In this demo there are two apis / modules

**1.SensorDeviceApplication**

This api exposes methods to send command to sensor device to swith on or off depending on data submitted by IrrigationAutomationApplication.We have mocked data to get response from this service.

**2.IrrigationAutomationApplication**

This api has methods to do CRUD operations for Plot and Config entity.

It has below packages

1.com.digitalfactory.controller

This package has below controller classes

PlotController

This has methods to create,get,delete and update plot details.

ConfigController

This has methods to create,get,delete and update plot details.

NotificationController

This has methods to send notification and subscribe for notifications using AWS SNS service.

2.com.digitalfactory.config

This has only one class AwsConfig to set configuration details for calling AWS SNS service.

3.com.digitalfactory.exception

This has only one class SensorDeviceServiceNotAvailableException to handle exception when Sensor Device Api is down.

4.com.digitalfactory.model

It has model classes like Plot,Config and Notification.

5.com.digitalfactory. repository

It has classes to interact with database.

6. com.digitalfactory. retry

It has services for retry logic when Sensor Device service is down.

7. com.digitalfactory. taskconfig

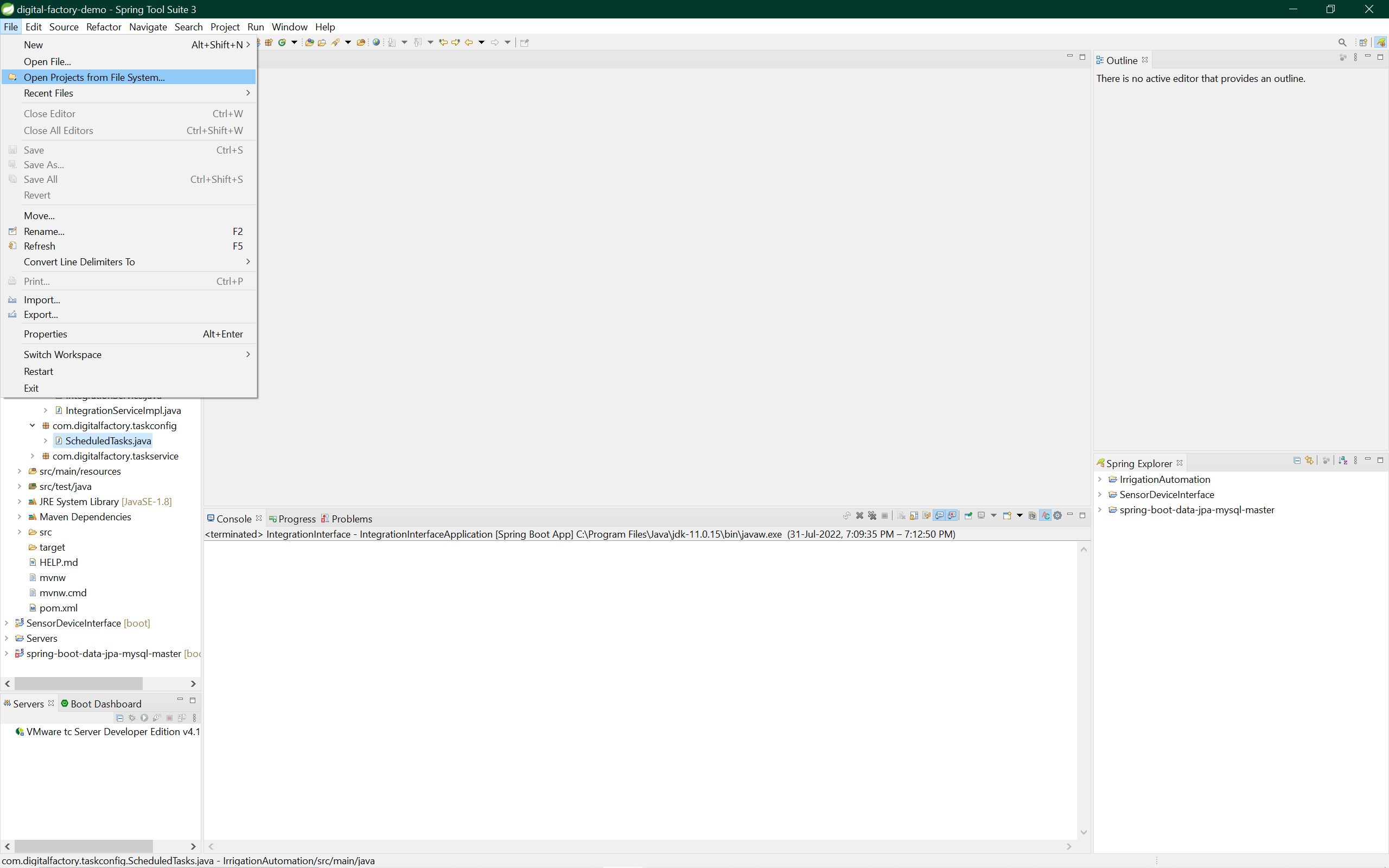
It has a class ScheduledTasks which will execute commands to switch on and switch off sensor device by fetching eligible plot details from database after certain interval of time.

