WeatherThing

ID2012 Project report

Abyel Tesfay, Remo Scolati
May 10, 2022

Introduction

This report covers the project assignment for the ID2012 Ubiquitous Computing course. The goal of the project is to create a prototype related to the field of Ubiquitous Computing and IoT. The system should update a state that can be used to provide some service to a user by aggregating data from different dimensions, including real-world data captured by some sensors.

In this project we created a weather library system, *WeatherThing*, which takes input in the form of location, time, and user preference and returns an output as contextualized weather data. The purpose of the library is to help aggregate raw data from different APIs into large structured JSON objects that other systems can derive useful information from. The library can be built into clients, deployed in servers, in embedded devices etc.

Proposed solution

To get an overall understanding of what we wanted to build, we created a Design graph (see figure 1) which describes how the WeatherThing system should interact with outside devices as well as its internal components.

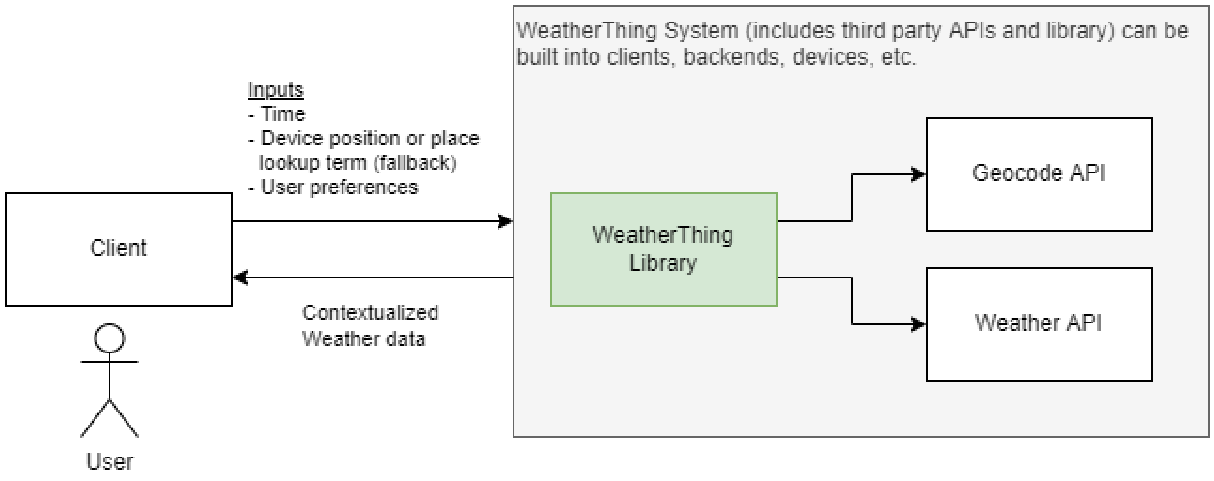

WeatherThing System (includes third party APIs and library) can be built into clients, backends, devices, etc.
Inputs
- Time
- Device position or place lookup term (fallback)
- User preferences
Client
User
Contextualized Weather data
WeatherThing Library
Geocode API
Weather API

Figure 1. Design graph of WeatherThing

The proposed system consists of the following components.

- The main library, which fetches data from the different APIs and aggregates the data into a single large data structure. This data is returned as output and can be used to derive useful information e.g. *The weather of the current location.*
- External APIs which provide the library with raw data from sensors.
    - Geocode API: An API which can be used both to create autocomplete suggestions as well as look up places for a given position (and vice-versa).
    - Weather API: An API that returns the current and hourly forecasted weather.

Further, we wanted to create a sample client implementation as a proof-of-concept of how the contextualized data could be shown to a user. We decided to implement a web application, since we expected to be able to use the browser API to access device data, perform requests, and let a user submit preferences. An early mock up for the client is shown in figure 2.

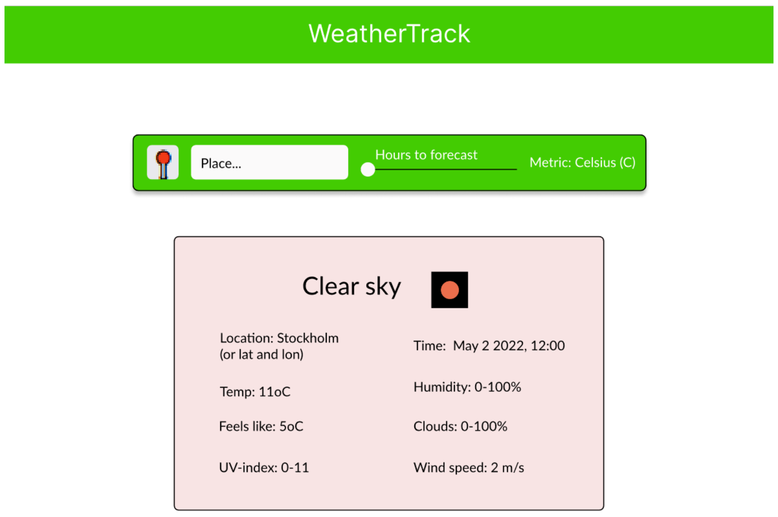


Figure 2. Mockup of the Web application

By separating the data aggregation and transformation from the presentation, the proposed implementation should be simple to integrate into clients and devices, for example websites and weather displays or smart clocks, or to be built into a backend, allowing the system to serve a wide range of different clients at the same time.

Implementation

Tools and frameworks

Since the proof of concept is a simple, static web application which includes both the main service and an example client, we used native browser APIs, that is, the Geolocation API[1] to access the device location and the Fetch API[2] to perform requests, and an HTML form to get the necessary input data.

