**Juan Camilo Restrepo Velez 000373886**

Tabla de contenido

[1) Ajuste MariaDB 2](#_Toc40629309)

[2) Ajuste API REST 3](#_Toc40629310)

[2.1) updateData.php 3](#_Toc40629311)

[2.2) postData.php 6](#_Toc40629312)

[3) ConexionApiRest 8](#_Toc40629313)

[3.1) Constructor 8](#_Toc40629314)

[3.2) getData 8](#_Toc40629315)

[3.3) setData 9](#_Toc40629316)

[3.4) updateData 9](#_Toc40629317)

[3.5) downloadData 10](#_Toc40629318)

[3.6) isConnected 10](#_Toc40629319)

[3.7) setHttps 11](#_Toc40629320)

[3.8) tryConnect 11](#_Toc40629321)

[Referencias 12](#_Toc40629322)

# 1) Ajuste MariaDB

Se debe adicionar en la base de datos EduApps de MariaDB las siguientes tablas:

USE EduApps;

CREATE TABLE Estados

    (ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

     Descripcion VARCHAR(255) NOT NULL);

CREATE TABLE Estudiantes\_Actividades

    (Estudiante\_ID INT NOT NULL,

     Actividad\_ID INT NOT NULL,

     Estado\_ID INT NOT NULL,

     Observaciones VARCHAR(255) NOT NULL,

     PRIMARY KEY (Actividad\_ID, Estudiante\_ID),

     CONSTRAINT FKEstudiante\_Actividad

     FOREIGN KEY (Actividad\_ID) REFERENCES Actividades(ID)

      ON DELETE CASCADE

      ON UPDATE RESTRICT,

     CONSTRAINT FKActividad\_Estudiante

     FOREIGN KEY (Estudiante\_ID) REFERENCES Estudiante(ID)

      ON DELETE CASCADE

      ON UPDATE RESTRICT,

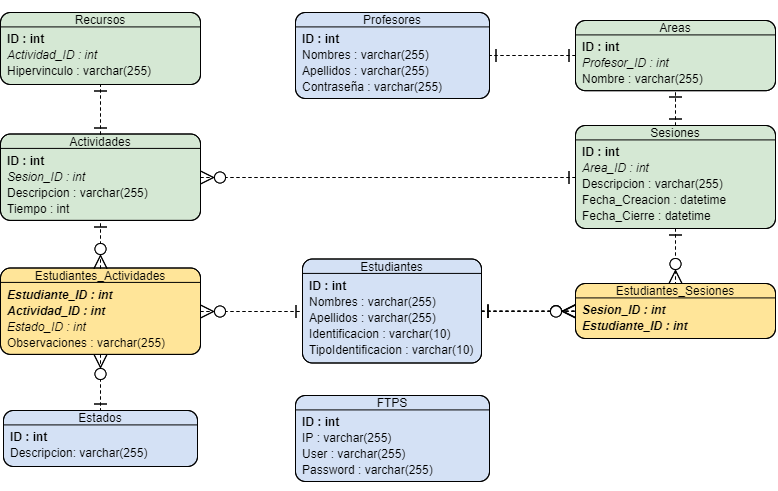
     CONSTRAINT FKActividad\_Estado

     FOREIGN KEY (Estado\_ID) REFERENCES Estados(ID)

      ON DELETE CASCADE

      ON UPDATE RESTRICT);

Dejando el siguiente modelo relacional en el nodo:



Por otro lado, se deben realizar la siguiente inserción:

INSERT INTO Estados VALUES (0,‘Iniciada’),(0,‘Finalizada’),(0,‘Abandonada’);

# 2) Ajuste API REST

Como el API REST que se decidió implementar no está completo (faltan las funciones de update y delete) se nos presenta el impedimento para poder actualizar el estado de la actividad por lo cual debemos implementar al menos la funcionalidad de actualizar los datos, además de mejorar la función de insertar datos ya que no nos devuelve el registro que se acaba de insertar (se requiere para notificar con cuál ID quedó dicho registro).

## 2.1) updateData.php

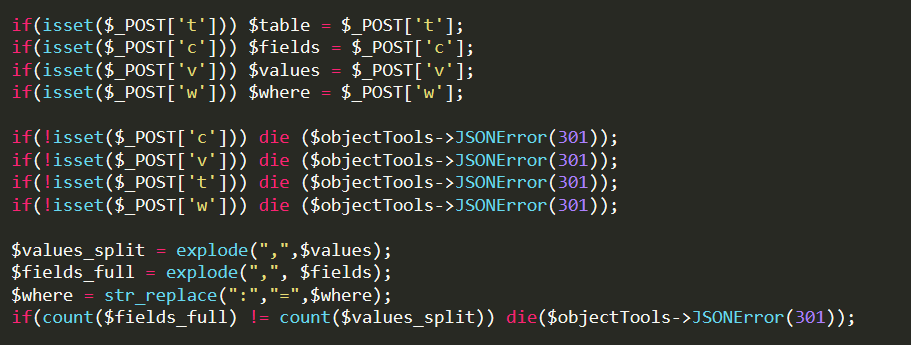
Ahora bien, para poder actualizar datos por medio del API REST se escribe el código necesario en el archivo updateData.php para implementar dicha función por medio de la URL que se deben enviar los siguientes parámetros mediante el método POST:

* Tabla
* Columna
* Valor
* Condición

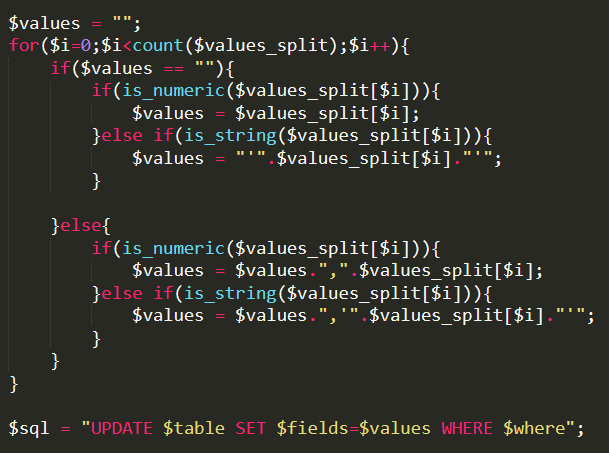
Entonces la URL queda (recordar que da tanto con HTTP como HTTPS por la instalación del certificado del sprint 3):