8. com.digitalfactory. taskservice

It has services to fetch records from database and to call sensor device api.

***SETTING UP APPLICATION***

Unzip the zip file on your local drive and import both projects in SpringBoot.



SCENARIO 1

TESTING THE APPLICATION FOR RETRY LOGIC AND NOTIFICATION AFTER RETRY ATTEMPTS ARE OVER

To test retry logic we will keep SensorDevice api down and try to consume its services from other apis in IrrigationAutomationApplication.

First create database “irrigationdb” in MySQL because when we start the application it will connect to the database and create tables plots and config .

After that start IrrigationAutomationApplication

Graphical user interface, application

Description automatically generated

When application start ScheduleTask will run and fetch data from plots table after regular interval and if there is any data available for irrigation as per the start time and end time it will submit the plot details to sensor device api for irrigation.Initially there will be no records so api call will not happen.So first we will call plots api to create records.

Open Postman and send post request to create plot details for automatic irrigation.

POST URL : <http://localhost:8080/api/plots>

SAMPLE DATA

{

"name" : "A144",

"description" : "Irrigation for Sugarcane",

"area" : "1200",

"cropType" : "SugarCane",

"soilType" : "Normal",

"startTime" : "05:22:00",

"endTime" : "05:23:00"

}

For testing keep startTime 1 minute ahead of current time and also endTime 1 minute ahead of current time so that Scheduler will pick records and call sensor device to switch on / off .

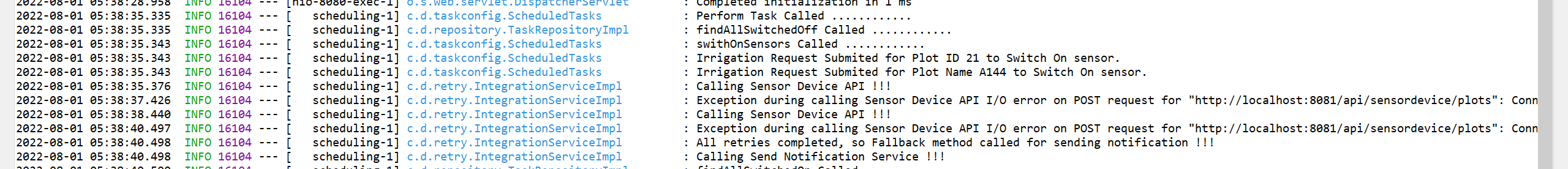
For example if current time is 05:21:00

Then in request put startTime equal to 05:22:00and endTime as 05:23:00.

Once scheduler pick data matching startTime it will call sensor device api to submit irrigation request to process.

In our case as SensorDevice api is down so retry logic will execute and after retry attempts are over , fall back method will send notification / alert.

Refer below log showing retry attempts and notification execution



SCENARIO 2

TESTING THE APPLICATION WHEN SENSOR DEVICE API IS UP

Now start the Sensor Device Api

Graphical user interface

Description automatically generated

Now both apis are up and we can test the api call to switch on and off sensor device.

Again submit request using Postman by sending nearest startTime and endTime of currenttime to execute commands for switching on and switching off sensor device.

