

Analyze Part For Project

Requirements for Weather Forecast Project are below.

Functional Requirements:

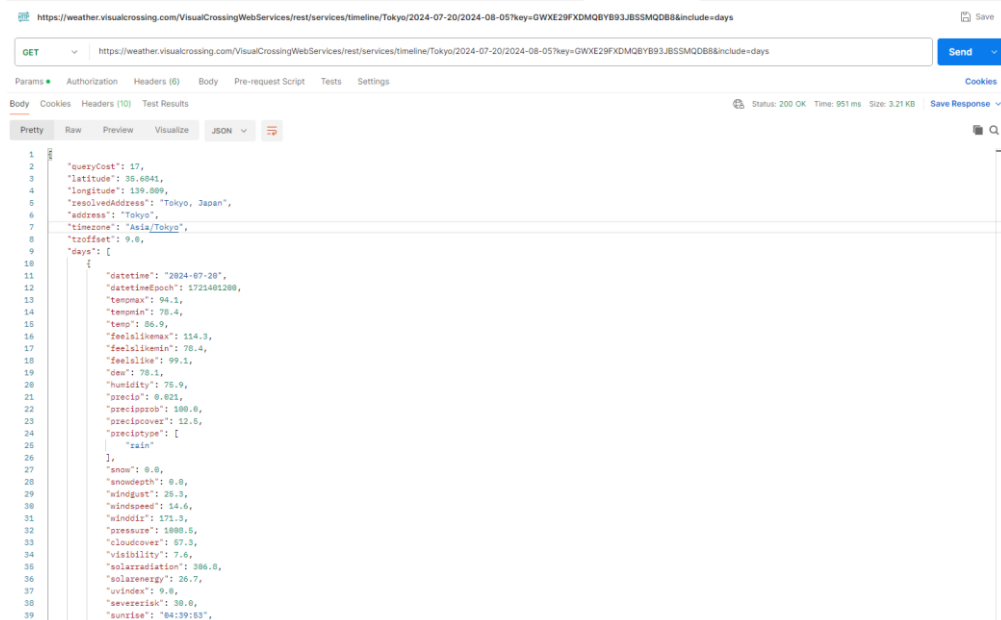
1. **City Selection:**
 - Implement a dynamic dropdown menu to select cities. Include cities like New Delhi, Istanbul, New York, and Paris.
2. **Temperature Display:**
 - Provide an option to display the temperature in Celsius or Fahrenheit.
3. **Date Range:**
 - Allow users to request weather data for a date range between the previous 7 days and the next 7 days.
4. **Asynchronous Data Processing:**
 - Fetch and process weather data asynchronously from the Weather API.
5. **Database Storage:**
 - Store weather data in PostgreSQL or SQLite.
6. **API Development:**
 - Create APIs to request weather data from the frontend.
7. **Testing:**
 - Prepare tests for the API to handle successful and unsuccessful responses.

Technical Requirements:

1. **Frontend Development:**
 - Use Angular for the frontend development. (Not a full implementation, just to show the results.)
2. **Backend Development:**
 - Develop the backend using Spring Boot with Java or Kotlin.
3. **Database Integration:**
 - Integrate PostgreSQL or SQLite for data storage.
4. **API Integration:**
 - Utilize the API for fetching weather data.
5. **Containerization:**
 - Use Docker for containerizing the application.
6. **Coding Principles:**
 - Ensure the code follows DRY (Don't Repeat Yourself) and DIE (Duplication is Evil) principles.
7. **Documentation:**
 - Document all the attribute formats and any other relevant details.

Design Part For Project

For the back-end side, project needs to work with third party web services for getting weather information according to relevant cities. For this requirement, Visual Crossing Weather API web service is used in the project. (<https://www.visualcrossing.com/weather-api>)



According to this API, response parameters are below.