[1] https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API
[2] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

Third party APIs

There is an abundance of free and commercial services which offer access to the required sensor data, and we picked the following two providers mostly based on the completeness of the data, availability of required endpoints, and whether a free tier is offered and how strict the usage quotas are.

We decided to rely on OpenWeather[3] to retrieve current and forecast weather data, while for geocoding and places search we settled for the Geoapify Location Platform[4] after trying some alternative solutions. See the linked documentations for details.

WeatherThing library

The main library component is implemented in JavaScript, since we decided to include it in a static client web application, and contains the logic for selecting, aggregating, and converting the requested weather data. It exposes the functionality to request weather data based on the following input.

- Time (current or forecast)
- Position (latitude and longitude)
- Some user preferences (measurement units)

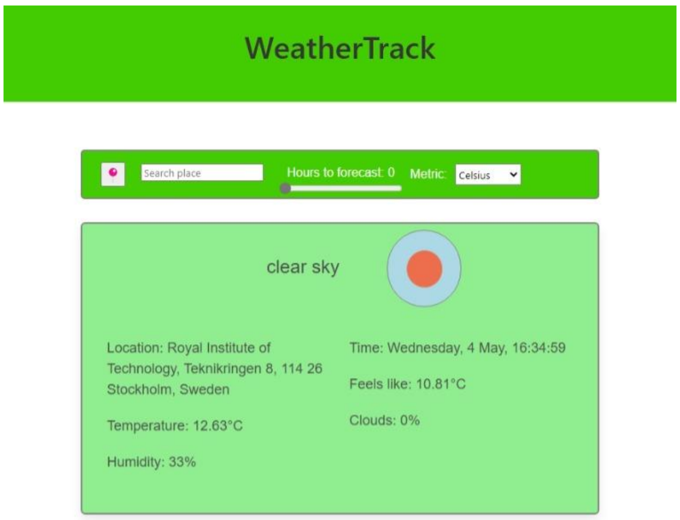
Based on the input data, the library performs a request on the weather API and filters and transforms the data. To convert a set of geographic coordinates into a more user-friendly property, the service uses the geocoding API to reverse geocode a position into a place or address.

Since the location is not always available to the client, for example if the user decides to disallow the access to the devices' location, we also provided a utility to suggest places or addresses based on a search term. This can be used to implement a search and autocomplete feature in the client.

Sample application

The implemented client application consists of a simple form which allows the user to set the location to either the devices' position or to a searched place, change the time for the requested weather through a slider, and to set their unit of measure preference. The input data is used to request the weather data through the WeatherThing library, and the result is then shown on the page. The deployed application (see screenshot in figure 3) is accessible online[5].

[3] https://openweathermap.org/api
[4] https://apidocs.geoapify.com/#docs
[5] https://u843.gitlab.io/id2012/


WeatherTrack

Search place · Hours to forecast: 0 · Metric: Celsius

clear sky

Location: Royal Institute of Technology, Teknikringen 8, 114 26 Stockholm, Sweden

Temperature: 12.63°C

Humidity: 33%

Time: Wednesday, 4 May, 16:34:59

Feels like: 10.81°C

Clouds: 0%

Figure 3. Final web application implementation

Use Cases

The idea behind the WeatherThing system is that we wanted a single source of data about the current or future weather based on location. Most public APIs give such services as raw data, but they miss important properties (location, time etc.) and must be combined with other APIs. Our library collects and aggregates data from three different sensor dimensions: weather (ambient), location and user preferences (e.g. units of measurement, time) , into one large, structured and readable object, as shown in figure 4.

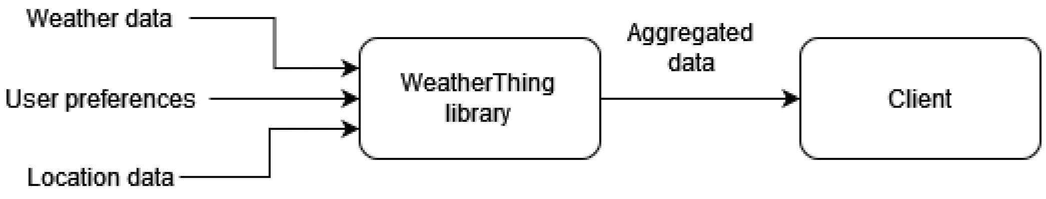


Figure 4. WeatherThing aggregates data from three different sources

The aggregated data can be used by clients to update their current state and therefore their behavior towards their users.

The WeatherThing system falls into the category of *calm technology*, as it does not crave our attention often or add any extra clutter to the environment. It is available when needed and can be dismissed as part of the environment when not important. Some use cases and possible future developments for the library include the integration into:

- A mirror with a built-in display that shows the current or forecast weather.
- A recommendation system, for example through push notifications, for clothing that fits the weather through the day.

- Existing home automation tools to control the ambient room lighting (intensity, color temperature) depending on the current weather properties e.g. sunlight, rain, (felt) temperature.

Conclusion

In conclusion, the project achieved its requirements by using public APIs to access sensors, which provides the system with real-world data. The system then aggregates data from three dimensions into a single large data object. This object can be used together with context to derive meaningful information from e.g. *the weather in Stockholm in 3 hours*.

The system can be integrated into existing clients and devices to help provide relevant information and service to the user such as fitting clothes for the day. This project has been a practical learning moment in understanding how computing can be applied to the environment, both for workspaces or personal life, and how sensors enable collection of real data which along with context gives us meaning.