If current time is “05:21:00” then sample request data will be

{

"name" : "A144",

"description" : "Irrigation for Sugarcane",

"area" : "1200",

"cropType" : "SugarCane",

"soilType" : "Normal",

"startTime" : "05:22:00",

"endTime" : "05:23:00"

}

Graphical user interface, text, application, email

Description automatically generated

Console Log

Text

Description automatically generated

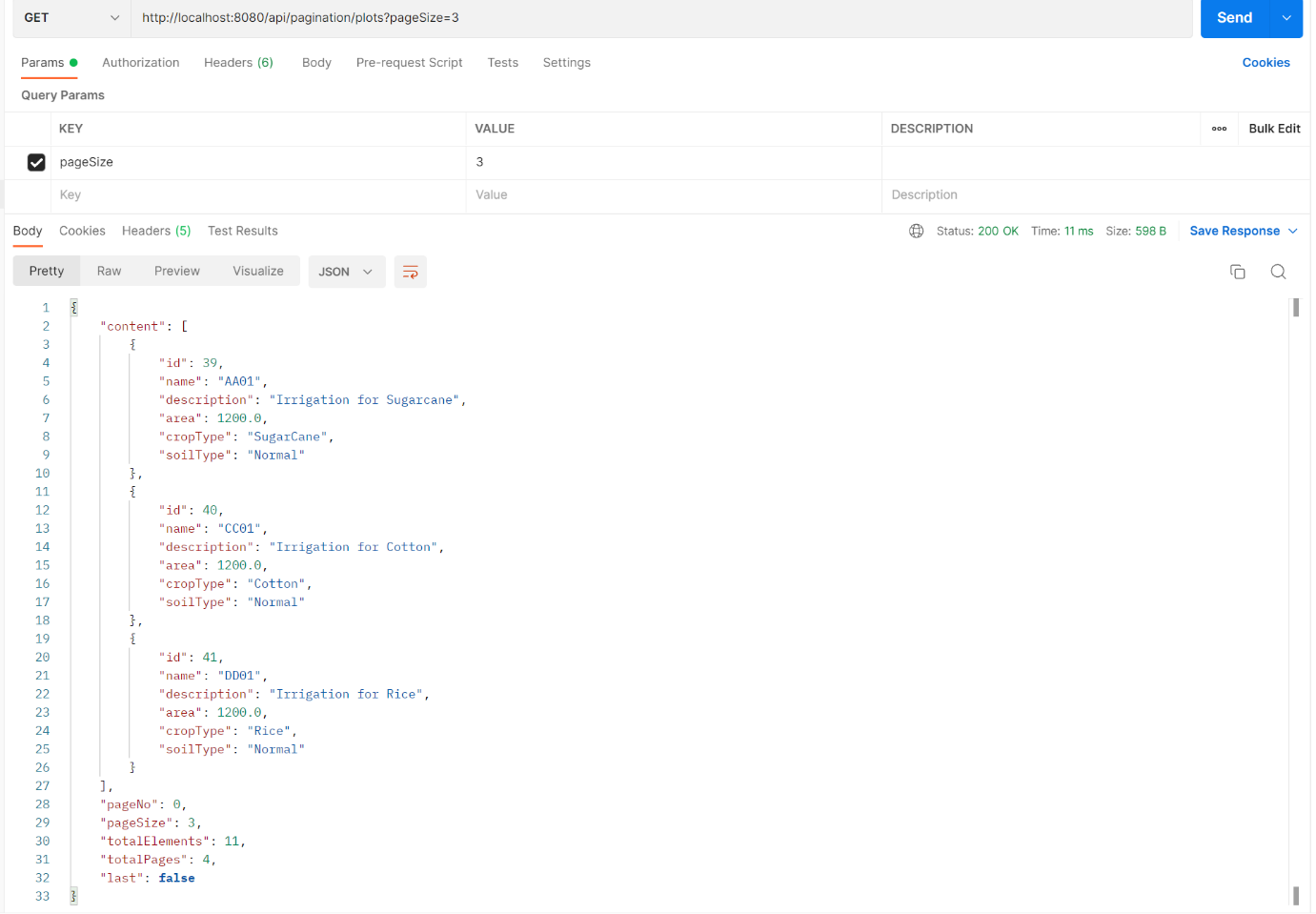
Console shows sensor is switched on by calling remote sensor device api when starttime is crossed and switched off when endtime is crossed.

Sensor Mode is also updated in the database after successful request execution.