"queryCost": 17,

"latitude": 35.6841,

"longitude": 139.809,

"resolvedAddress": "Tokyo, Japan",

"address": "Tokyo",

"timezone": "Asia/Tokyo",

"tzoffset": 9.0,

"days": [

{

"datetime": "2024-07-20",

"datetimeEpoch": 1721401200,

"tempmax": 94.1,

"tempmin": 78.4,

"temp": 86.9,

"feelslikemax": 114.3,

"feelslikemin": 78.4,

"feelslike": 99.1,
"dew": 78.1,
"humidity": 75.9,
"precip": 0.021,
"precipprob": 100.0,
"precipcover": 12.5,
"preciptype": [
 "rain"
],
"snow": 0.0,
"snowdepth": 0.0,
"windgust": 25.3,
"windspeed": 14.6,
"winddir": 171.3,
"pressure": 1008.5,
"cloudcover": 57.3,
"visibility": 7.6,
"solarradiation": 306.8,
"solarenergy": 26.7,
"uvindex": 9.0,
"severerisk": 30.0,
"sunrise": "04:39:53",
"sunriseEpoch": 1721417993,
"sunset": "18:54:10",
"sunsetEpoch": 1721469250,
"moonphase": 0.46,
"conditions": "Rain, Partially cloudy",
"description": "Partly cloudy throughout the day with late afternoon rain.",
"icon": "rain",
"stations": [
 "47727099999",
 "47670099999",
 "47642043313",
 "RJTI",

```

"RJTF",
"47674099999",
"47662099999",
"47671099999",
"47643099999",
"RJTY",
"47626099999",
"47682099999",
"RJTT",
"RJAA"
],
"source": "obs"
}

```

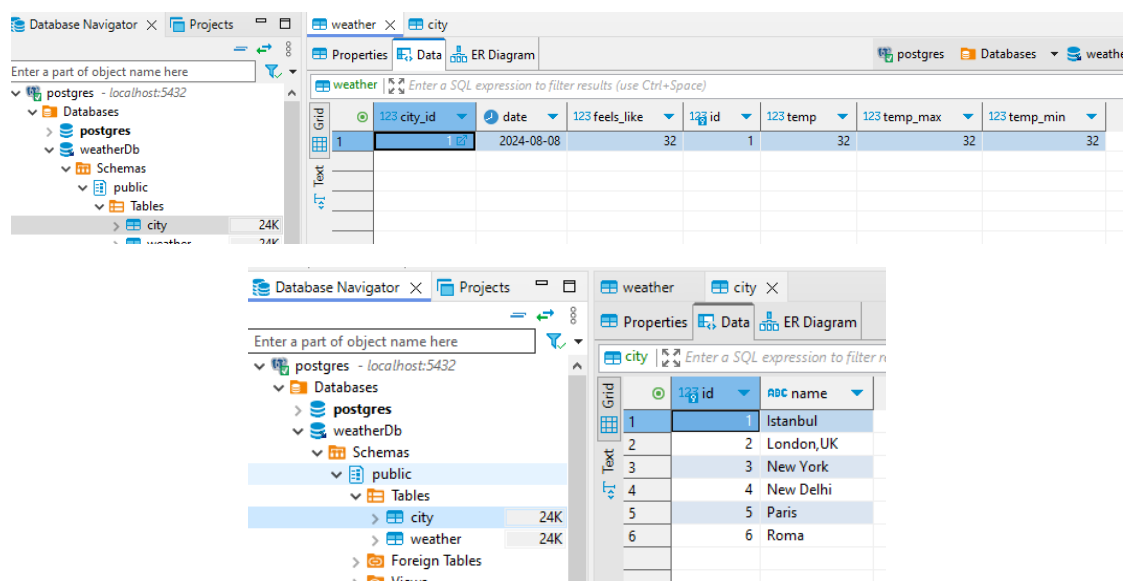
For integration this API, the project needs to a job for getting weather data from project's database. Also, these data should be inserted into the relevant table in the database. Then, the project should use these data from the database (like show).

Moreover, the project should have a back-end side for meeting the requirements from frontend side.

