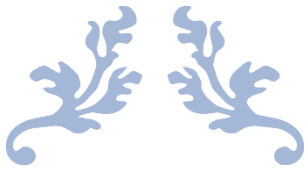


SISTEMA DE RIEGO AUTOMÁTICO

Desarrollo de Aplicaciones Distribuidas

ÁLVARO GARCÍA RODRÍGUEZ
MARCOS STEPHAN PERALVO GERMAN

Contenido

1.- PRIMERA ITERACIÓN

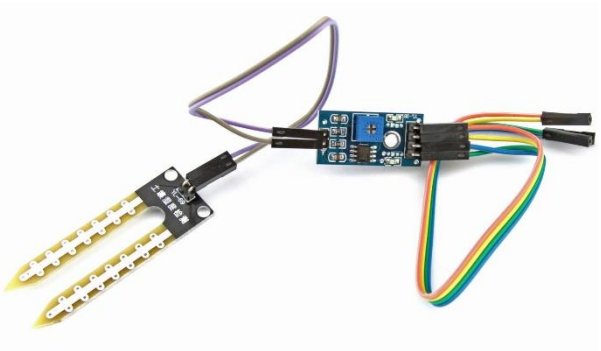
1.1.- Introducción

El proyecto por realizar consistirá en un sistema de riego automático.

Para la realización de este, se estima que se utilizará una placa WiFi NodeMCU ESP8266, un sensor de humedad, una mini bomba de agua, un relé para controlar la potencia de la bomba, unas resistencias y tres diodos LEDs para informar al usuario del estado de la tierra a regar: amarillo si la tierra está lista para ser regada, verde si la tierra está húmeda, rojo y parpadeando, no está el sensor en la tierra.

A lo largo de la realización del proyecto, se verá si implementar alguna variación de este y/o algunas recomendaciones del profesor.

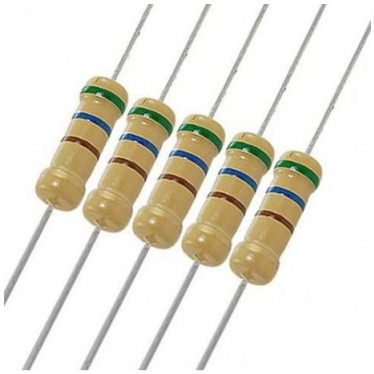
NodeMCU ESP8266



Sensor de humedad Arduino

Mini bomba de agua

Diodos LEDs

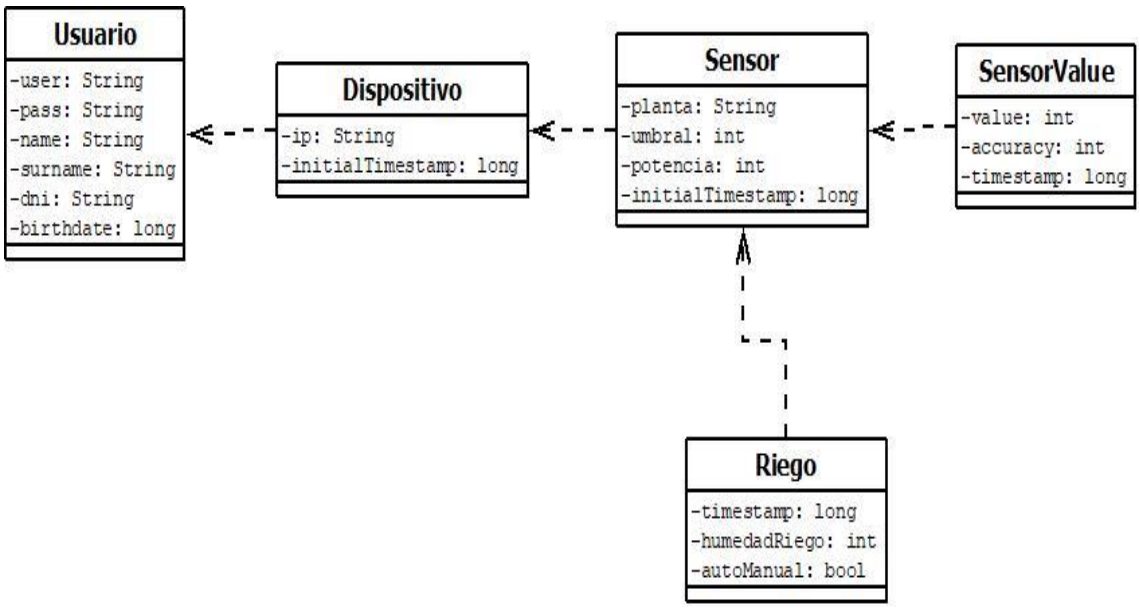


Resistencias

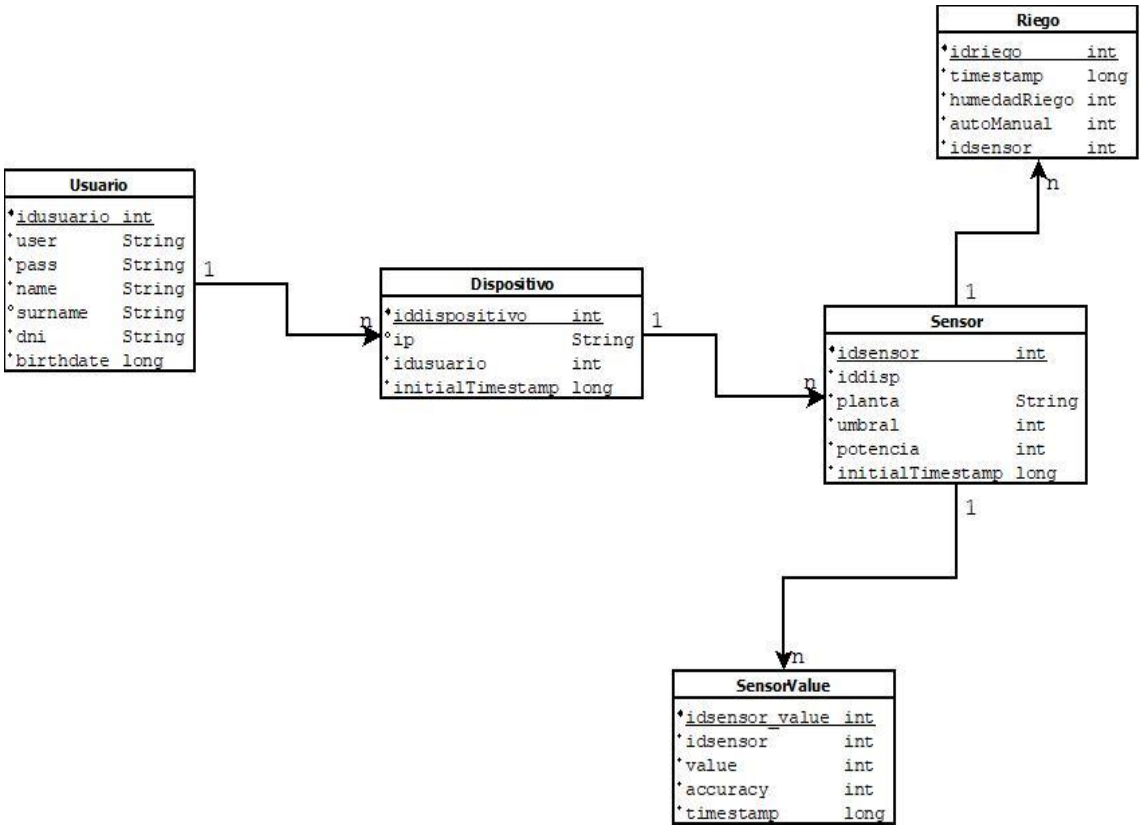
Relé 3,3 V

2.- SEGUNDA ITERACIÓN (BBDD)

2.1.- UML



2.2.- ESQUEMA E/R



3.- TERCERA ITERACIÓN (API Rest)