*https://192.168.0.101/ApiRest/updateData.php?t=table&c=column&v=newValue&w=condicion*

Para el ***código en php*** se reutilizan todas las validaciones que se hacen los demás archivos php por lo que solo se mostrará lo esencial como la obtención de los parámetros:



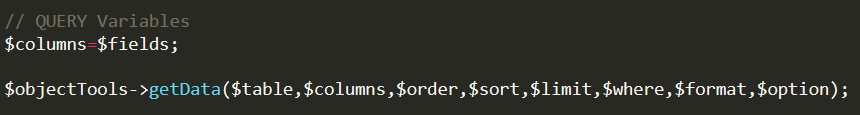
Luego se realizan las validaciones posteriormente mencionadas, para luego realizar la construcción de la instrucción update



Luego se utiliza una clase Tools que se encarga de la conexión con el servidor MariaDB y aplica la sentencia update:

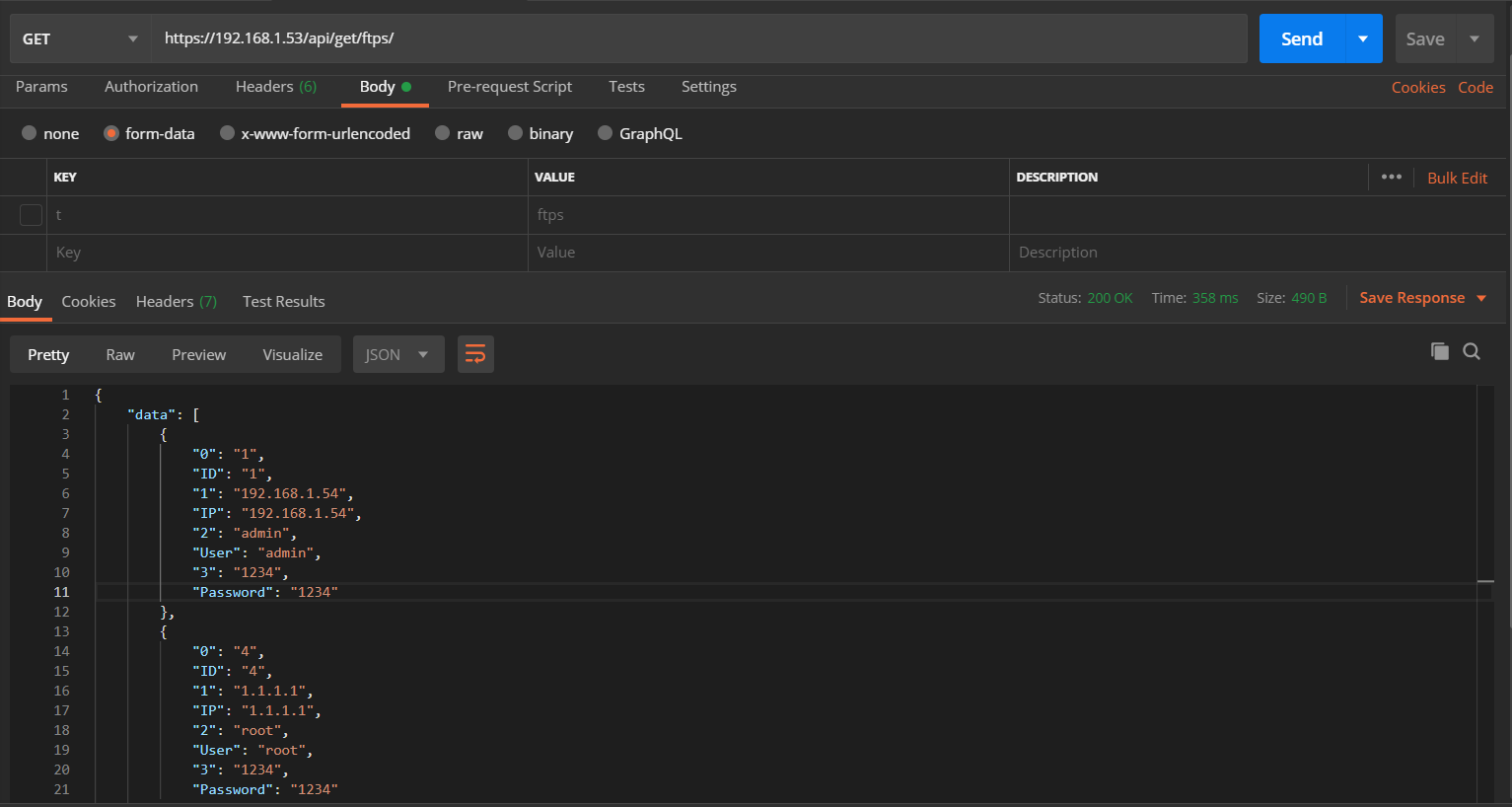


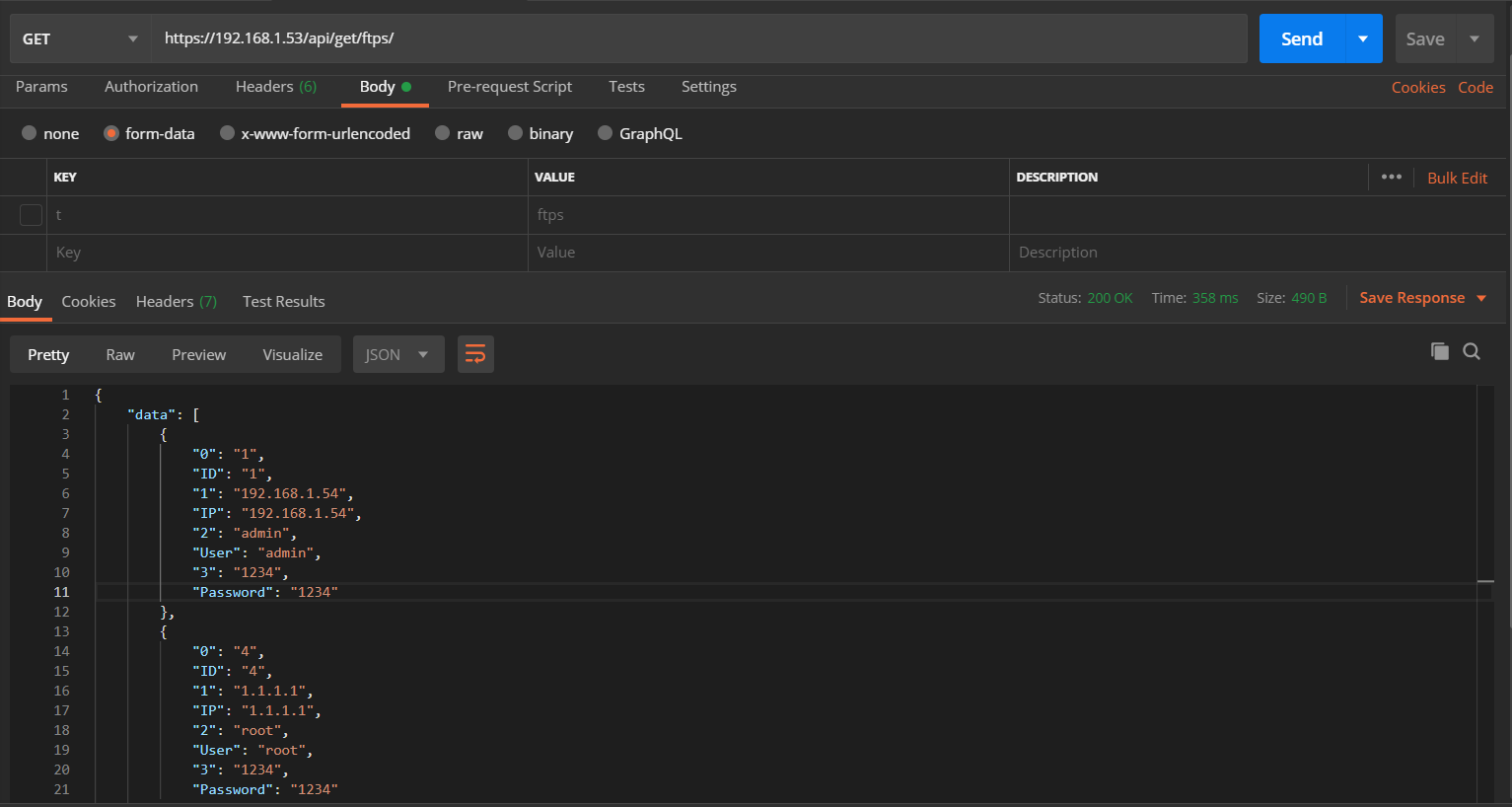
Luego se procede a llamar nuevamente la clase, pero esta vez devolver un JSON con el ID y la columna modificada de la tabla:



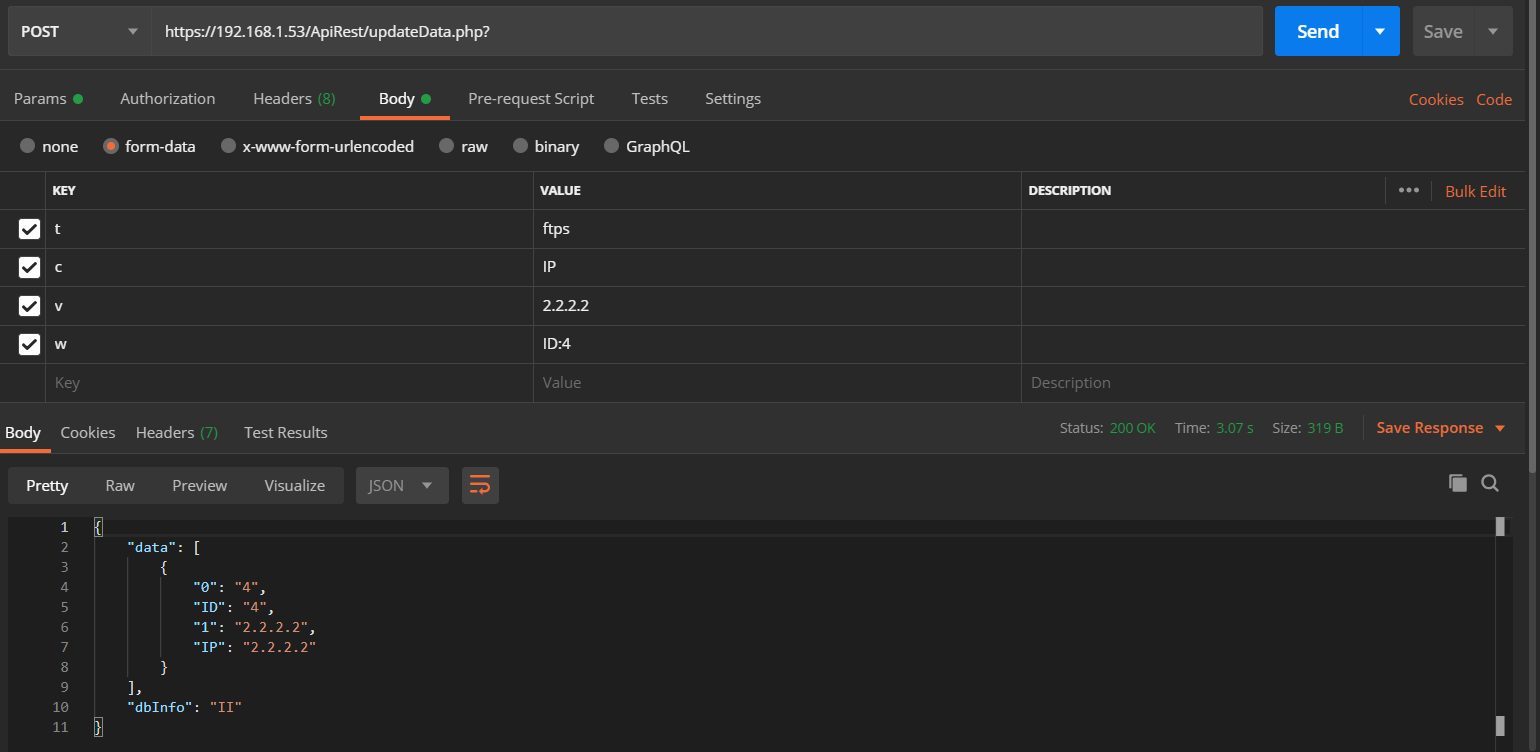
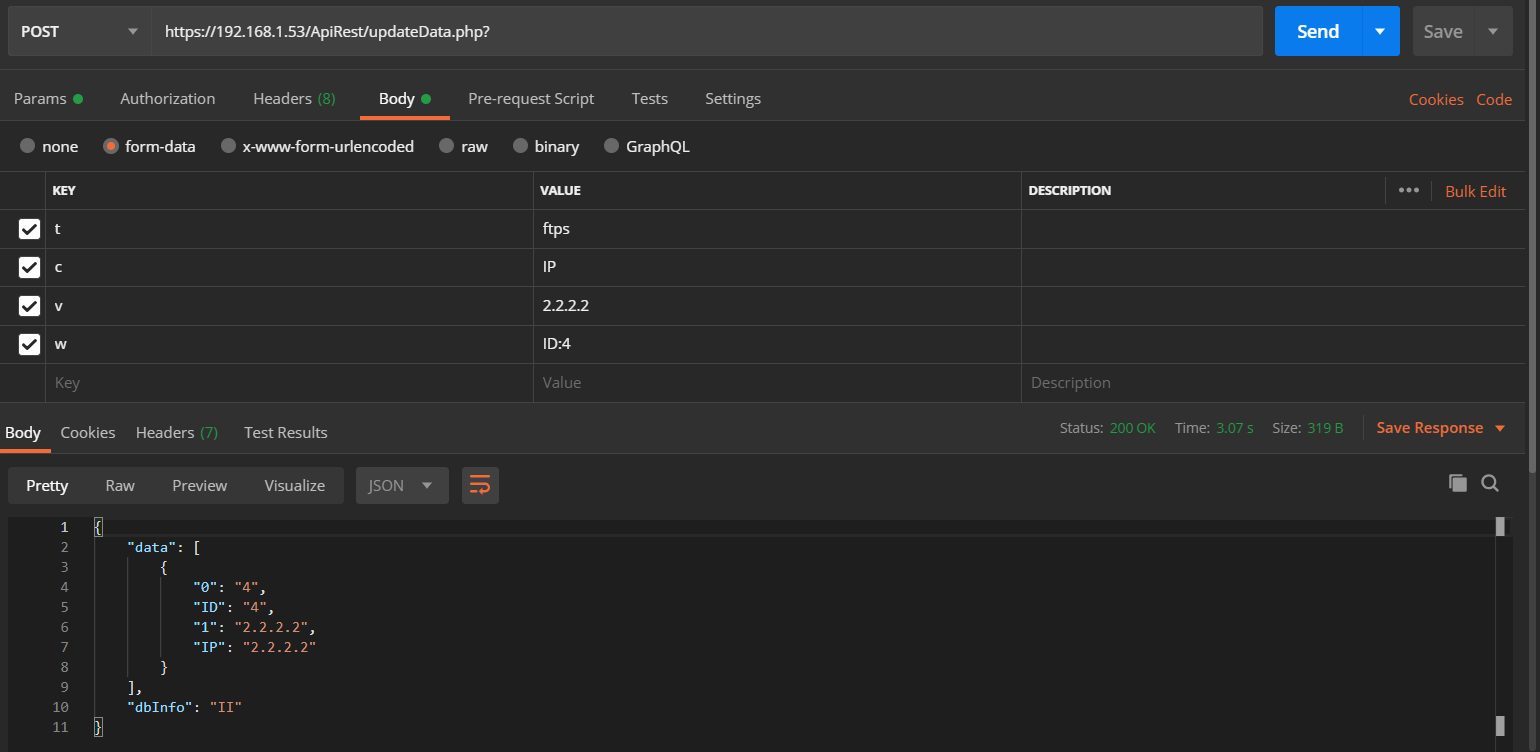
***Prueba desde Postman*** que es un programa que nos permite probar las funcionalidades de nuestra API REST para enviar parámetros por la URL por medio de los diferentes modos (GET, POST, PUT, entre otros).

Primero consultamos los registros que tiene la tabla ftps (Estos son los del servidor de prueba):

