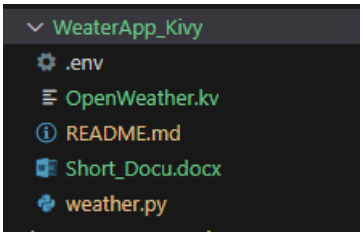
Da ich keinen Bock hatte einen mega Aufriss an Doku zu machen, hier einfach ein paar Low-Level Pictures, um euch zu zeigen, was ich zum Anbieten hätte 😊

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

```
BoxLayoutExample:    # referenced to the calss 'BoxLayoutExample' => Main Interface


<BoxLayoutExample>:
    orientation: 'vertical'
    spacing: 10

    Label:
        text: 'City'
    TextInput:
        id: city
        hint_text: "Wangen"

    Label:
        text: 'State (token)'
    TextInput:
        id: state
        hint_text: "BW"

    Label:
        text: 'Country'
    TextInput:
        id: country
        hint_text: "Germany"

    Button:
        text: "Search"
        on_press: root.on_button_click(self, city.text, state.text, country.text)

    Button:
        text: "Application Quit"
        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 ausschaut. Hier kann man sich noch austoben 😊

Die weather.py Datei

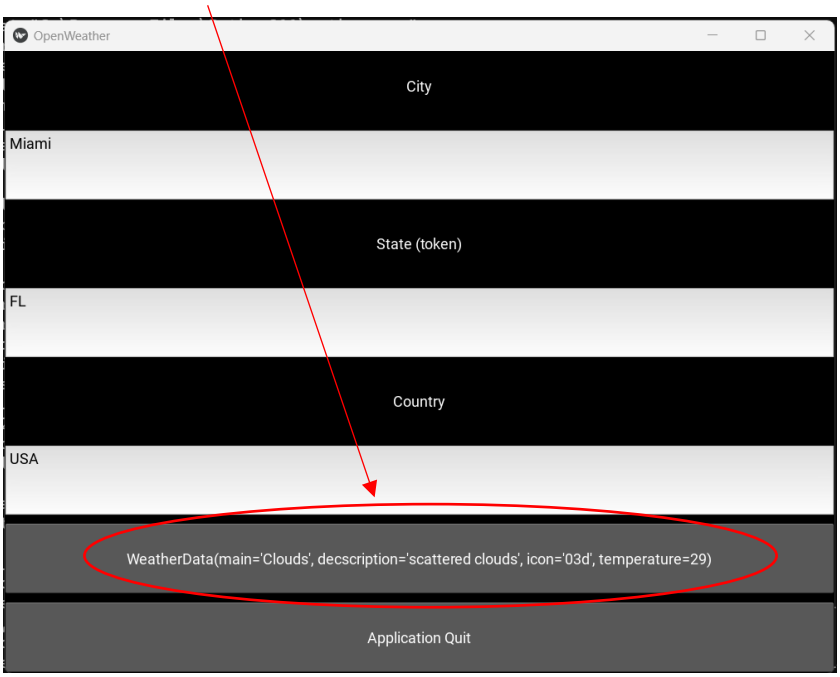
```python
import requests
from dotenv import load_dotenv
import os
from dataclasses import dataclass
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout


# pull data from file .env
load_dotenv()
api_key = os.getenv('API_KEY')    # crap API_KEY from file


# ############################## methods ############################

# give me location information
def get_lan_lon(city_name, state_code, country_code, API_key):
    print("[method: get_lan_lon(city_name, state_code, country_code, API_key)] - call successful")
    resp = requests.get(
        f'http://api.openweathermap.org/geo/1.0/direct?q={city_name},{state_code},{country_code}&appid={API_key}').json()    #c c
    data = resp[0]
    lat , lon = data.get('lat'), data.get('lon')
    return lat, lon


# give me the location weather data
def get_current_weather(lat, lon, API_key):
    print("[method: get_current_weather(lat, lon, API_key)] - call successful")
    resp = requests.get(
        f'http://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_key}&units=metric').json()    # gibe me b
    data = WeatherData(
        main=resp.get('weather')[0].get('main'),
        decscription=resp.get('weather')[0].get('description'),
        icon=resp.get('weather')[0].get('icon'),
        temperature=int(resp.get('main').get('temp'))
    )
    return data


def main(city_name, state_name, country_name):
    print("[method: main(city_name, state_name, country_name)] - call successful")
    lat, lon = get_lan_lon(city_name, state_name, country_name, api_key)
    weather_data = get_current_weather(lat, lon, api_key)
    return weather_data
```