3.1.- POST

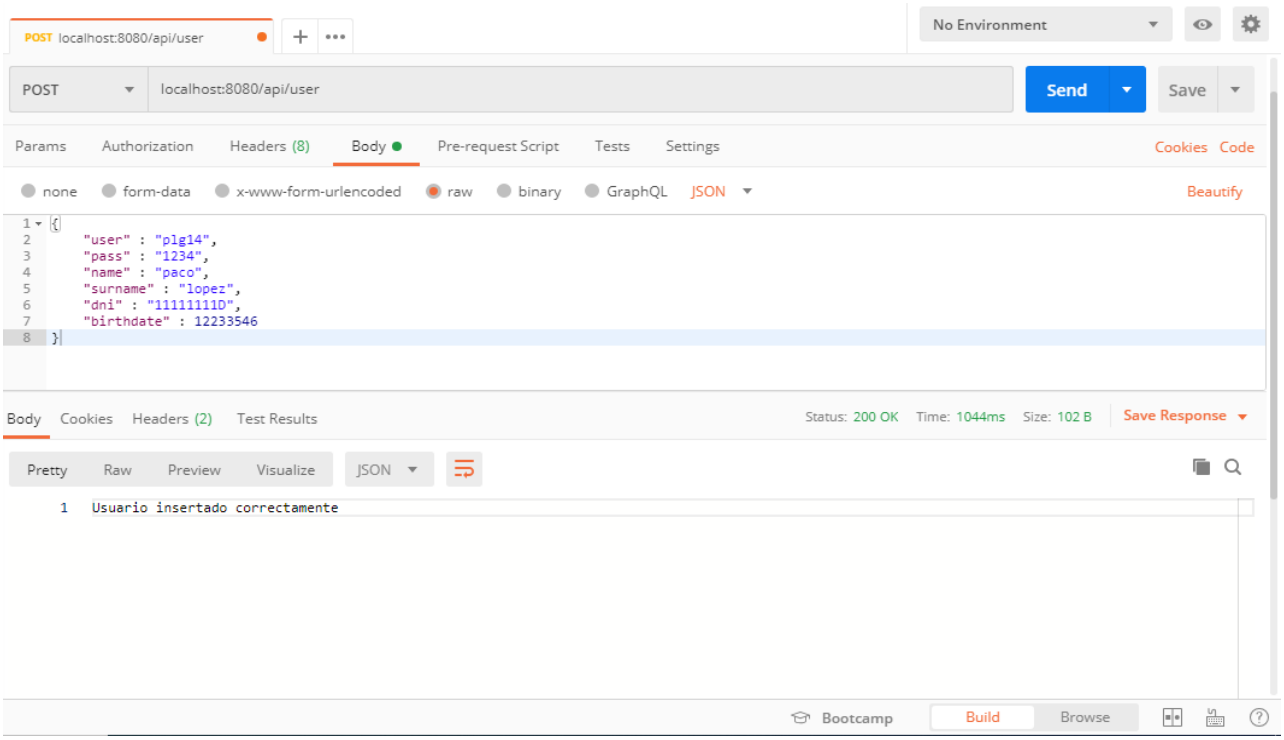
Usaremos los POST para las inserciones en la base de datos.

- "/api/user"

  Inserción en la base de datos de un nuevo usuario (registro). No se le pasa ningún parámetro.
  El cuerpo es el siguiente (ejemplo):

{
"user" : "plg14",
"pass" : "1234",
"name" : "paco",
"surname" : "lopez",
"dni" : "11111111D",
"birthdate" : 12233546
}

En cuanto se inserte en la base de datos, una respuesta al cliente sería que el registro se ha completado satisfactoriamente.



- "/api/device"

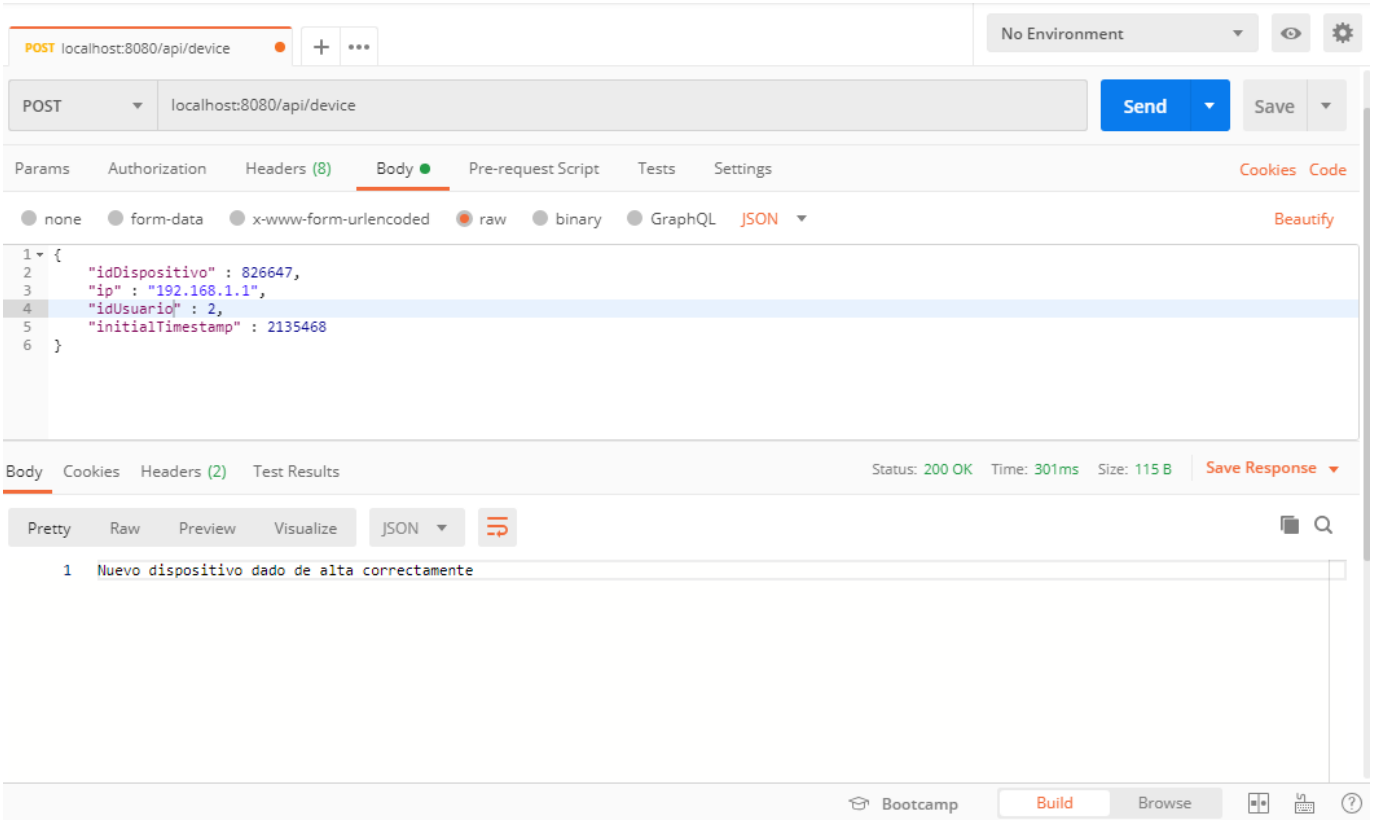
Se llamará a esta función cuando un usuario quiere dar de alta a un nuevo dispositivo (ESP8266).

El cuerpo es el siguiente:

{
"idDispositivo" : 826647,
"ip" : "192.168.1.1",
"idUsuario" : 2,
"initialTimestamp" : 2135468
}

El id se pondrá de forma manual para identificar siempre el dispositivo físico.

El mensaje al usuario sería que el dispositivo se ha dado de alta correctamente.



- "/api/device/sensor"

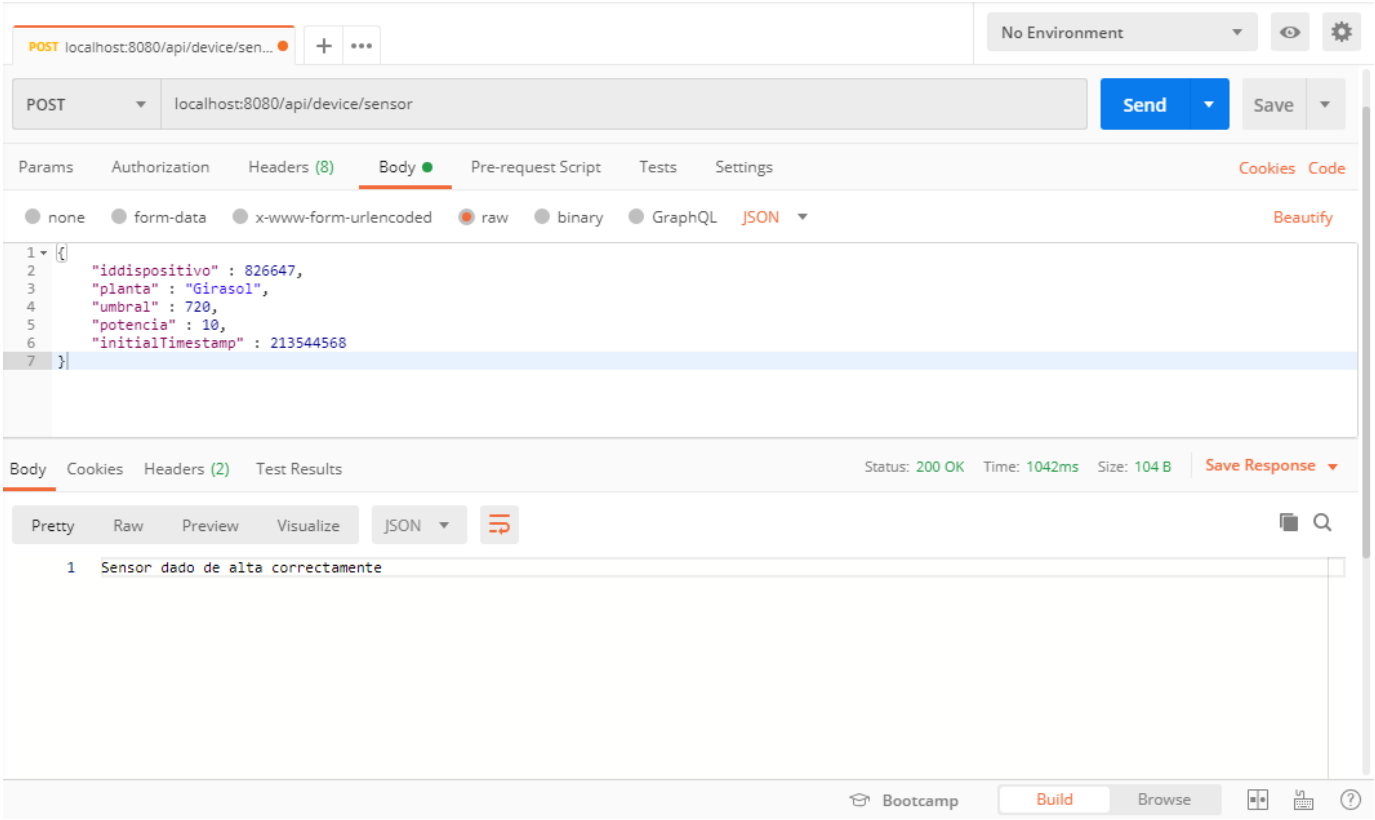
  Cuando el usuario conecte un sensor al dispositivo, dicho usuario dará de alta al sensor.

