

# Weather microservice

Der Wetter Mikroservice ist ein Dienst, welche aktuell über den Telegram Client angesprochen werden kann. Dabei kann der Service per Chat einfach angesprochen werden ohne vorher etwas einrichten zu müssen. Dieser Dienst antwortet dabei auf Fragen, welche das Wetter im Allgemeinen, aber auch im stündlichen betreffen, ebenso wie das vergangene Wetter. Aktuell ist eine Anfrage von bis zu 7 Tage in der Zukunft, sowie 5 Tage in der Vergangenheit möglich. Ab 47 Stunden in der Zukunft kann der Bot nur noch mit dem allgemeinen Wetter antworten, sprich keine stündliche anfrage.

## Table of content

1. [Weather microservice](#)
2. [Table of content](#)
3. [Getting Started](#)
  - a. [Prerequisites](#)
  - b. [Cloning](#)
  - c. [Installing](#)
4. [Overview](#)
  - a. [Structure](#)
  - b. [Functionalities](#)
    - I. [The scripts-folder](#)
    - II. [The services-folder](#)
      - A. [generateResponse.js](#)
      - B. [weatherService.js](#)
      - C. [fiveDayWeatherService.js](#)
  3. [The routes-folder](#)
5. [Further Development](#)
6. [Further Reading](#)
7. [Built With](#)
8. [Versioning](#)
9. [Authors](#)

## Getting Started

### Prerequisites

- [node.js](#)
- [express.js](#)

### Cloning

Get the source code by cloning its repository via [https: weather\\_microservice](https://weather_microservice)

## Installing

After cloning the repository, you will need to make sure that you have node and npm installed on your working system. To check if you already have node installed, try

```
node -version
```

Same for checking if npm is installed, just with npm instead of the node command

```
npm -version
```

If you don't have node or npm installed, download the Software via the links provided in [Prerequisites](#) or search for them via your preferred search engine.

After that install all necessary dependencies

```
npm install
```

Now you can start the local development server to play around with the API and make your calls

```
npm run dev
```

This will fire up a development server that listens on port 8000.

If you direct your browser to <http://localhost:8000/weather>, you will get the weather forecast for the current day for Berlin.

## Overview

The weather microservice is basically a *Node-Express-Backend*. Incoming requests are checked and specifically handled. It can give you a general forecast for the next five days or a detailed forecast for the current day.

## Structure

The microservice consists of four folders containing several scripts, which are designated to perform certain tasks. We have the *\*scripts\**-folder containing scripts, that will be called by cron-jobs mainly for caching purposes. Then we have the *\*services\**-folder containing files, that consist of functions useful to process incoming requests from the chatbot and to generate a formatted answer-string, that contains the weather forecast for Berlin. The *\*routes\**-folder consists of all the routes, that can be addressed. In the next chapters we will get into more details about the scripts and their functions.

## Functionalities

On request, this microservice makes calls to the [OpenWeatherMap API](#). The received data is processed by services that return a nicely formatted string containing the weather forecast for Berlin. Mainly this service was built throughout the Masterprojekt module that is a mandatory part of the media informatics master course of the Beuth University for Applied Sciences.

### The scripts-folder

This folder contains two scripts, that will be called by a cron job multiple times a day, since we can only do a maximum of 60 requests to the API per day. `getWeather.js` makes a request to the [OpenWeatherMap API](#) and caches the answer. After that `writeResponseFile.json` is called and generates a pretty formatted answer-string. Now everytime a User wants to know the current weather forecast, we can just read it out of the cached data and don't need to call the API.

### The services-folder

This folder consists of several services, that perform specific tasks for the microservice.

#### **`generateResponse.js`**

Creates a nicely formatted string from a weather-JSON-object and caches it.

#### **`weatherService.js`**

Makes a request to the OpenWeatherMap-API to get the current weather forecast and stores the response.

#### **`fiveDayWeatherService.js`**

If a weather forecast for the next days is requested, than this script requests the necessary data from the OpenWeatherMap-API and stores its response.

### The routes-folder

This folder contains all the routes, that can be addressed on this server. The `index.js` manages all the routes. We've only got two routes in our project. The `/swagger-route` leads you to the swagger documentation of this project. The `/weather-route` will be called by another component of the Beuthbot. It expects a

message object containing the necessary details for this service. It then calls all the functions needed to perform requests and generates an answer, which is finally send back as a response to the Chatbot.

## Further Development

This is still a work in progress, so functionalities and structure might still change during development

## Further Reading

- [OpenMensa API](#)

## Built With

- [Node.js](#)
- [Express.js](#)
- [Axios](#)

## Versioning

We use [SemVer](#) for versioning. For the versions available, see the [tags on this repository](#).

## Authors

- **Tolga Karaoglu**
- **Steven Sobkowski**

See also the list of [contributors](#) who participated in this project.

I.

Nutzungshinweis: Auf dieses vorliegende Schulungs- oder Beratungsdokument (ggf.) erlangt der Mandant vertragsgemäß ein nicht ausschließliches, dauerhaftes, unbeschränktes, unwiderrufliches und nicht übertragbares Nutzungsrecht. Eine hierüber hinausgehende, nicht zuvor durch *datenschutz-maximum* bewilligte Nutzung ist verboten und wird urheberrechtlich verfolgt.