ICC SOFT Test

For this test, you must create a RESTFul Application based on the following requirements:

- Java
- Spring Boot Framework
- MySQL Database
- Maven

The REST API exposed by your application must communicate with MySQL DB to make create, update, read and delete operations for Stocks. Each Stock has its name and Quotes. The Quotes represents the values of each Stock in a given time. Is important to note the each Stock name must be unique and identify each Stock uniquely.
Below there is an example of a Stock entity (in JSON format):

```
{
    "name": "PETR4",
    "quotes": [
        15.5,
        15.1,
        10.5,
        18.6
    ]
}
```

There will be 5 operations expected for your application and all of them are described below:

Create Stock

This operation will create a new Stock entry in database. For create operations only the name is a mandatory parameter, Quotes are optional. Below there is an example of a create operation:

URL: http://<host>:<port>/stock HTTP Method: POST

Request Payload:

```
{
    "name": "PETR4",
    "quotes": [
        15.5,
        15.1,
        10.5,
        18.6
    ]
}
```

Response Payload:

```
{
    "name": "PETR4",
    "quotes": [
        15.5,
        15.1,
        10.5,
        18.6
    ]
}
```

Update Stock

This operation will update an existing Stock entry in database. For update operations only the Quotes are a mandatory parameter in the request payload, the Stock name will be used as a path parameter. This update operation must be used to introduce new Quote values in the Stock. Below there is an example of an update operation:

URL: http://<host>:<port>/stock/<stock_name>   HTTP Method: PATCH

Request Payload:

```
{
    "quotes": [
        9.6
    ]
}
```

Response Payload:

```
{
    "name": "PETR4",
    "quotes": [
        15.5,
        15.1,
        10.5,
        18.6
        9.6
    ]
}
```

Read Stock

This operation will read existing Stocks in database. For read operations there will be two different responses, the first one must return all existing Stocks in database and the second one must find one Stock by name. Below there are examples for both scenarios:

Read All Stock

URL: http://<host>:<port>/stock   HTTP Method: GET

Request Payload: N/A

Response Payload:

```
[
    {
        "name": "PETR3",
        "quotes": [
            2.3,
            6.1,
            9.6
        ]
    },
    {
        "name": "PETR4",
        "quotes": [
            15.5,
            15.1,
            10.5,
            18.6
            9.6
        ]
    }
]
```

Read Stock by Name

URL: http://<host>:<port>/stock?name=<stock_name>   HTTP Method: GET

Request Payload: N/A

Response Payload:

```
{
    "name": "PETR4",
    "quotes": [
        15.5,
        15.1,
        10.5,
        18.6
        9.6
    ]
}
```

Delete Stock

This operation will delete an existing Stock entry in database. For delete operations only the Quotes the Stock name will be used as a path parameter (mandatory). Below there is an example of a delete operation:

URL: http://<host>:<port>/stock/<stock_name> HTTP Method: DELETE

Request Payload: N/A

Response Payload: N/A