  El cuerpo es el siguiente:

  {
  "iddispositivo" : 826647,
  "planta" : "Girasol",
  "umbral" : 720,
  "potencia" : 10,
  "initialTimestamp" : 213544568
  }

  Se tiene que especificar a qué planta estará conectado el sensor, umbral que de llegar el sensor activará la bomba de agua, y la potencia por la que funcionará la bomba.

  La respuesta al cliente será que el sensor se ha dado de alta correctamente.



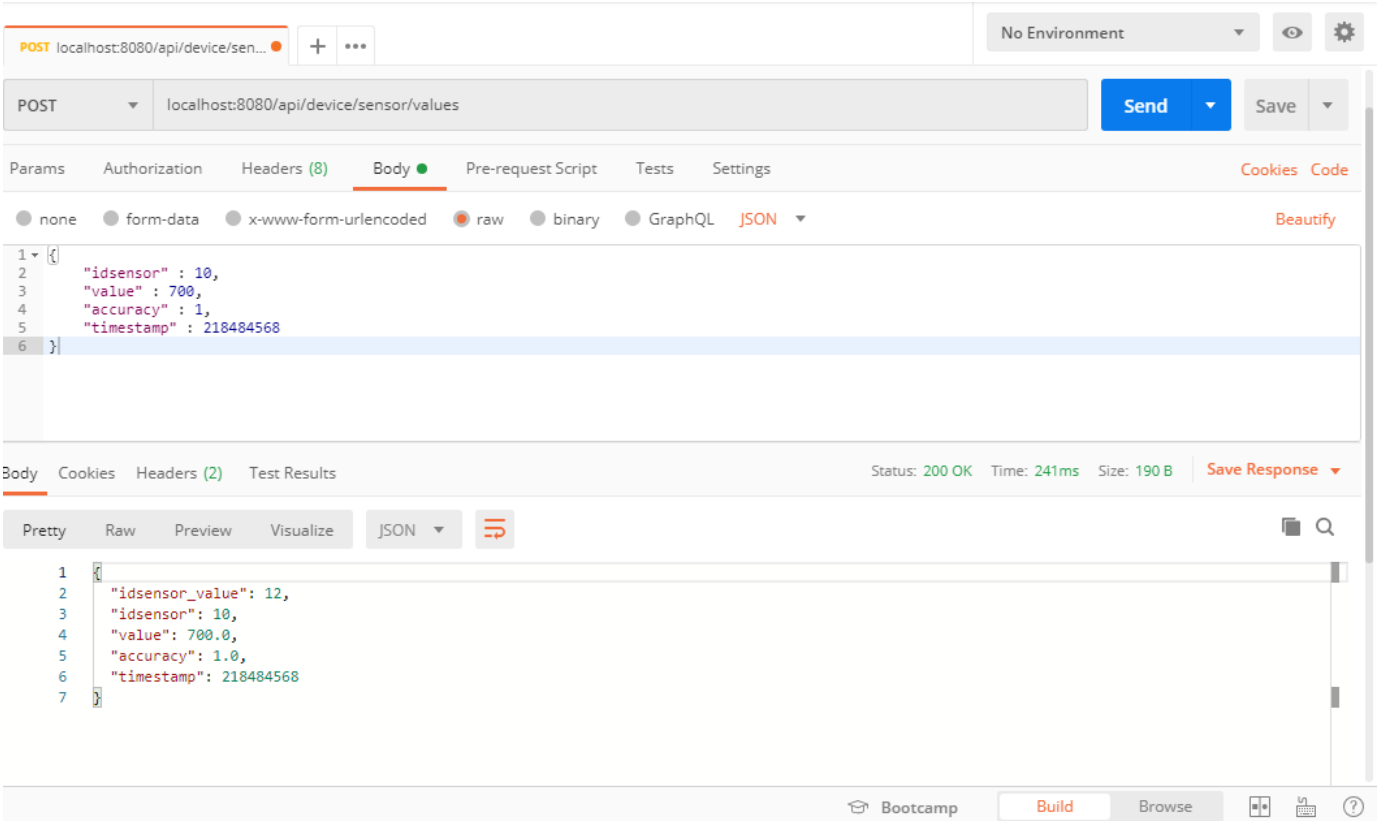
- "/api/device/sensor/values"

  Este método se llamará cuando el sensor haga lecturas de la humedad.

  El cuerpo es el siguiente:

  {
  "idsensor" : 10,
  "value" : 700,
  "accuracy" : 1,
  "timestamp" : 218484568
  }

  No habrá respuesta al usuario.



- "/api/device/sensor/riego"

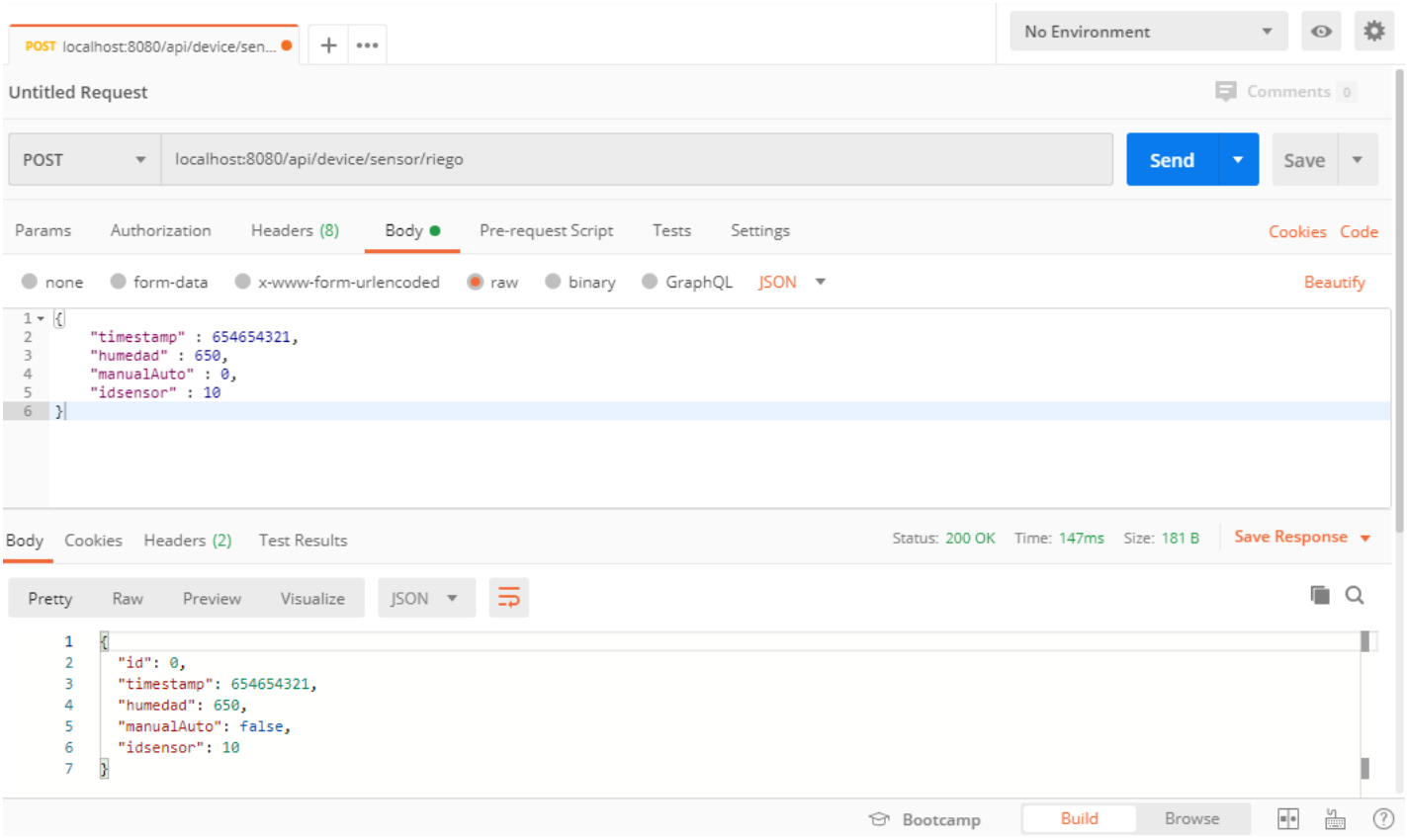
  Este método se llamará cada vez que se realice la acción de riego, para guardar un historial de los riegos.