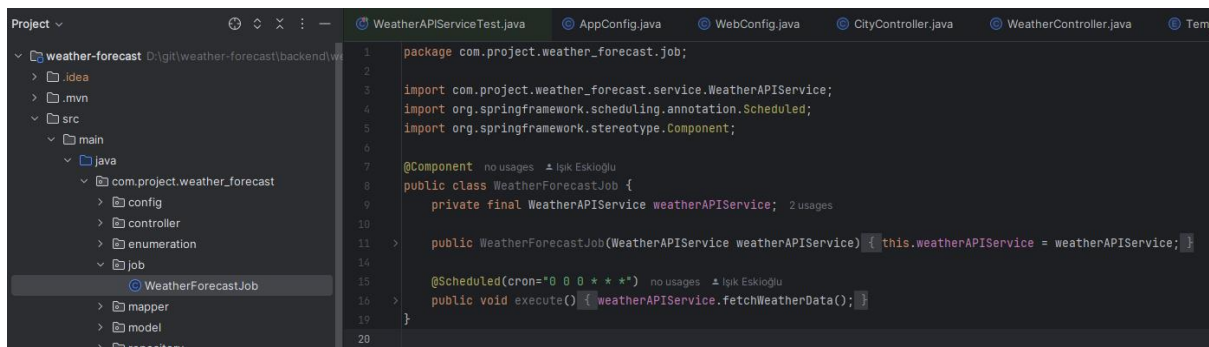
For the front-end side, the project should mainly have country, date range, Celsius/Fahrenheit selection parts and to show data for relevant selections.

Development Part For Project

For the back-end side, there are pictures which is for the project to database part below.



For getting weather data, the project has a job part and it gets new data for every day at 12:00 am.



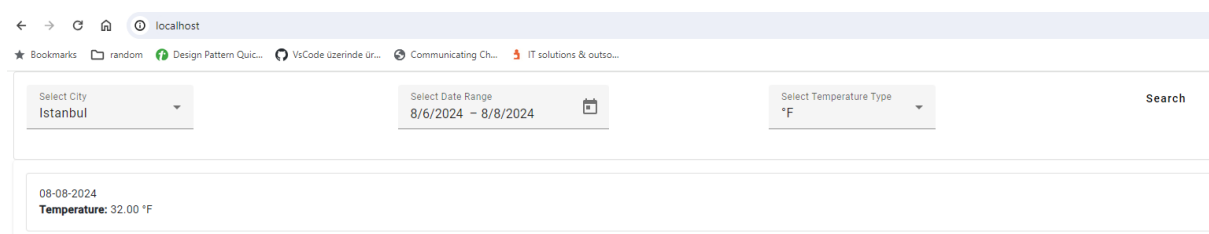
```
1 package com.project.weather_forecast.job;
2
3 import com.project.weather_forecast.service.WeatherAPIService;
4 import org.springframework.scheduling.annotation.Scheduled;
5 import org.springframework.stereotype.Component;
6
7 @Component
8 public class WeatherForecastJob {
9     private final WeatherAPIService weatherAPIService;
10
11     public WeatherForecastJob(WeatherAPIService weatherAPIService) { this.weatherAPIService = weatherAPIService; }
12
13     @Scheduled(cron="0 0 * * * *")
14     public void execute() { weatherAPIService.fetchWeatherData(); }
15 }
```

Converting the temperature Celsius and Fahrenheit, the project has a mapper class as below.



```
package com.project.weather_forecast.mapper;
import com.project.weather_forecast.model.Weather;
import org.springframework.stereotype.Service;
@Service
public class WeatherMapper {
    public Weather toCelsius(Weather weather) {
        weather.setFeelsLike(fahrenheitToCelsius(weather.getFeelsLike()));
        weather.setTemp(fahrenheitToCelsius(weather.getTemp()));
        weather.setTempMin(fahrenheitToCelsius(weather.getTempMin()));
        weather.setTempMax(fahrenheitToCelsius(weather.getTempMax()));
        return weather;
    }
    private double fahrenheitToCelsius(double fahrenheit) {
        return (fahrenheit - 32) / 1.8;
    }
}
```

For the front-end side, last view of the project is below.



Select City: Istanbul

Select Date Range: 8/6/2024 - 8/8/2024

Select Temperature Type: *F

Search

08-08-2024

Temperature: 32.00 *F