Pagination Testing

Request - GET

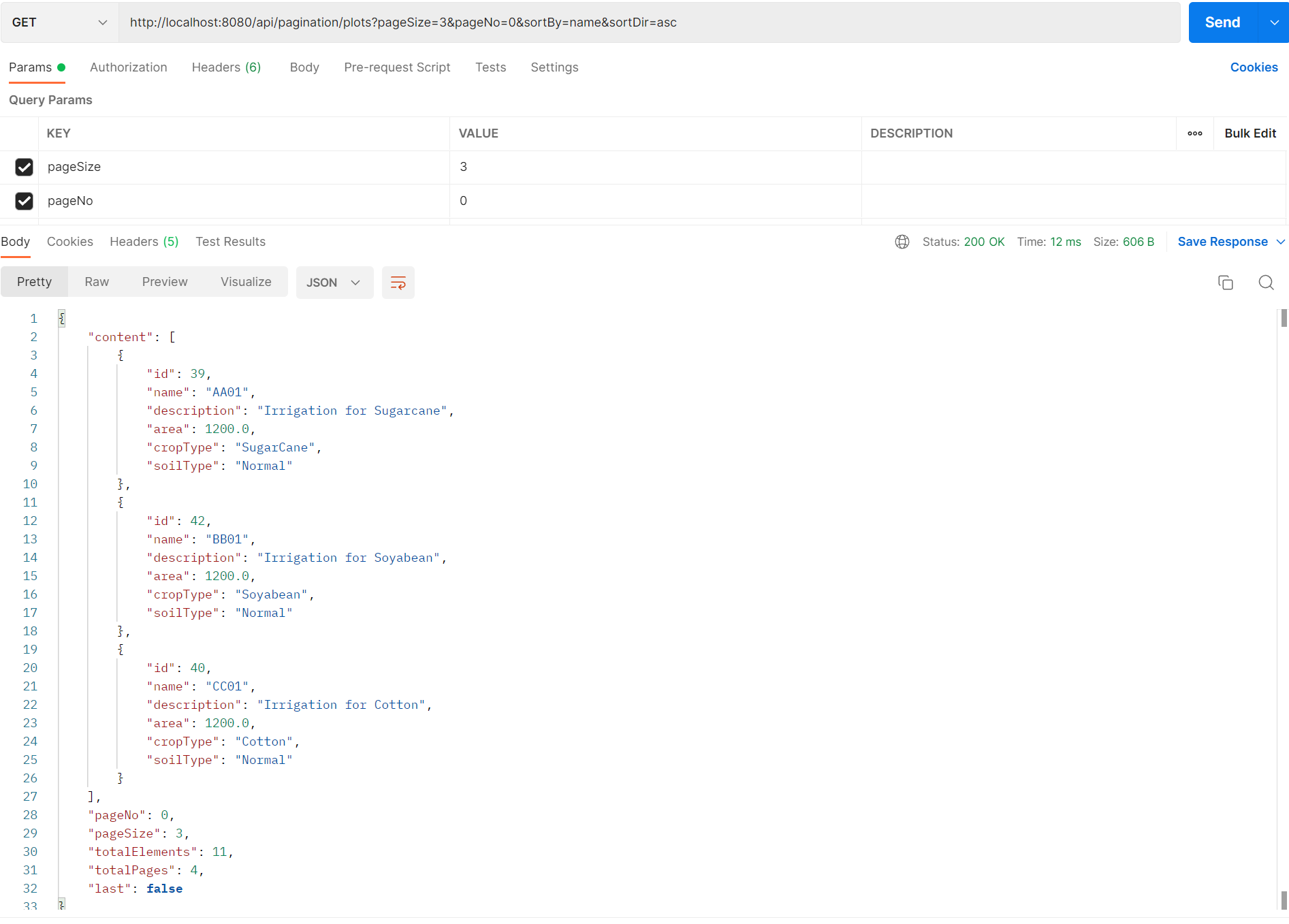
URL - http://localhost:8080/api/pagination/plots?pageSize=3



Sorting

Request - GET

URL - http://localhost:8080/api/pagination/plots?pageSize=3&pageNo=0&sortBy=name&sortDir=asc



Records are sorted by name.