  El cuerpo es el siguiente:

  {
  "timestamp" : 654654321,
  "humedad" : 650,
  "manualAuto" : 0,
  "idsensor" : 10
  }

  El campo humedad es la humedad registrada cuando se regó y el campo manualAuto es 0 si es un riego manual o 1 si ha sido automático.

  No habrá respuesta al usuario.



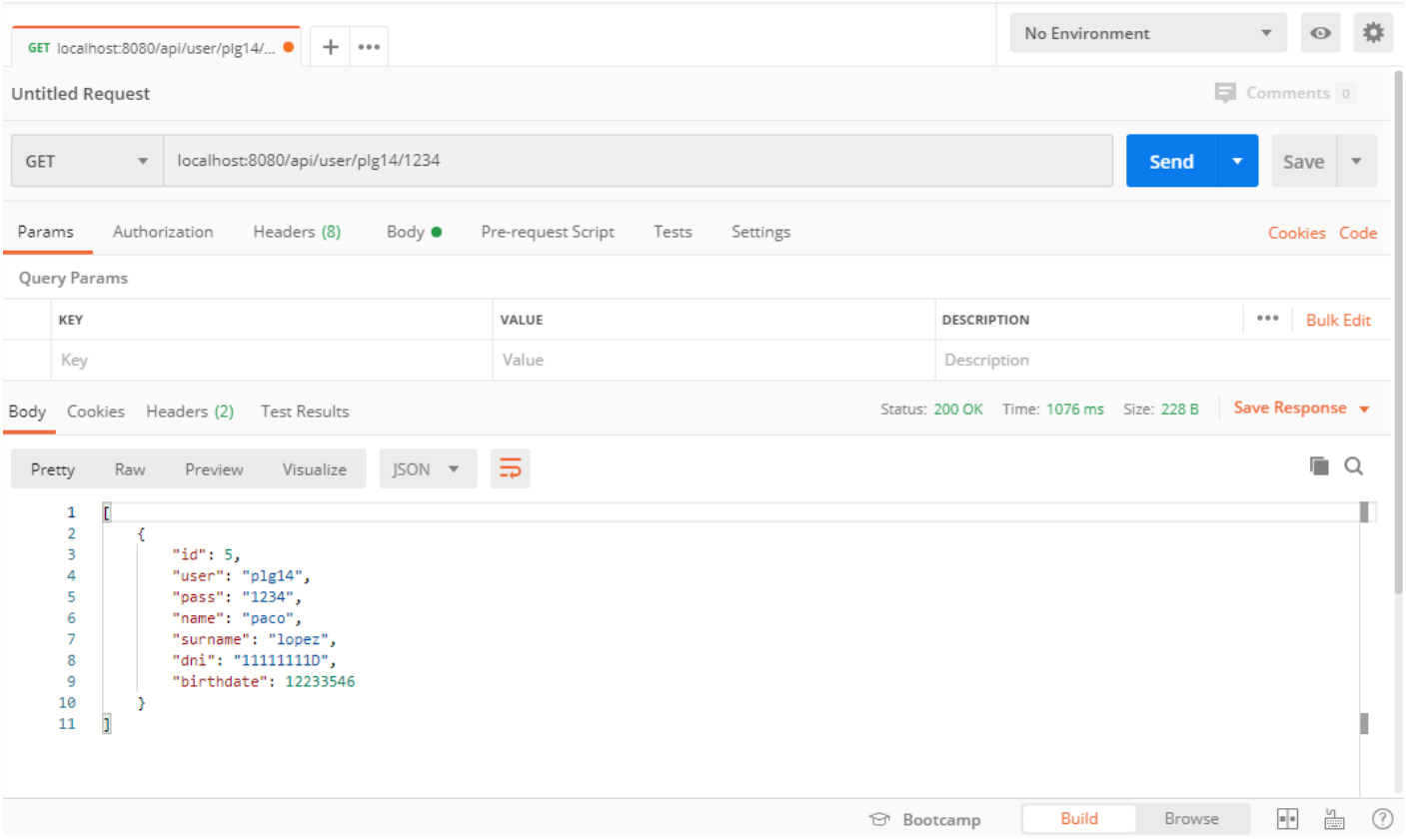
3.2.- GET

Usaremos los GET para hacer SELECT a la BBDD.

- "/api/user/:user/:pass"

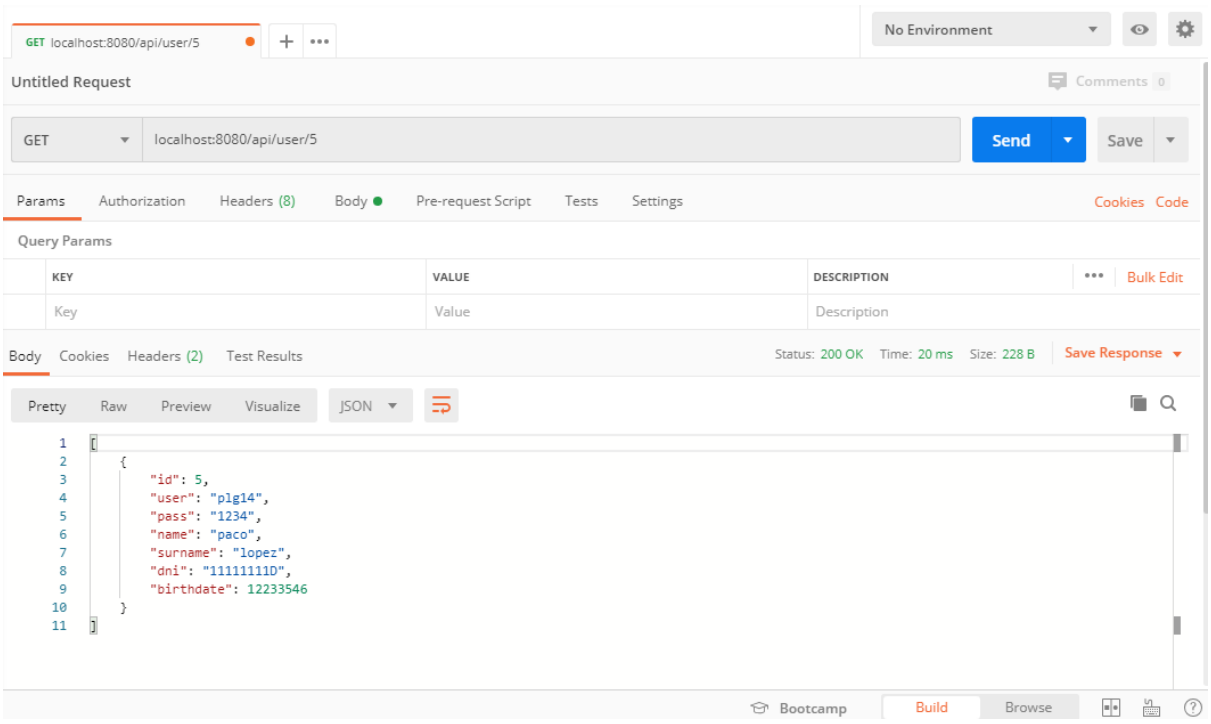
  Esta URL se usará para el login del usuario. Se pedirá el usuario y la contraseña, rellenada en el formulario de login. Si es correcto, se accederá a la aplicación, si es incorrecto, se enviará un mensaje de error.

  Como es un GET, no hay cuerpo.