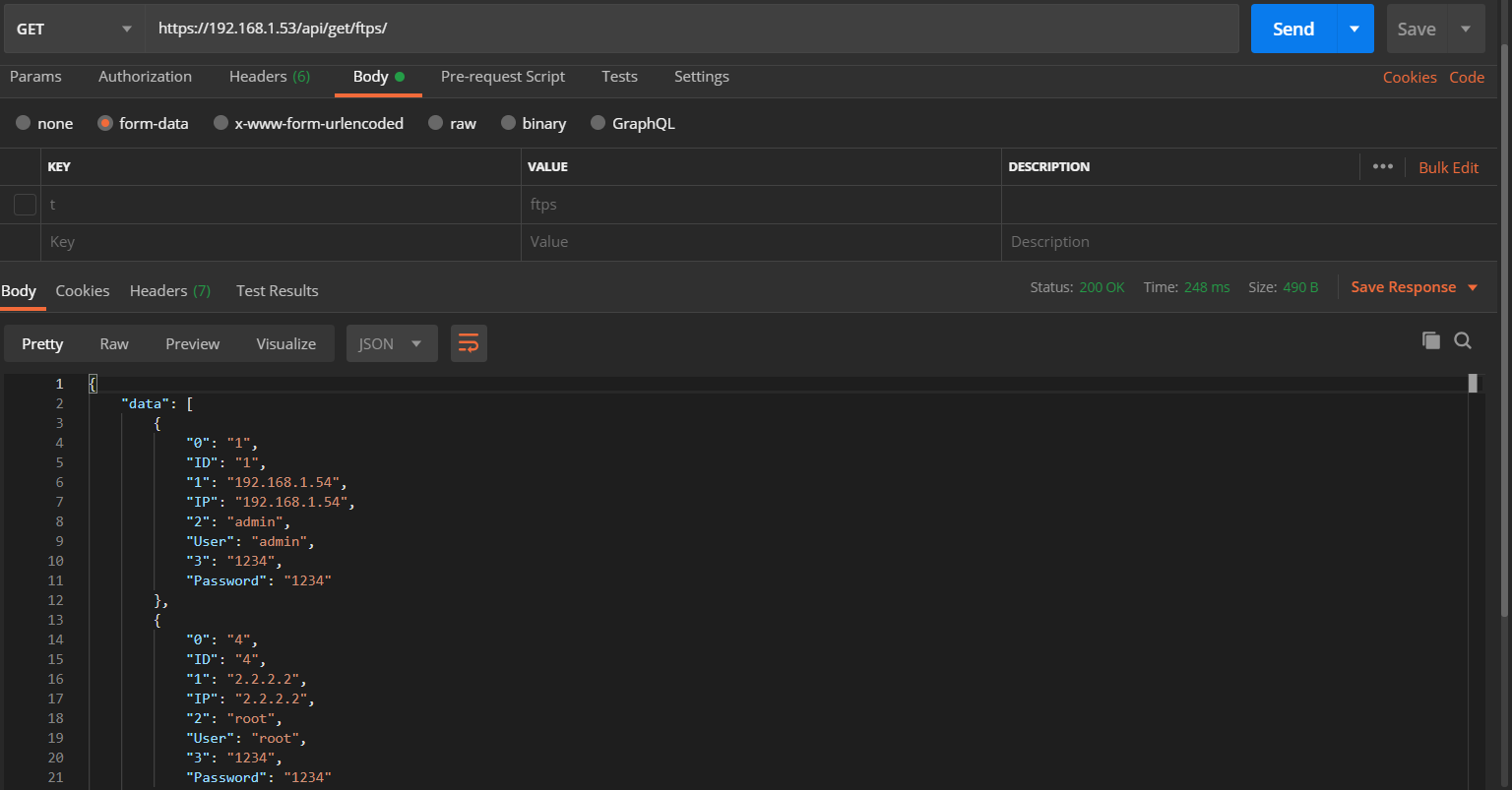
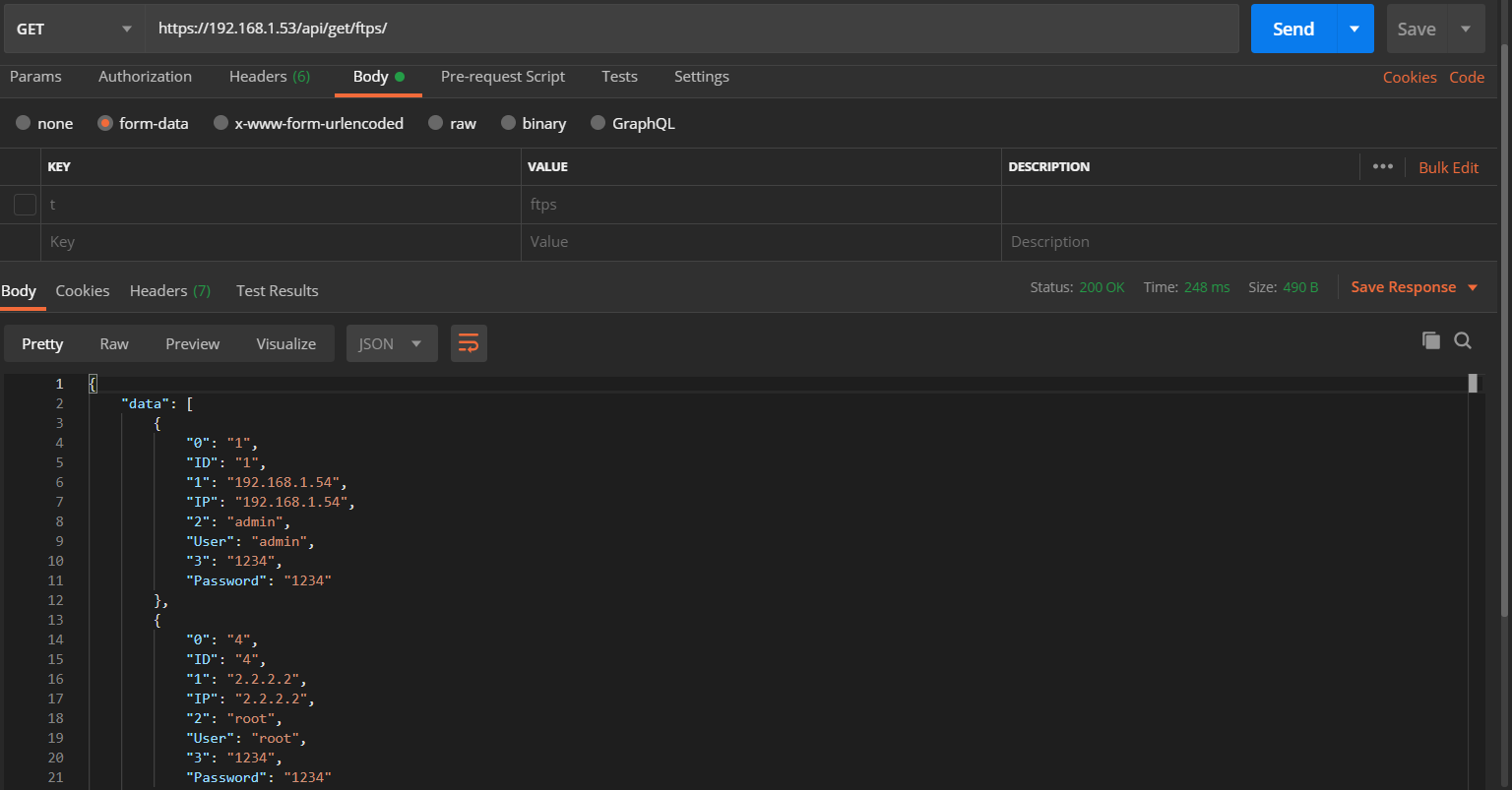




