in der `return` Methode durch folgende Aufforderung anzeigen.

```
<Image source={require('./source/components/assets/images/weather.png')} style
={styles.icon}/>
```

Anschließend lud ich ein <`ActivityIndicator/`> ein, um die Fetching Zeit zu überbrücken und anschaulich zu machen.

Die Methode „`timer()`“, welches auf funktionalität der Api prüft kam ebenfalls in die „`return()`“ Methode rein.

**Da ich (leider) nur ein Windows Rechner besitze hatte ich leider nicht die Möglichkeit mit xCode die iOS Version zu überprüfen.**