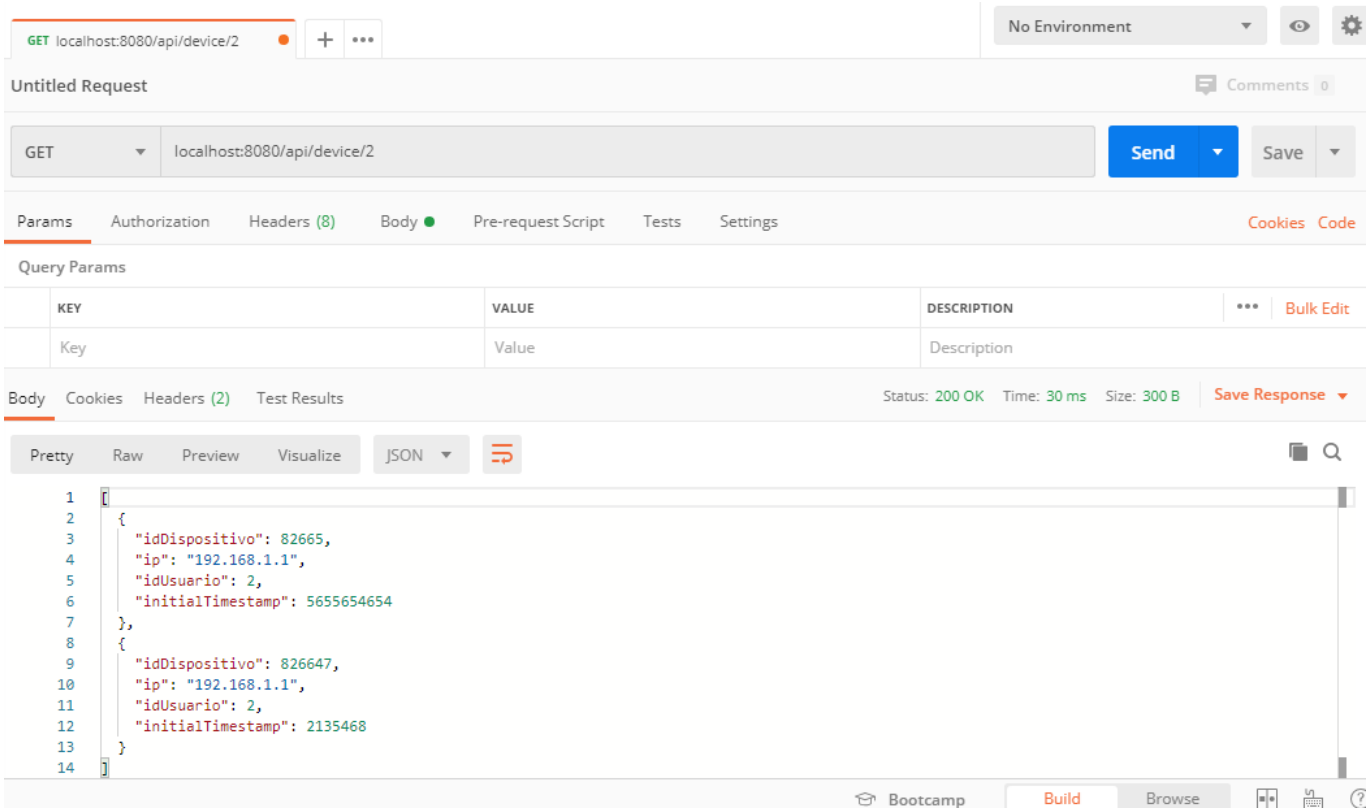
- "/api/user/:idusuario"

  Dado el id de un usuario, carga todos los datos de dicho usuario. Usado para cargar las cajas de texto de edición de usuario. Como es un GET, no hay cuerpo.



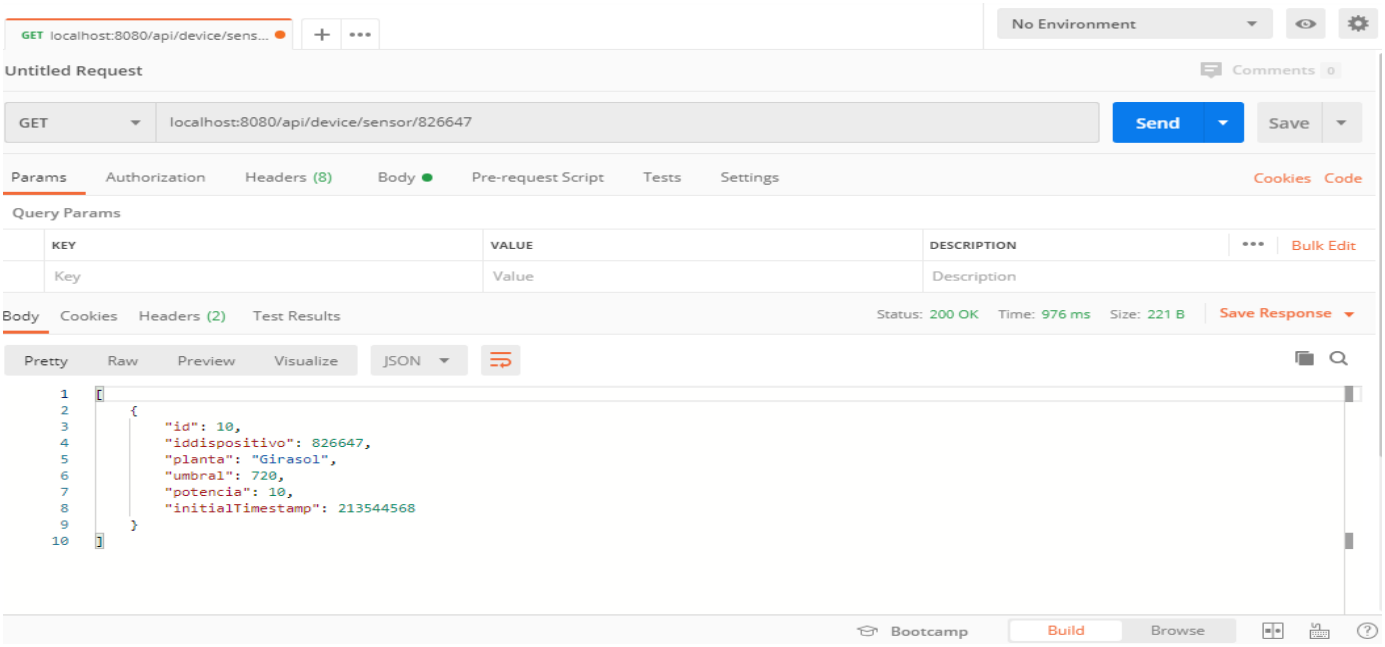
- "/api/device/:idusuario"

  Dado el id del usuario, permite visualizar una lista de dispositivos que tiene dicho usuario dado de alta. Como es un GET, no tiene cuerpo.



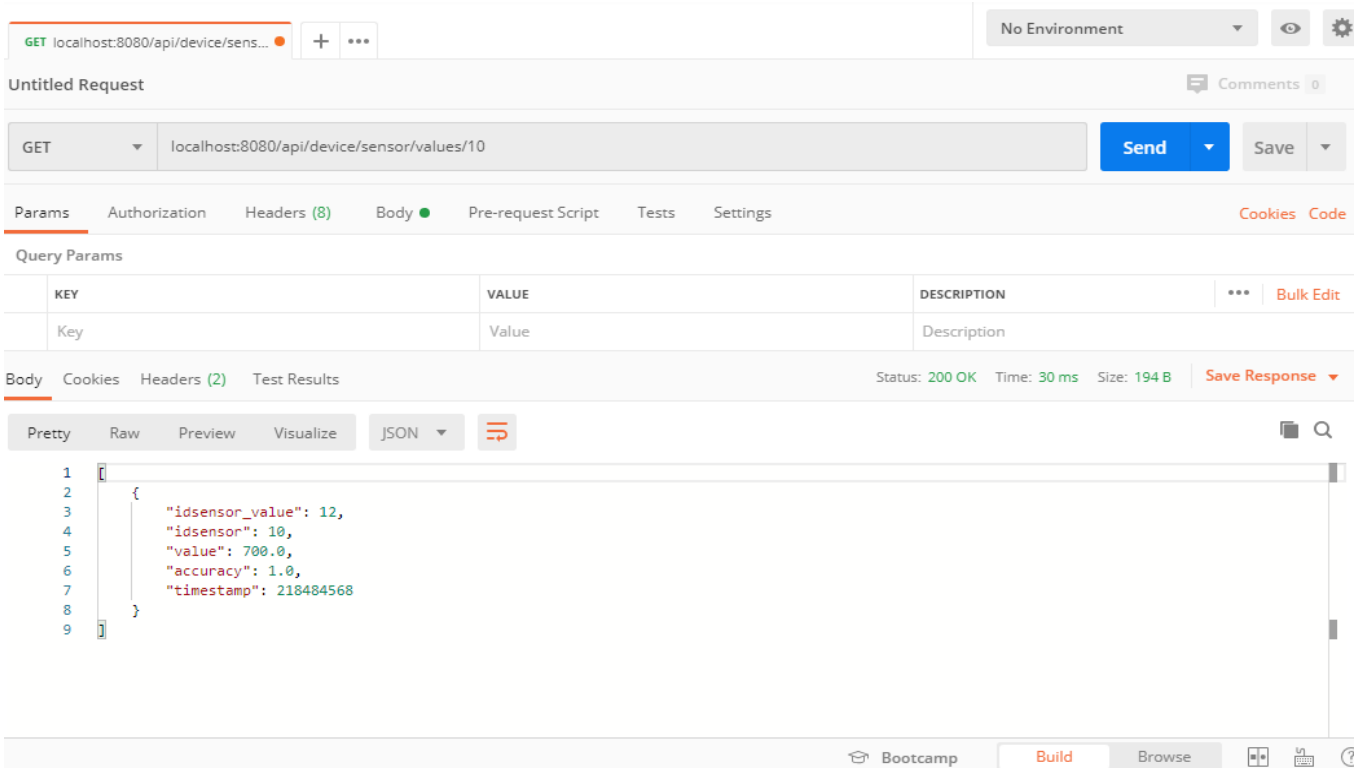
- "/api/device/sensor/:idsensor"

  Dado el id del dispositivo, carga una lista de sensores vinculados a dicho dispositivo. Como es un GET, no tiene cuerpo.



