

Test Task

Task: Develop a SDK for accessing a weather API (<https://openweathermap.org/api>)

Objective: The objective of this task is to develop a software development kit (SDK) that can be used by other developers to easily access a weather API and retrieve weather data for a given location.

Requirements:

The SDK must be written in any programming language from a list: Java, C#, NodeJS, Python, Ruby.

The SDK must be easy to use, with clear and concise documentation.

The SDK must provide an interface for querying the weather API and returning the weather data in a standard JSON format.

The SDK must include examples of how to use the SDK to retrieve weather data for a given location.

The SDK must handle any errors that might occur when accessing the weather API.

Any tool (including AI ones) can be used to carry out this task.

Deliverables:

The source code for the SDK in at least one programming language.

Documentation for the SDK, including installation and usage instructions.

Examples of how to use the SDK to retrieve weather data.

Any additional deliverables you may think useful and that would show either your expertise or knowledge of development processes.

Evaluation Criteria:

We intentionally don't provide evaluation criteria. Try to think autonomously about what would be important for this project, i.e. if you were the designer of this SDK (or part of the team working on it) for real use by real customers, what would you do? Everything that you provide (or fail to provide) will be critical for us to evaluate your current skill level and understanding of software projects.

Technical Documentation:

1. The SDK should accept API KEY for OpenWeatherAPI on initialization.
2. The SDK should accept the name of the city and return information about the weather at the current moment. The SDK returns information about the first city which was found by searching for the city name.
3. The SDK should store weather information about the requested cities and if it is relevant, return the stored value (Weather is considered to be up to date if less than 10 minutes have passed)
4. In order to save memory, the SDK can store information for no more than 10 cities at a time.
5. The SDK should have two types of behavior: on-demand and polling mode. In on-demand mode the SDK updates the weather information only on customer requests. In polling mode SDK requests new weather information for all stored locations to have zero-latency response for customer requests. Mode of the SDK should be passed as parameter on initialization.
6. The SDK's methods should throw an exception with a description of the reason in case of failure.
7. Advantage: design the process of creating the SDK so that you can work with different keys, while creating two copies of an object with the same key is not possible. Also add a method to delete the object.

Intended JSON structure in SDK API response:

```
{  
  "weather": {  
    "main": "Clouds",  
    "description": "scattered clouds",  
  },  
  "temperature": {  
    "temp": 269.6,  
    "feels_like": 267.57,  
  },  
  "visibility": 10000,  
  "wind": {  
    "speed": 1.38,  
  },  
  "datetime": 1675744800,  
  "sys": {  
    "sunrise": 1675751262,  
    "sunset": 1675787560  
  },  
  "timezone": 3600,  
  "name": "Zocca",  
}
```