Weather Forecast

Bang Le

Agenda

- Objectives
- Requirements
- Assumptions
- Project plan
- System Design
- Detail Design
- Testing
- Space for improvements

Objectives

Main objectives of this project

- Project planning
- System/software design
- Implement Restful API using Spring boot
- Consume other API provider using RestTemplate and Json mapping
- Unit testing using Junit and Mockito
- Documentation using Swagger
- Logging using log4j
- Working with custom properties

Requirements

Non-functional requirements

- Scalability
- Fault tolerance
- Security
- Lightweight
- Performance
- Modularity

Requirements (cont.)

Functional requirements

- As a user running the application I can view tomorrow's predicted temperatures for a given zip-code in the United States so that I know which will be the coolest hour of the day.

Assumptions

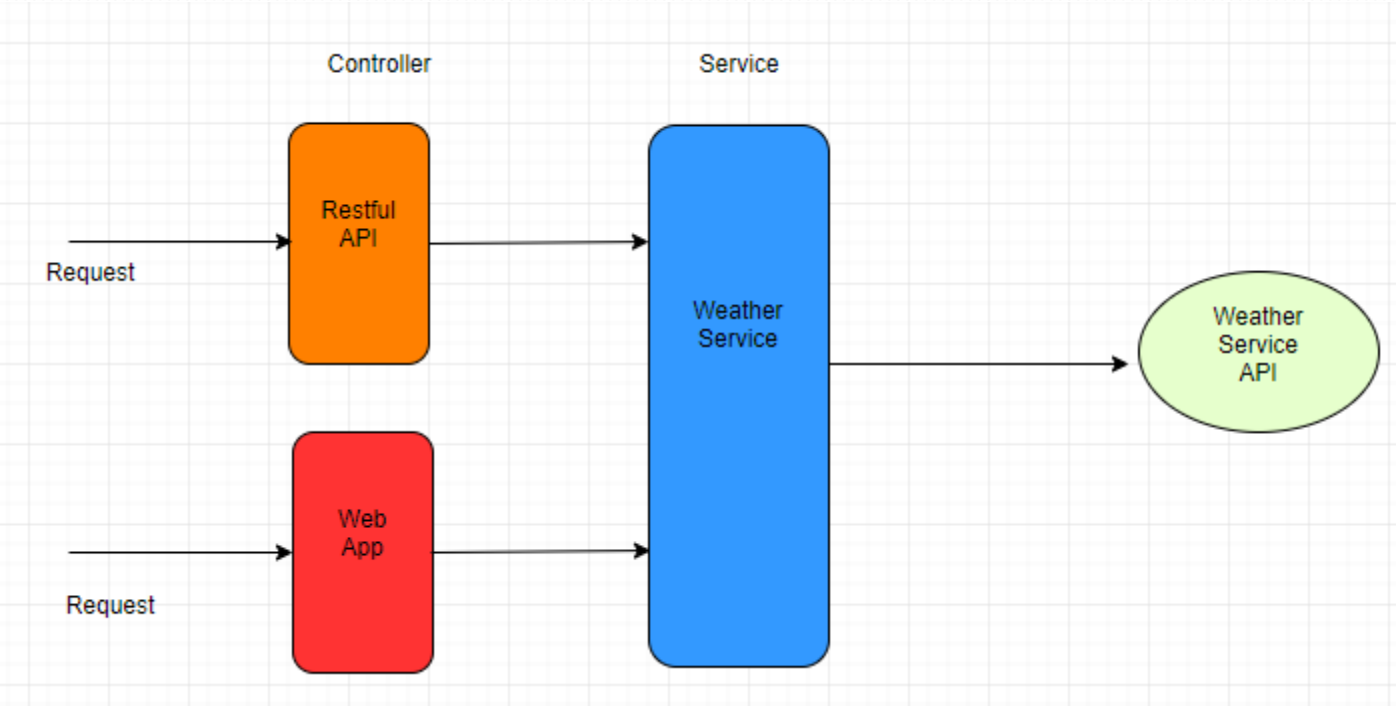
- We can choose any Weather API provider.
- We should build Restful API and Web app.
- No need to make the complete project. Just want to use technologies to implement required feature.
- With free API key, we cannot get hourly weather forecast, but each 3 hours. That is not a problem for demo project.