Luego se procede a cambiar la IP del registro con ID igual a 4:

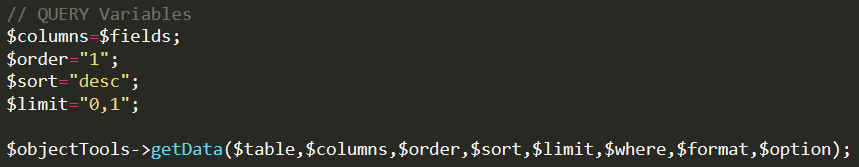


Y ahora volvemos a consultar todos los registros:



## 2.2) postData.php

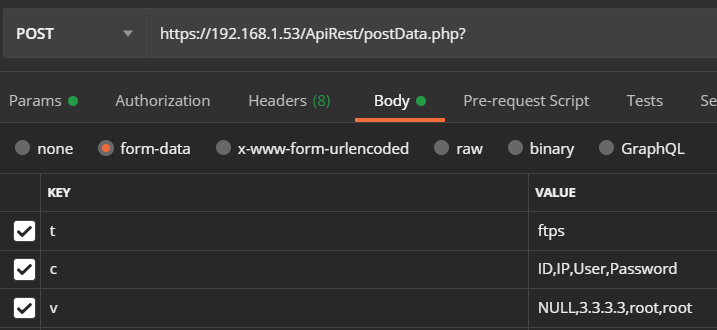
Por otro lado, la opción de insertar datos ya viene implementada, pero esta retorna el ID incorrecto del registro que se acaba de insertar por lo que se modifica la parte después de construir la consulta insert, que se hace lo mismo que con update, se llama un método de la clase Tools para consultar el último registro de la tabla, por ello se especifican todas las columnas, se ordena descendentemente por el ID y se limita solo a un resultado:

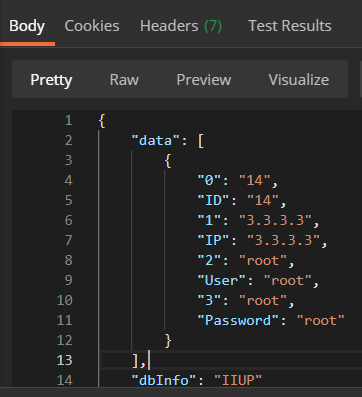


Al igual que update, se envían los parámetros por la URL así con el método POST:

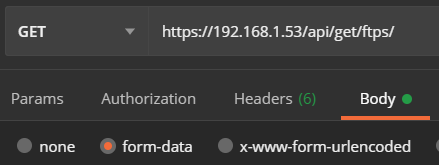
*https://192.168.0.101/ApiRest/postData.php?t=table&c=column1,column2&v=value1,value2*

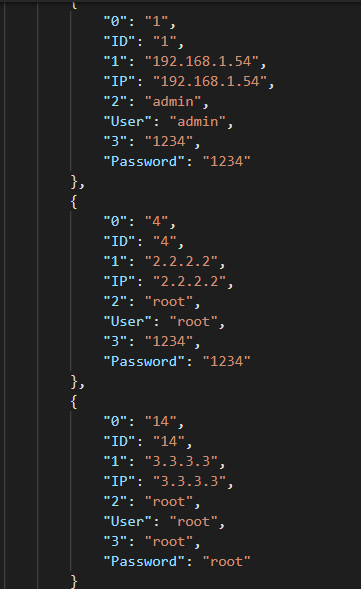
***La prueba en el programa Postman*** para insertar un nuevo registro:





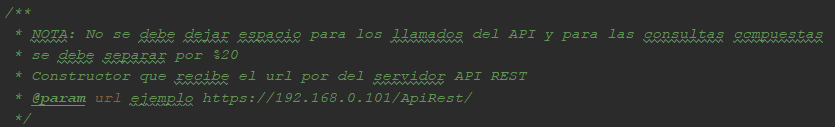
Y la consulta sobre la tabla donde se ve el nuevo registro:





# 3) ConexionApiRest

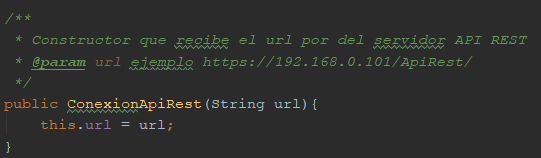
Con un mejor entendimiento del API REST se realiza una mejora de la clase ***ConexionApiRest*** haciendo que sea de más alto nivel para facilitar su uso y comprensión, además todos los métodos llevan el comentario con su descripción de lo que realiza, recibe y retornar, ejemplo:



Y como se ve en el comentario, en todos los llamados de los métodos del API REST no se puede dejar espacio ya que estos se hacen por la url, entonces para las consultas compuestas se debe separar escribiendo *%20*