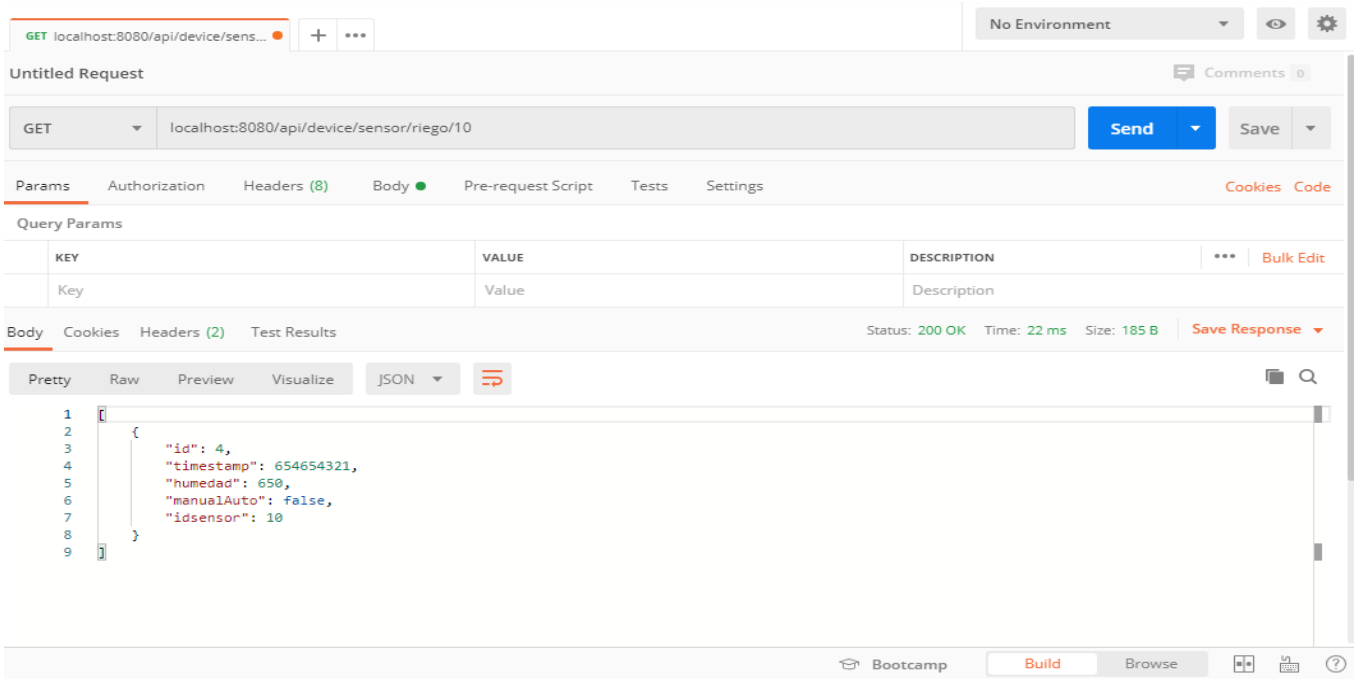
- "/api/device/sensor/values/:idSensor"

  Dado el id del sensor, mostrará una tabla con todos los valores leídos por dicho sensor. Como es un GET, no tiene cuerpo.



- "/api/device/sensor/riego/:idsensor"

  Dado el id del sensor, mostrará un registro de los riegos que se han realizado dicho sensor. Como es un GET, no lleva cuerpo.



3.3.- PUT

Usaremos los PUT para hacer UPDATE en la BBDD.

- "/api/user/:idusuario"

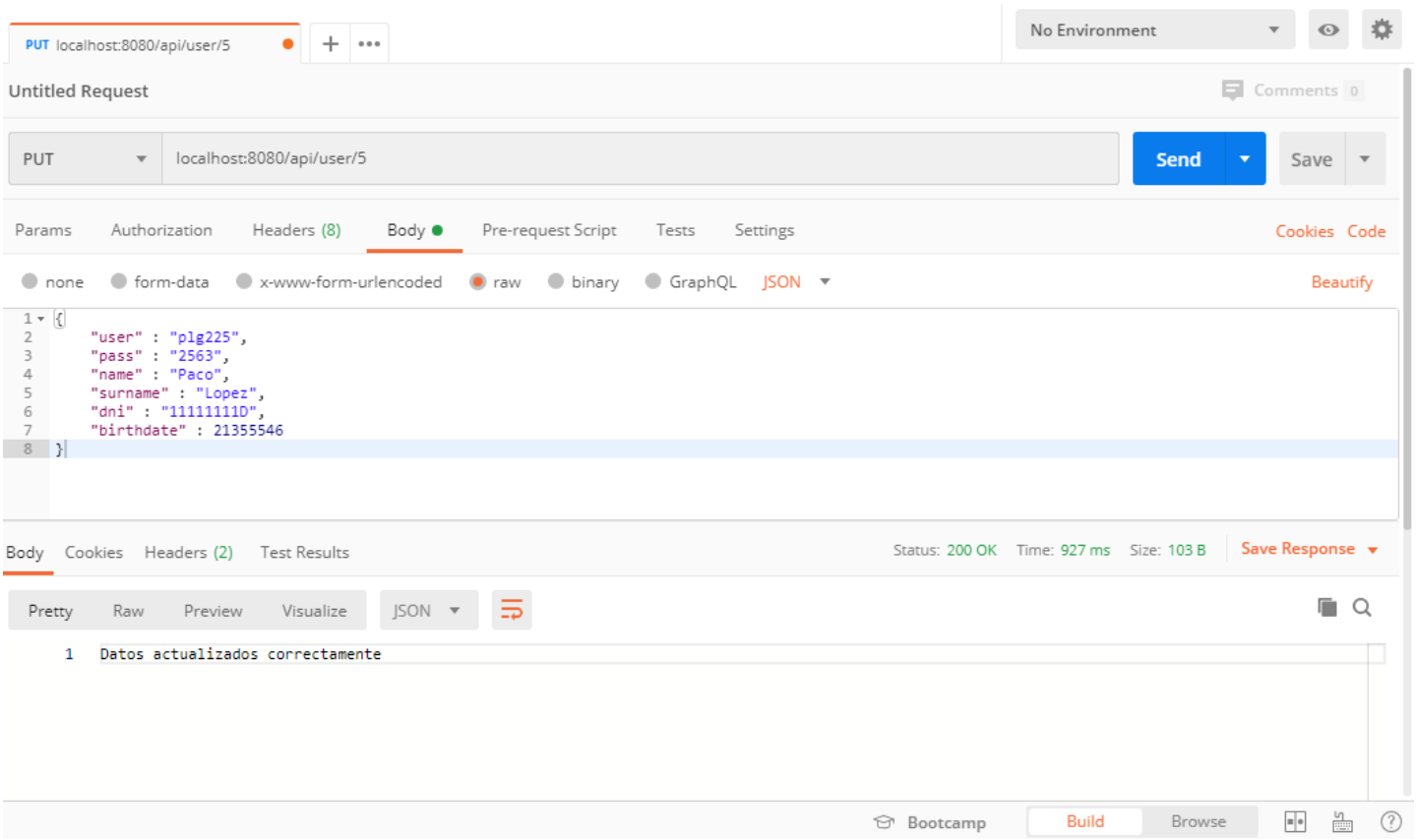
  Dado el id del usuario, se actualizarán los datos de dicho usuario con los datos rellenados en un formulario.

  El cuerpo es el siguiente:

  ```
  {
        "user" : "plg225",
        "pass" : "2563",
        "name" : "Paco",
        "surname" : "Lopez",
        "dni" : "11111111D",
        "birthdate" : 21355546
  }
  ```

  Como se puede ver, se permite cambiar cualquier dato del usuario.

  Una respuesta al cliente sería que los datos del usuario se han actualizado correctamente.