Other CRUD Operations

1.No records found

Request - GET

URL - http://localhost:8080/api/plots/100

Graphical user interface, text, application, email

Description automatically generated

2.Create new Plot

POST <http://localhost:8080/api/plots>

Request Body

{

"name" : "MM01",

"description" : "Irrigation for Tea",

"area" : "1200",

"cropType" : "Tea",

"soilType" : "Normal",

"startTime" : "05:22:00",

"endTime" : "05:23:00"

}

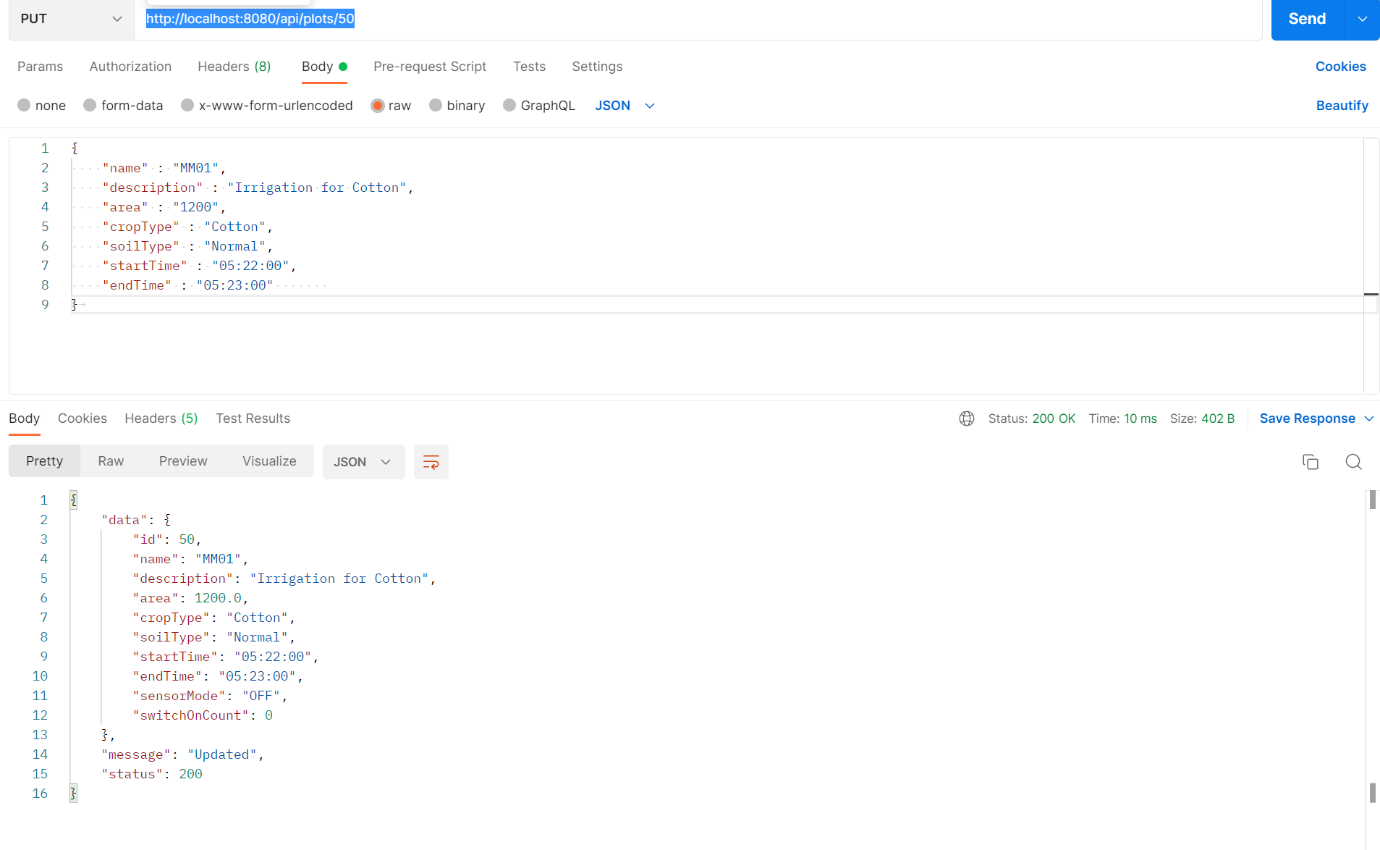
Graphical user interface, text, application, email

