VYTAUTAS MAGNUS UNIVERSITY
FACULTY OF INFORMATICS
DEPARTMENT OF APPLIED INFORMATICS

INTERNET TECHNOLOGY (INF3002_EN)

API documentation weather

Student: Dmytro Vodianytskyi IF2100075

Lab assistant: Vytautas Barzdaitis

Kaunas, 2023

Contents

1 This is a Django web application that retrieves and displays weather information from the OpenWeatherMap API.

The main components of the application include:

1.1 weather_get.py

A module that contains two functions:

- `get_temperature(city_name, api_key)`: A function that retrieves the current temperature and weather data for a given city using the OpenWeatherMap API. The function takes in the city name and API key as arguments, and returns the temperature in Celsius and the raw JSON data.

- `evaluation(data)`: A function thatparses the raw JSON data returned by the `get_temperature()` function and returns a dictionary containing relevant weather information such as temperature, humidity, wind speed, and weather description.

1.2 urls.py

A Django module that maps URLs to views. In this application, the only URL is the root URL (`/`) and (`weather/`), which maps to the `weathe` view.

1.3 views.py

A Django module that contains the logic for handling HTTP requests and returning HTTP responses. In this application, the `weather` view retrieves the weather information for a given city using the `get_temperature()` and `evaluation()` functions and renders the data in an HTML template.

```
from django.shortcuts import render
from django.contrib.auth.decorators import login_required

from .weather_get import get_temperature, api_key, evaluation
```

Necessary imports

```
@login_required
def index(request):
    return render(request, 'API/index.html')
```

renders index page for `API` app.

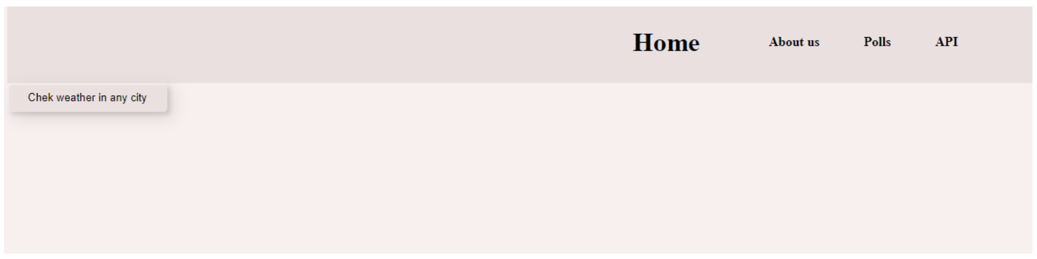


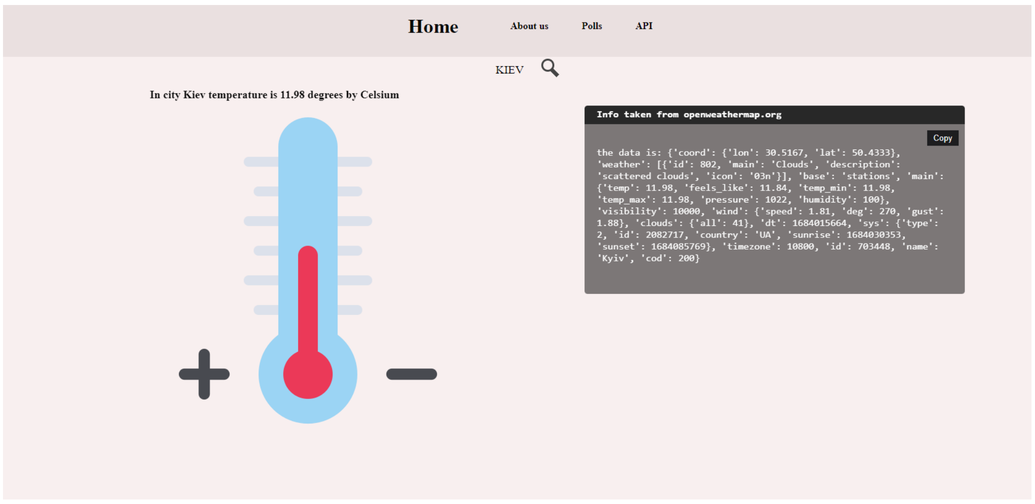
Figure 1: Screenshot of the index page

```
@login_required
def weather(request):
    if request.method == 'GET':
        city_name = request.GET.get('site', 'Kiev')
    else:
        city_name = 'Kiev'

    try:
        temperature, data_json = get_temperature(city_name=city_name,
                                                api_key=api_key)
        eval = evaluation(data_json)
    except Exception as e:
        temperature = 'Write the correct name of the city'
        data_json = "Write the correct name of the city"
        eval = 'Unknown'


    context = {
        'temperature': temperature,
        'data_text': data_json,
        'evaluation': eval
    }
    return render(request, 'API/weather.html', context)
```

The view checks whether the request method is GET or not. If it is, it gets the value of the city name from the request's GET parameters. If not, it sets the city name to 'Kiev' by default.

It then calls the `get_temperature function`, which retrieves the current temperature and weather data for the given city using an API key. If there is an error during this process, the view returns an error message to the user.

The view then creates a context dictionary containing the temperature, weather data, and an evaluation of the weather data based on a custom algorithm. Finally, it renders the weather.html template with the context dictionary and returns the resulting HTML to the user.



In city Kiev temperature is 11.98 degrees by Celsium

Info taken from openweathermap.org

```
the data is: {'coord': {'lon': 30.5167, 'lat': 50.4333},
'weather': [{'id': 802, 'main': 'Clouds', 'description':
'scattered clouds', 'icon': '03n'}], 'base': 'stations', 'main':
{'temp': 11.98, 'feels_like': 11.84, 'temp_min': 11.98,
'temp_max': 11.98, 'pressure': 1022, 'humidity': 100},
'visibility': 10000, 'wind': {'speed': 1.81, 'deg': 270, 'gust':
1.88}, 'clouds': {'all': 41}, 'dt': 1684015664, 'sys': {'type':
2, 'id': 2082717, 'country': 'UA', 'sunrise': 1684030353,
'sunset': 1684085769}, 'timezone': 10800, 'id': 703448, 'name':
'Kyiv', 'cod': 200}
```

Figure 2: Screenshot of the weather page

1.4 templates/index.html

An HTML template that displays the weather information retrieved by the `weather` view. The template uses Django's templating language to render the weather data in a user-friendly format.

1.5 API key

To use the OpenWeatherMap API, you need to obtain an API key from their website (https://openweathermap.org/api). Once you have an API key, you can pass it as an argument to the `get_temperature()` function to retrieve weather information for a given city.

2 Usage

To use the application, follow these steps:

1. Start the Django development server by running `python manage.py runserver` in your terminal.

2. Open a web browser and navigate to `http://127.0.0.1:8000/API/weather/` to access the application.

3. Enter a city name in the search box and click the `Search` button to retrieve the weather information for that city.

3 Conclusion

In conclusion, this Django web application provides a simple interface for retrieving and displaying weather information from the OpenWeatherMap API. By following the usage instructions, you can easily set up and use the application to retrieve weather information for any city in the world.