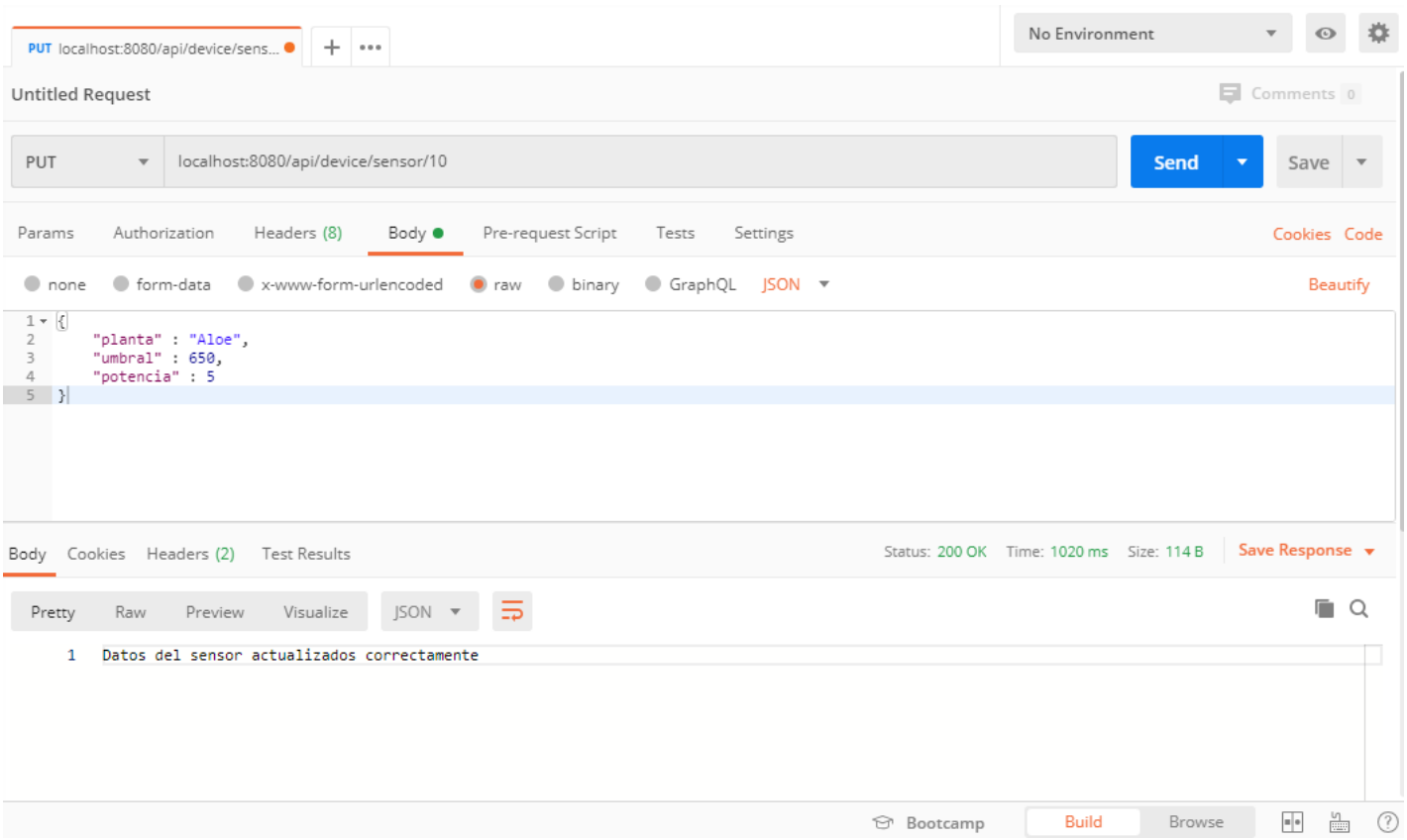
- "/api/device/sensor/:idsensor"

  Dado el id del sensor, se actualizarán los datos del sensor, por si cambias de planta por ejemplo.

  El cuerpo es el siguiente:

  ```
  {
          "planta" : "Aloe",
          "umbral" : 650,
          "potencia" : 5
  }
  ```

  Una respuesta al cliente sería que los datos han sido actualizados correctamente.



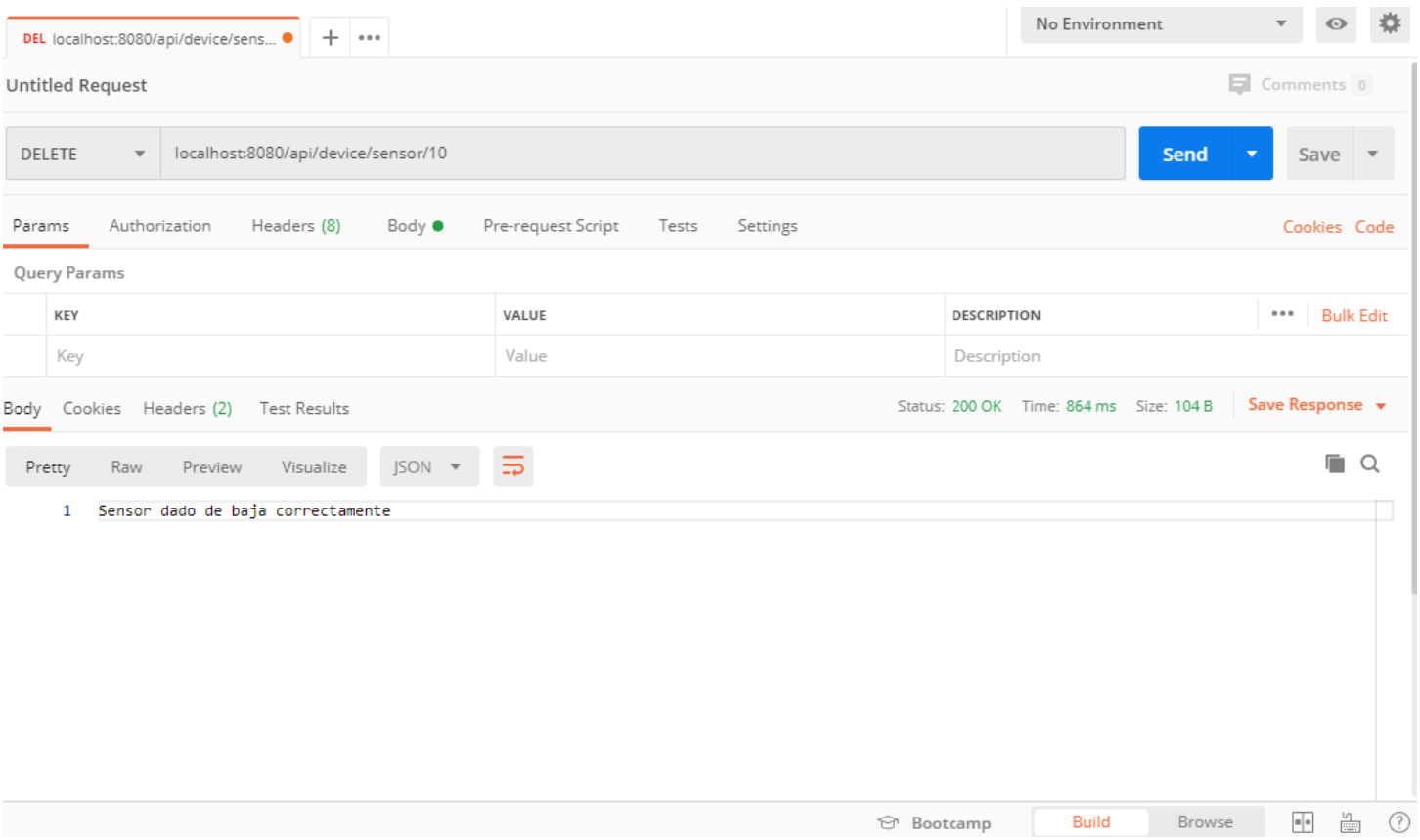
3.4.- DELETE

Usaremos los métodos DELETE para hacer DELETE en la BBDD.

- "/api/device/sensor/:idsensor"

  Dado el id del sensor, se borrará todo dato de dicho sensor. Como está configurado para un borrado en CASCADE, las tablas de los valores del sensor y de los riegos se borrarán.

