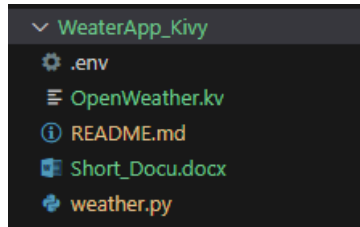


Da ich keinen Bock hatte einen mega Aufriss an Doku zu machen, hier einfach ein paar Low-Level Pictures, um euch zu zeigen, was ich zum Anbieten hätte 😊

Für die App braucht es nicht viel.



1. Eine **.env** Datei, in welcher der **App-Key** liegt. Diesen bekommt man, wenn man auf <https://openweathermap.org/> einen Account anlegt.
2. Eine **OpenWeather.kv** Datei. Diese Datei kann man sich vorstellen wie eine index.html. Hier definiert man wie seine Applikation aussehen soll und zum Teil auch agieren soll.
3. Eine **README.md**. Macht man halt 😊
4. Eine **Short_Docu.docx**. Die lest ihr gerade
5. Eine **weather.py** Datei. In dieser passiert die Magie. Zugriff auf <https://openweathermap.org/> um dort die Wetter-Daten abzuholen, aufbereiten, weiterleiten und Eingaben der App entgegennehmen.

Die OpenWeather.kv Datei:

```
Internship > WeaterApp_Kivy > OpenWeather.kv
1  BoxLayoutExample:      # referenced to the calss 'BoxLayoutExample' => Main Interface
2
3
4  <BoxLayoutExample>:
5      orientation: 'vertical'
6      spacing: 10
7
8      Label:
9          text: 'City'
10     TextInput:
11         id: city
12         hint_text: "Wangen"
13
14     Label:
15         text: 'State (token)'
16     TextInput:
17         id: state
18         hint_text: "BW"
19
20     Label:
21         text: 'Country'
22     TextInput:
23         id: country
24         hint_text: "Germany"
25
26     Button:
27         text: "Search"
28         on_press: root.on_button_click(self, city.text, state.text, country.text)
29
30     Button:
31         text: "Application Quit"
32         on_press: app.stop()
```

Das hier ist eine **Kivy-Datei**. Das ist ein spezielles Framework zum Erstellen von App Oberflächen/Applikationen. Gibt tausend andere wie z.B. *React*, *Flutter* usw... Aber ich habe jetzt eben mal Kivy genommen für den ersten Versuch. Ich kenne mich da aber auch noch nicht so richtig aus!

Wie ihr sehen könnt, passiert hier nicht viel. Aber weiter unten seht ihr auch, dass sie auch nicht viel ausgibt oder Fancy aussieht. Hier kann man sich noch austoben 😊