Description automatically generated

3.Update Plot Details

Request Type - PUT

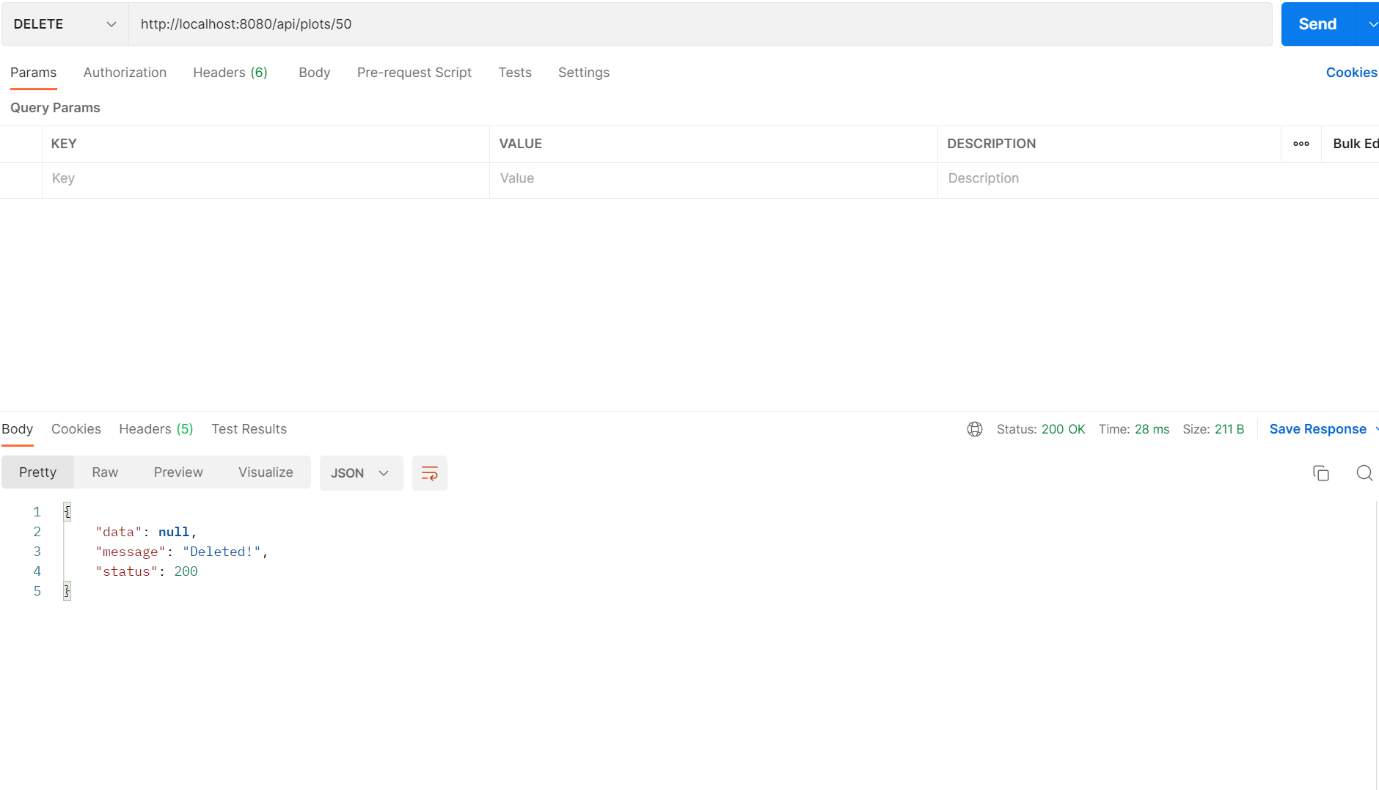
URL - http://localhost:8080/api/plots/50



4.Delete details by ID

Request Type - DELETE

URL - http://localhost:8080/api/plots/50



Request Validation

name,cropType and soilType are mandatory

URL : <http://localhost:8080/api/plots>

Body

{

"area" : "1200",

"startTime" : "15:49:00",

"endTime" : "15:50:00"