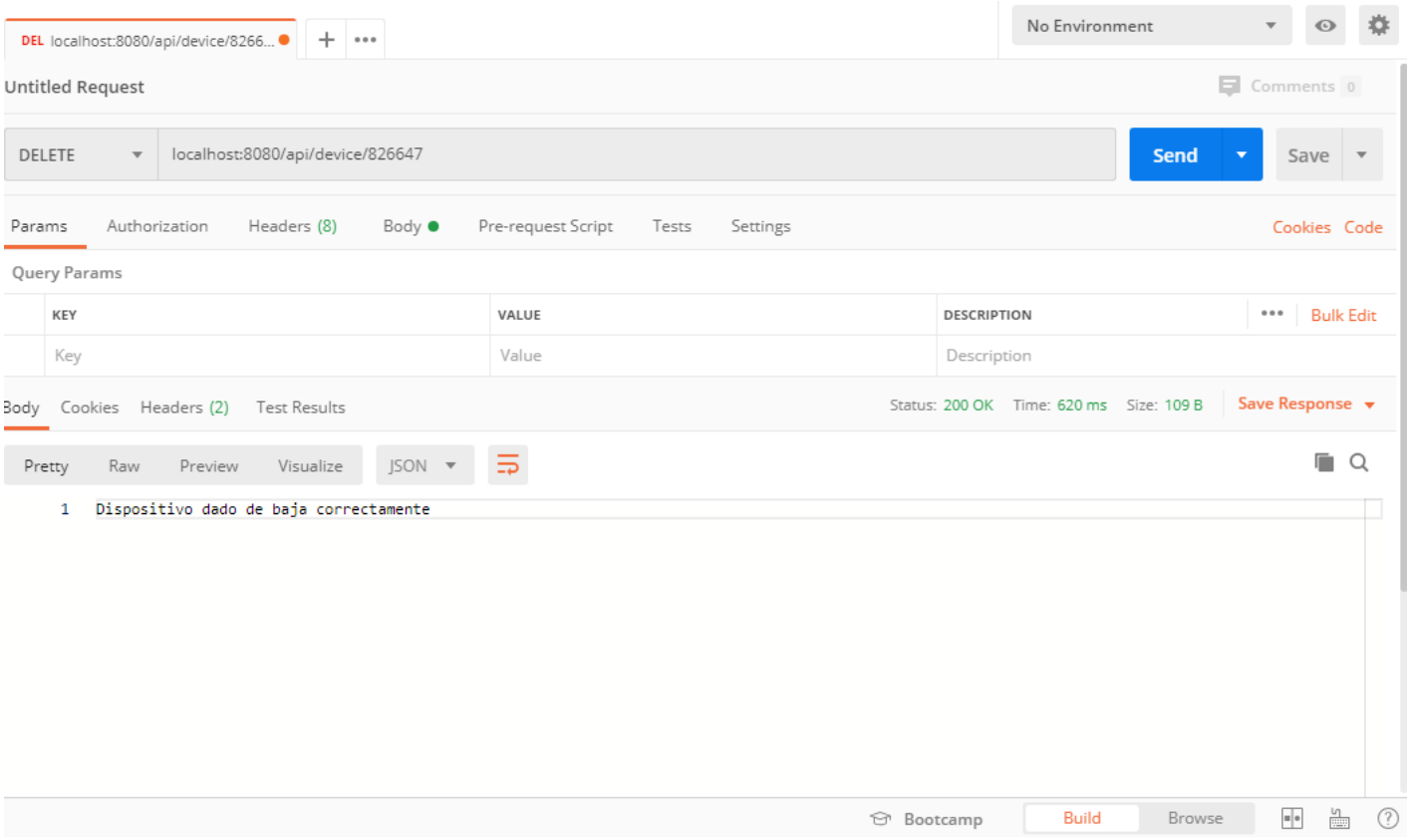
  Como sólo nos interesa el id, no habrá cuerpo. Un mensaje al cliente sería que el sensor se ha dado de baja correctamente.



- "/api/device/:iddispositivo"

  Dado el id del dispositivo, se borrará todo dato de dicho dispositivo. Como está configurado para un borrado en CASCADE, los datos de todos los sensores conectados al dispositivo, se borrarán.

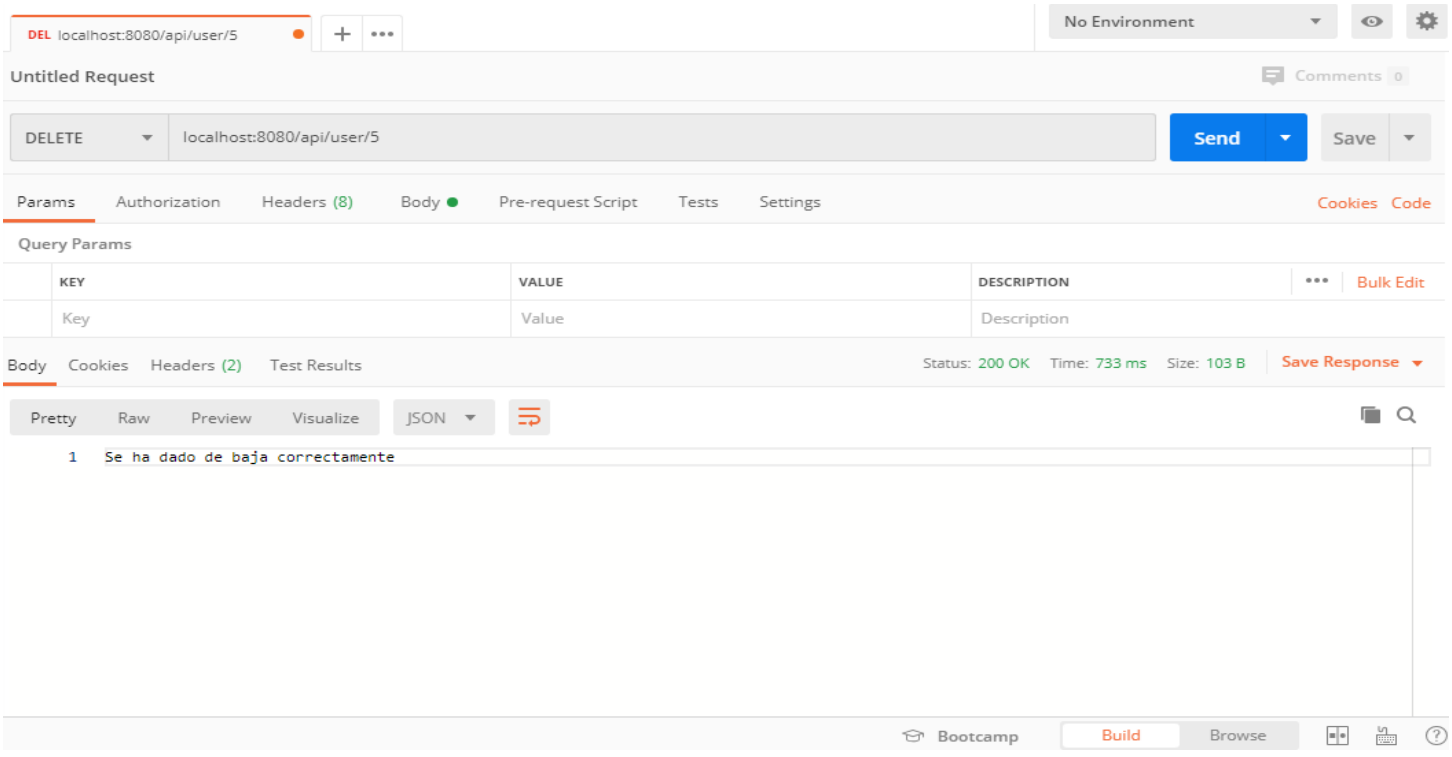
  Como sólo nos interesa el id, no habrá cuerpo. Un mensaje al cliente sería que el dispositivo se ha dado de baja correctamente.



- "/api/user/:idusuario"

  Dado el id del usuario, se borrará todo dato de dicho usuario. Como está configurado para un borrado en CASCADE, los datos de todos los dispositivos asociados al usuario, se borrarán.

  Como sólo nos interesa el id, no habrá cuerpo. Un mensaje al cliente sería que se ha dado de baja correctamente.



4.- CUARTA ITERACIÓN (MQTT)

4.1.- CANAL SENSOR

El propósito de este canal es recibir todas las lecturas de los sensores que el módulo ESP8266 recoge, así si un cliente se suscribe a este canal podrá tener acceso a las lecturas del sensor.

El formato de mensaje será un JSON que constará con los siguientes parámetros:

- Idsensor: el id del sensor que haya hecho la lectura.
- Value: valor de la lectura del sensor.
- Timestamp: indica la hora en la que se realizó la lectura.
- Accuracy: representa el porcentaje de error que el sensor tiene al realizar una lectura.

Ejemplo:

```
{
        "idsensor" : 10,
        "value" : 650,
        "timestamp" : 1587252138,
        "accuracy" : 1
}
```

4.2.- CANAL INFO

En este canal el módulo ESP8266 se encarga de enviar mensajes en intervalos de tiempo sobre el estado actual de los sensores para saber si está en funcionamiento o no.

El formato de mensaje será de texto plano que constará con un mensaje que nos diga el id del sensor y su estado actual:

“El sensor con id 1 está capturando información”

“Error!!! el sensor con id 5 no responde”

4.3.- CANAL RIEGO

El propósito de este canal es que un cliente envíe un mensaje el cual será recibido por el modulo ESP8266 el cual hará un riego de agua al sensor que se haya indicado en el mensaje si la planta en cuestión no tiene una humedad demasiado alta para evitar dañar a la planta, esto se indicara con un mensaje de respuesta que indicara si el riego se realizó o no.

El formato del mensaje que va a ser recogido por el módulo será un JSON en donde se indique el id del sensor que va a realizar el riego.

Cuando el modulo reciba este mensaje se encarga de realizar el riego solo si se detecta que la humedad actual no es demasiado alta en comparación al umbral de humedad que este configurado, en este caso se responde con un mensaje de texto plano que diga que la operación se ha realizado:

“El riego se ha realizado exitosamente”

En caso contrario se debe enviar una respuesta que diga que no ha podido realizar el riego:

“Peligro! El riego ha fallado, la humedad actual de la planta es muy alta y se ha cancelado el riego para evitar daños a la planta”