Die weather.py Datei

```
You, 19 minutes ago | 1 author (You)
1 import requests
2 from dotenv import load_dotenv
3 import os
4 from dataclasses import dataclass
5 from kivy.app import App
6 from kivy.uix.boxlayout import BoxLayout
7
8
9 # pull data from file .env
10 load_dotenv()
11 api_key = os.getenv('API_KEY') # crap API_KEY from file
12
13
14 # ##### methods #####
15
16 # give me location information
17 def get_lan_lon(city_name, state_code, country_code, API_key):
18     print("[method: get_lan_lon(city_name, state_code, country_code, API_key)] - call successful")
19     resp = requests.get(
20         f'http://api.openweathermap.org/geo/1.0/direct?q={city_name},{state_code},{country_code}&appid={API_key}').json() # c
21     data = resp[0]
22     lat, lon = data.get('lat'), data.get('lon')
23     return lat, lon
24
25
26 # give me the location weather data
27 def get_current_weather(lat, lon, API_key):
28     print("[method: get_current_weather(lat, lon, API_key)] - call successful")
29     resp = requests.get(
30         f'http://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_key}&units=metric').json() # gibe me b
31     data = WeatherData(
32         main=resp.get('weather')[0].get('main'),
33         description=resp.get('weather')[0].get('description'),
34         icon=resp.get('weather')[0].get('icon'),
35         temperature=int(resp.get('main').get('temp'))
36     )
37     return data
38
39
40 def main(city_name, state_name, country_name):
41     print("[method: main(city_name, state_name, country_name)] - call successful")
42     lat, lon = get_lan_lon(city_name, state_name, country_name, api_key)
43     weather_data = get_current_weather(lat, lon, api_key)
44     return weather_data
45
46
47 # ##### class #####
48
49 # The name of THIS class must be the same as the kv-file + 'App'
50 You, 19 minutes ago | 1 author (You)
51 class OpenWeatherApp(App):
52     print("[class: OpenWeatherApp(App)] - call successful")
53     pass
54
55 You, 19 minutes ago | 1 author (You)
56 class BoxLayoutExample(BoxLayout):
57     print("[class: WeatherWidget(BoxLayout)] - call successful")
58     pass
59
60     def on_button_click(self, widget, city, state, country):
61         print("[method: on_button_click(self)] - button pressed")
62         if city and state and country:
63             print(f"{city}; {state}; {country}")
64             lat, lon = get_lan_lon(city, state, country, api_key)
65             weather_data = get_current_weather(lat, lon, api_key)
66             print(f"Main: {weather_data.main} | Description: {weather_data.description} | Temperature: {weather_data.temperature}")
67             print("")
68             widget.text = str(weather_data)
69         else:
70             widget.text = "ERROR: Missing inputs"
71
72 # with '@dataclass' you dont need '__init__' and '__repr__' and '__eq__' method
73 You, 19 minutes ago | 1 author (You)
74 @dataclass
75 class WeatherData:
76     print("[class: WeatherDate] - call successful")
77     main: str
78     description: str
79     icon: str
80     temperature: int
81
82 # ##### _main_ #####
83
84 if __name__ == "__main__":
85     print("[_main_] pass")
86     #lat, lon = get_lan_lon('Wangen', 'BW', 'Germany', api_key)
87     #print([(lat, lon, api_key)]: ", get_current_weather(lat, lon, api_key))
88     OpenWeatherApp().run()
89
```

Das ist zum Großteil der Python Code von <https://www.youtube.com/watch?v=JCD7YdOSsWU>. Nur habe ich hier das ganz eben nicht als Web-App gemacht, sondern eben in Kivy als „Handy“ App realisiert. Daher ein paar Abwandlungen.

Trommelwirbel.... Hier die development status Application 😊

The screenshot shows the OpenWeather application interface with the following elements and labels:

- City**: Input field containing "Wangen". Label: "Eingabefeld für die Stadt".
- State (token)**: Input field containing "BW". Label: "Eingabefeld für das Bundesland/Bundesstaat".
- Country**: Input field containing "Germany". Label: "Eingabefeld für das Land".
- Search**: Button. Label: "Start/Such Button".
- Application Quit**: Button. Label: "Beenden der App".

Werden korrekte Wert eingegeben, ändert sich aktuell einfach der „Search“ Button und zeigt in Code-Syntax das aktuelle Wetter der Region an.

The screenshot shows the OpenWeather application interface with the following elements:

- City**: Input field containing "Miami".
- State (token)**: Input field containing "FL".
- Country**: Input field containing "USA".
- Search**: Button, now displaying the weather data in code syntax: `WeatherData(main='Clouds', decription='scattered clouds', icon='03d', temperature=29)`. This button is circled in red.
- Application Quit**: Button.

Wird ein Parameter vergessen, zeigt der „Search“ Button ein ERROR an. Wird eine fehlerhafte Eingabe gemacht, stürzt das Programm ab 😊 Muss noch behoben werden^^

The screenshot shows the error message displayed on the Search button: "ERROR: Missing inputs".

Web-App Beispiel

Als Beispiel habe ich hier noch einen Screenshot der Web-App Variante. Die schaut besser aus, macht aber dasselbe.

Weather App

City Name: Wangen

State (token): BW

Country: Germany

Find

Clouds: overcast clouds

15 C

That's it!