}

Graphical user interface, text, application, email

Description automatically generated

Config API

1.Create new Config

Request Type - POST

URL - http://localhost:8080/api/configs

Payload

{

"plotArea" : "1200",

"cropType" : "Rice",

"soilType" : "Dry",

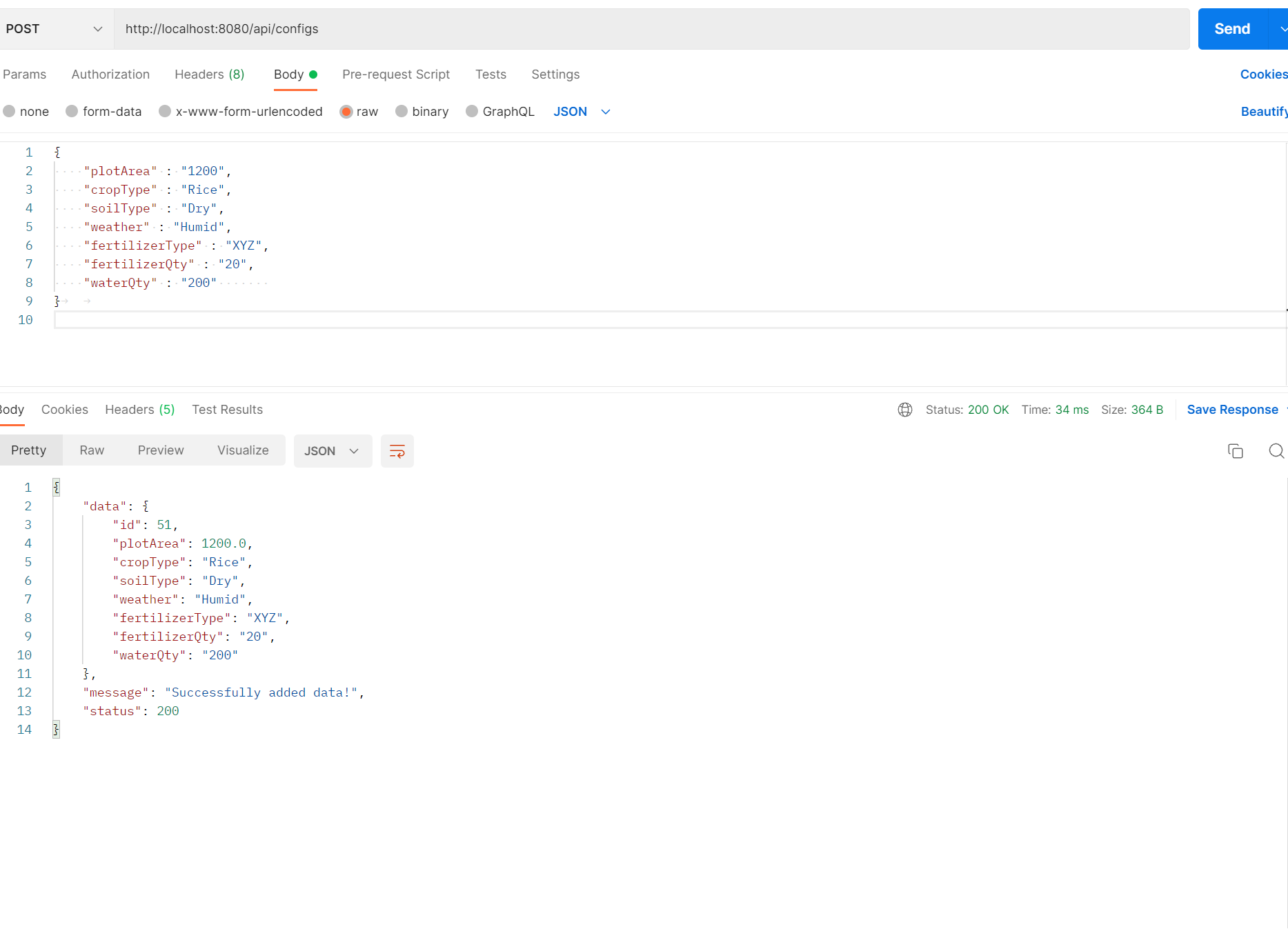
"weather" : "Humid",

"fertilizerType" : "XYZ",

"fertilizerQty" : "20",

"waterQty" : "200"

}



2.Update Config

Request Type - PUT

URL - http://localhost:8080/api/configs/51

Payload

{

"plotArea" : "1200",

"cropType" : "Rice",

"soilType" : "Dry",

"weather" : "Humid",

"fertilizerType" : "XYZ",

"fertilizerQty" : "30",

"waterQty" : "300"

}

