Backend Coding Challenge: Field Condition Statistics

We would like you to build a restful API that provides field conditions insights. Your API must be able to handle two use cases:

1. Record incoming field condition measurements
2. Provide field condition statistics from the last 30 days

Specifications

POST `/field-conditions`

Whenever a new field condition is captured, this endpoint will be called.

Request body:

```
{
    "vegetation" : 0.82,
    "occurrenceAt" : "2019-04-23T08:50Z"
}
```

Where:

- `vegetation` is any decimal number to represent the current level of vegetation at that date and time
- `occurenceAt` is defined using ISO_8601 date time format to represent the date and time of the field condition occurrence.

Response body: Empty.

GET `/field-statistics`

This is the main endpoint of this challenge. It must execute in constant time and space (O(1)). It must return the field condition statistics related to the past 30 days.

Response body:

```
{
    "vegetation" : {
        "min" : 0.01,
        "max" : 1.0,
        "avg" : 0.5
    }
}
```

Where:

- `min` is a decimal number specifying the lowest measurement
- `max` is a decimal number specifying the highest measurement
- `avg` is a decimal number specifying the average of measurements

Requirements

- Please complete code challenge in **Java** with **Spring Boot** framework.
- The delivery must contain the project and a README.md file that says at least how to build and run the project.
- You should add to the README.md your notes and assumptions that you made during the development if you think it would help us to see your motivation around technical decisions.

Some other requirements, which are obvious but listed here explicitly, in order to point out their importance.

- Your API must handle concurrent requests in a threadsafe manner
- Your API should function correctly and honor the contract defined in **Specifications** section above
- Your project's build must be functioning, and tests should also complete successfully (e.g. if maven is your build tool of choice, then ``mvn clean verify` should be successful)

How we will evaluate your project?

1. Your project must comply with all the defined requirements
2. We love tests. We will check how you've written your tests (clean, understandable, comprehensive and meaningful)
3. Feel free to use any supporting frameworks to show your experience and knowledge while simplifying your code.

If you feel like going an extra mile, there are a couple of things we would love to see how you do:

- Consider to add a persistence layer. (i.e. what data storage and why?) that could help you to solve potential future requirement on building historical statistics for a given date range.
- Document your API using Open API Specification / Swagger