Project Plan

Item	Type	Priority	Estimation (Story point)
Research weather API	Task	High	2
Create github repository	Task	High	1
Design system	Task	High	1
Implement code framework	Task	High	2
As a user, I want to view forecast for tomorrow and see the coolest hour	Epic		
As a user, I want to input zipCode to view forecast	User Story	High	0
As a user, I want to view forecast for tomorrow to see the coolest hour	User Story	High	6
As a user, I want to view forecast for tomorrow to see the coolest hour in web page and json format	User Story	High	3
Add swagger	Task	Medium	1
Add unit test	Task	Medium	1
Add logging	Task	Medium	1
Readme file and slides	Task	Low	1

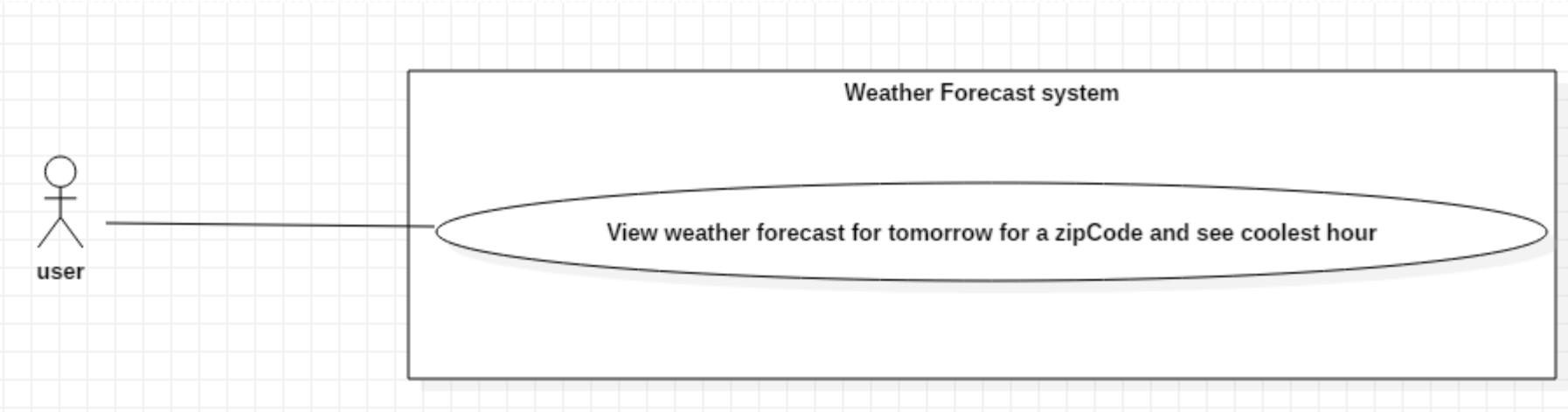
System Design

System design



Detail Design

Usercase Diagram



Detail Design (cont.)

Class Diagram

Detail Design (cont.)

User Sequence Diagram

Testing

- Testing activities:
  - API test using Postman
  - Unit test using Junit/Mockito

Demo

Weather Forecast

- **Elk Grove Village**

2019-11-15

06:00:00 : 22.46°F
09:00:00 : 21.43°F
12:00:00 : 21.72°F
15:00:00 : 26.01°F
18:00:00 : 33.1°F
21:00:00 : 33.51°F


```
{
    city: "Elk Grove Village",
    date: "2019-11-15",
  - hourlyTemperatures: [
      - {
            temperature: 21.9,
            hour: "06:00:00",
        },
      - {
            temperature: 21,
            hour: "09:00:00",
        },
      - {
            temperature: 21.45,
            hour: "12:00:00",
        },
      - {
            temperature: 25.86,
            hour: "15:00:00",
        },
      - {
            temperature: 33.1,
            hour: "18:00:00",
        },
      - {
            temperature: 33.51,
            hour: "21:00:00",
        },
    ],
}
```

Space for improvements

- Re-architecture the backend into microservices architecture to be able to scale later.
- Cache, Load balancing
- Improve security: add authentication/login and authorization and use token and data encryption during client/service communication.
- Improve exception handler using ControllerAdvice/AOP

Supported documents

- Code: https://github.com/banglekim/WeatherForecast
- Executable output file: https://github.com/banglekim/WeatherForecast/blob/master/src/ExecutableFile/weather-0.0.1-SNAPSHOT.jar
- Readme file: https://github.com/banglekim/WeatherForecast/blob/master/README.md