```python


# ############################## class ############################

# The name of THIS class must be the same as the kv-file + 'App'
You, 19 minutes ago | 1 author (You)
class OpenWeatherApp(App):
    print("[class: OpenWeatherApp(App)] - call successful")
    pass


You, 19 minutes ago | 1 author (You)
class BoxLayoutExample(BoxLayout):
    print("[class: WeatherWidget(BoxLayout)] - call successful")
    pass

    def on_button_click(self, widget, city, state, country):
        print("[method: on_button_click(self)] - button pressed")
        if city and state and country:
            print(f"{city}; {state}; {country}")
            lat, lon = get_lan_lon(city, state, country, api_key)
            weather_data = get_current_weather(lat, lon, api_key)
            print(f"Main: {weather_data.main} | Description: {weather_data.decscription}| Temperature: {weather_data.temperature}")
            print("")
            widget.text = str(weather_data)
        else:
            widget.text = "ERROR: Missing inputs"


# with '@dataclass' you dont need '__init__' and '__repr__' and '__eq__' method
You, 19 minutes ago | 1 author (You)
@dataclass
class WeatherData:
    print("[class: WeatherDate] - call successful")
    main: str
    decscription: str
    icon: str
    temperature: int


# ############################## __main__ ############################

if __name__ == "__main__":
    print("[__main__] pass")
    #lat, lon = get_lan_lon('Wangen', 'BW', 'Germany', api_key)
    #print("[(lat, lon, api_key)]: ", get_current_weather(lat, lon, api_key))
    OpenWeatherApp().run()
```

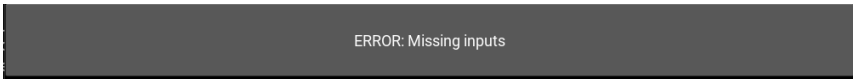
Das ist zum Großteil der Python Code von https://www.youtube.com/watch?v=JCD7YdOSsWI. Nur habe ich hier das ganz eben nicht als Web-App gemacht, sondern eben in Kivy als „Handy“ App realisiert. Daher ein paar Abwandlungen.

Trommelwirbel…. Hier die ***development status Application*** 😊

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

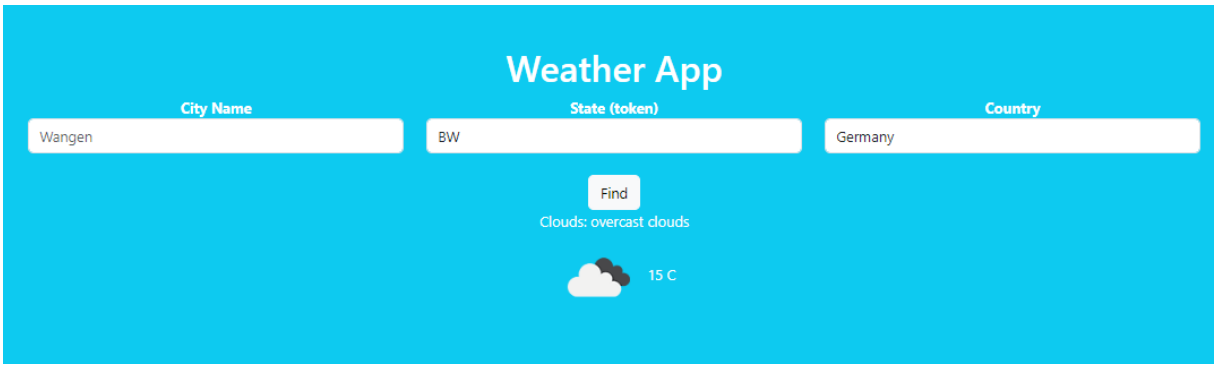


Wird ein Parameter vergessen, zeigt der „Search“ Button ein ERROR an. Wird eine fehlerhafte Eingabe gemacht, stürzt das Programm ab 😊 Muss noch behoben werden^^



Web-App Beispiel

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



That's it!