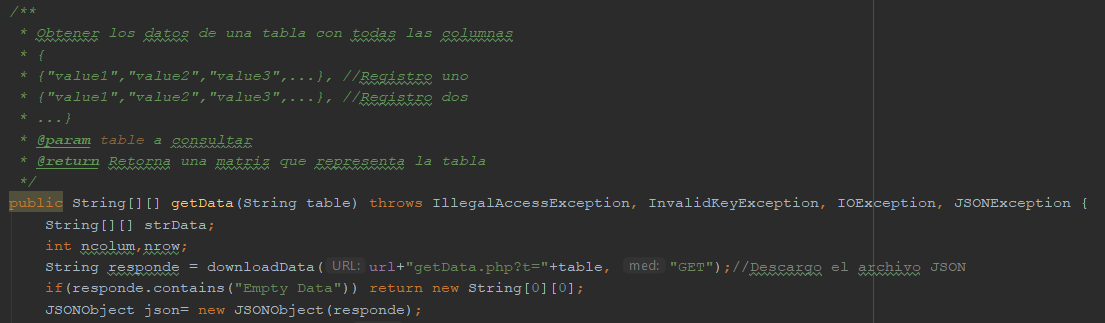
## 3.1) Constructor

Recibe la URL del servidor: *https://192.168.0.101/ApiRest/*

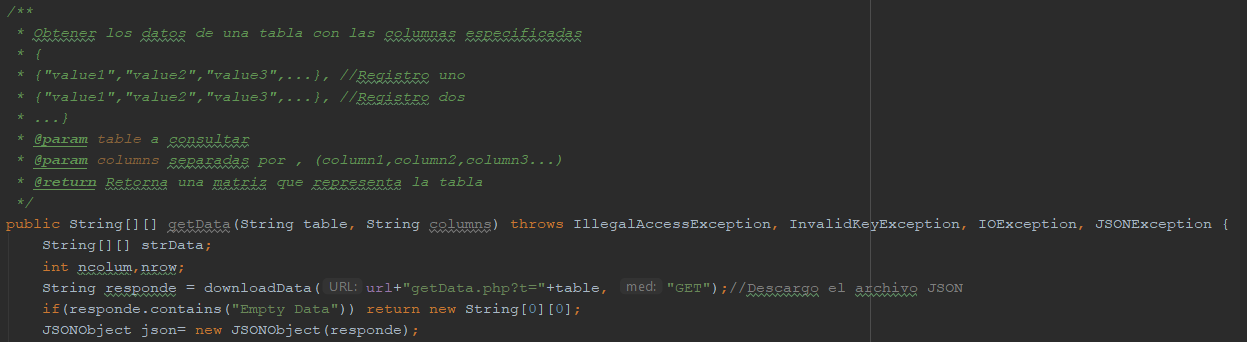


## 3.2) getData

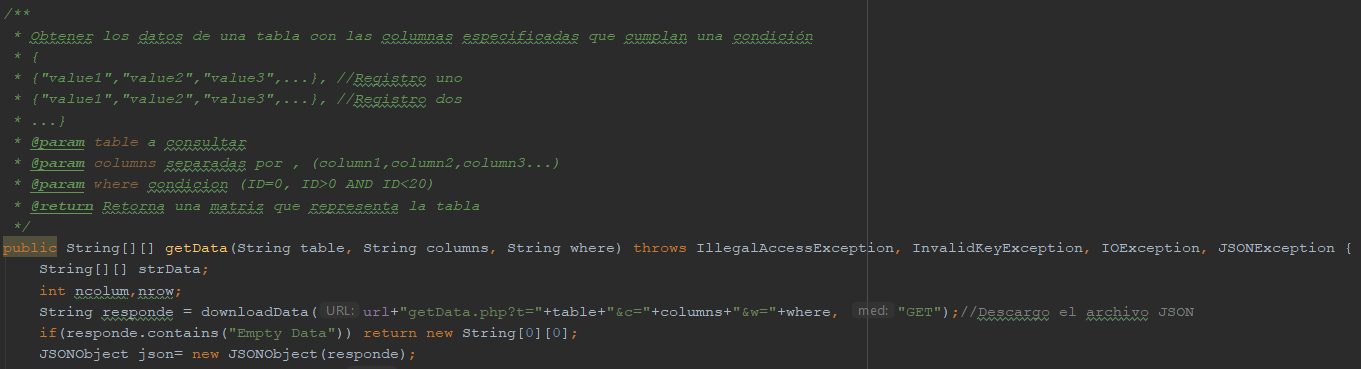
En la decodificación del JSON se mantiene, cambia la forma en que como se llama y en los parámetros que recibe. En este sentido puede obtener el contenido de toda la tabla:



Obtener solo algunas columnas de la tabla:

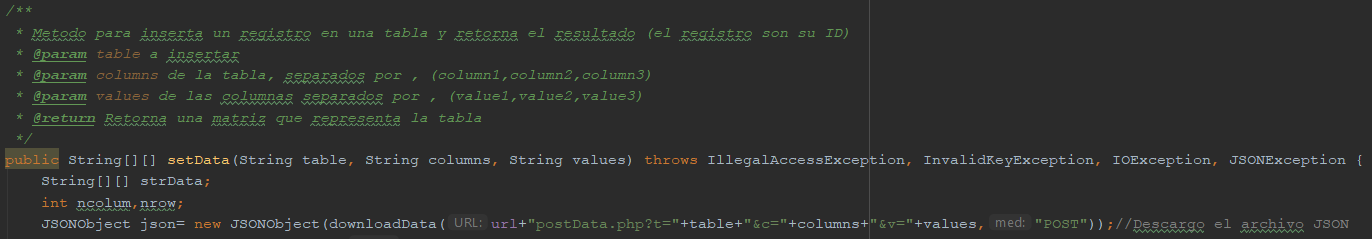


Obtener algunas columnas filtradas por un where:



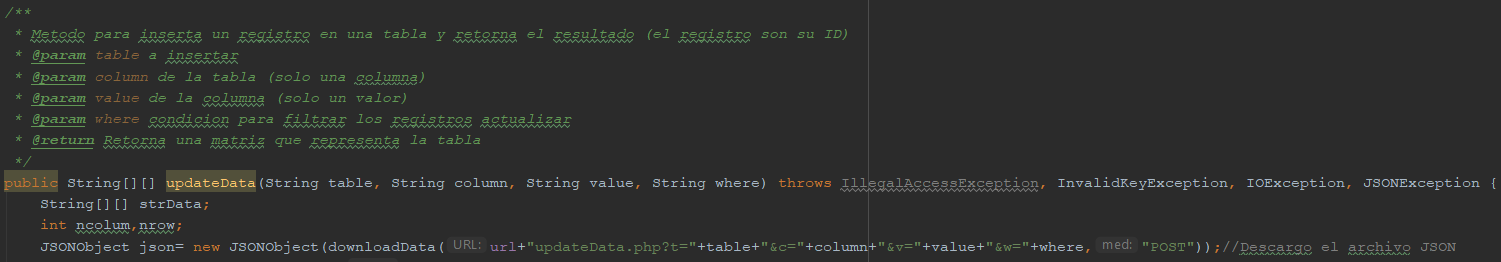
## 3.3) setData

El método enviar por la URL los parámetros para realizar la inserción por el método POST de HTTP y como también devuelve un JSON, tiene la misma decodificación que el método getData, por lo que cambia el URL a enviar a download y se especifica el método POST:



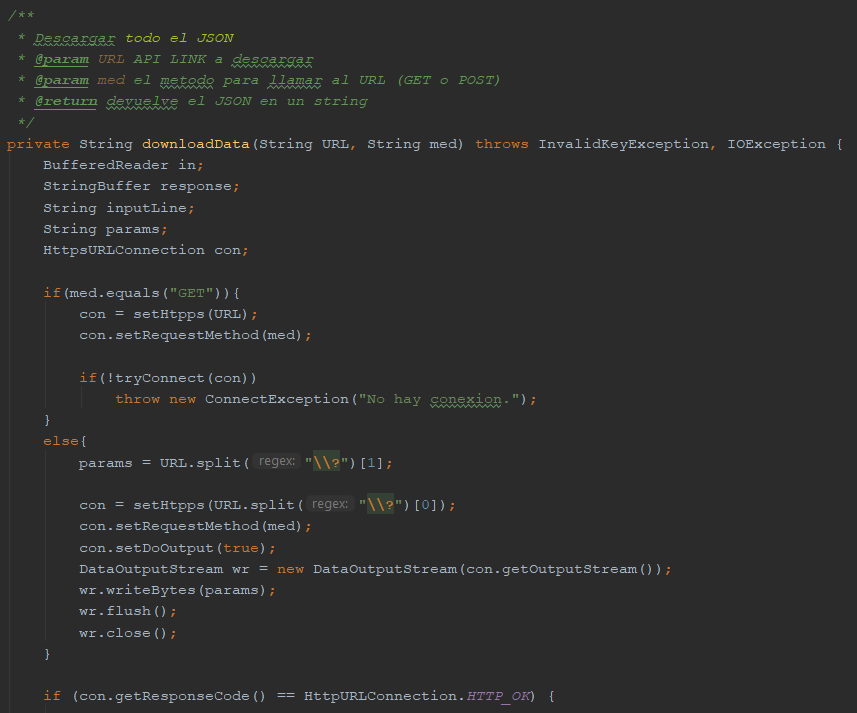
## 3.4) updateData

Al igual que setData, solo cambia el URL a enviar a download, ya que también se devuelve un JSON con el ID y la columna cambiada:



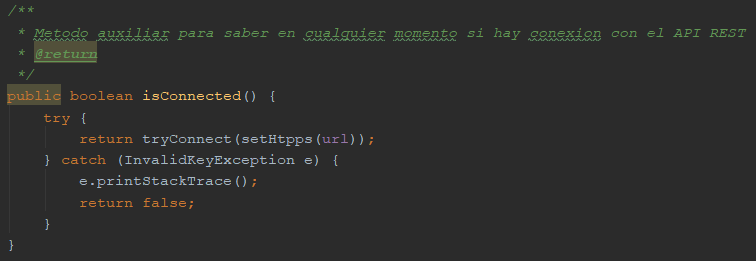
## 3.5) downloadData

Método privado que se encarga de descargar los JSON que retornan los API LINK, además de llamar un método para comprobar la conexión con la página cuando se van a consultar datos:



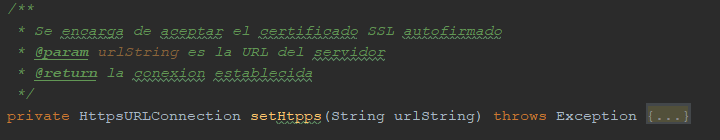
## 3.6) isConnected

Método para saber si el dispositivo tiene conexión al API REST en cualquier momento, este será llamado siempre que se vayan a realizar las actualizaciones o sincronizaciones en el nodo para que la app no se pared y deje de responder:



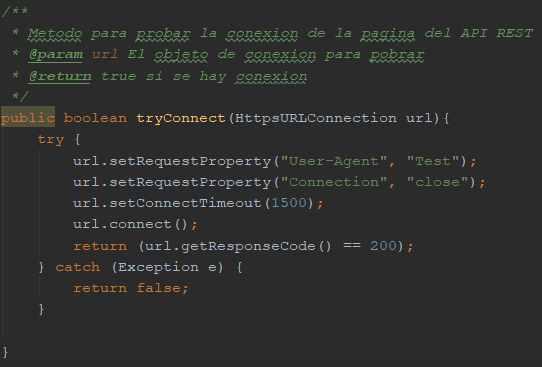
## 3.7) setHttps

Método privado que se encarga de gestionar el permiso para aceptar la conexión con servidor ya que este tiene un certificado SSL auto-firmado:



## 3.8) tryConnect

Método para probar la conexión de una página (API LNK) de forma rápida y evitando que se cuelgue la app cuando no se pueda acceder al nodo:



# Referencias

Rodriguez. (2020). Automatic API REST Droidcon. Slideshare.net. Retrieved 5 May 2020, from <https://www.slideshare.net/AlejandroEsquiva/automatic-apirestdroidcon>