

iPet,

el collar inteligente para mascotas.



Tercera iteración – 17 de abril del 2020

Autores:

Heredia Pérez, Elena

Reina Ballesta, Irene

Tejera García, Juan Manuel

Desarrollo de Aplicaciones Distribuidas

Introducción

iPet, el collar inteligente para mascotas, diseñado para obtener un cuidado y una atención adicional para su mascota.

Se trata de un aparato que consta de **diversos tipos de sensores y de herramientas** que proporcionan al usuario un seguimiento en tiempo real de lo que está ocurriendo.

Este dispositivo se colocará en la mascota a modo de collar proporcionando al usuario a **través de un smartphone** una gran variedad de funciones que tendrá al alcance de su mano.

Es válido tanto para gatos como para perros, aunque este último dispondrá de más funcionalidades. También estará disponible para mascotas de diferente peso, pudiendo elegir entre **dos diseños**: el collar, para mascotas con mayor tamaño, o el arnés, para los más pequeños.

Funcionalidad

A continuación, vamos a mostrar qué funciones incorporará nuestro dispositivo:

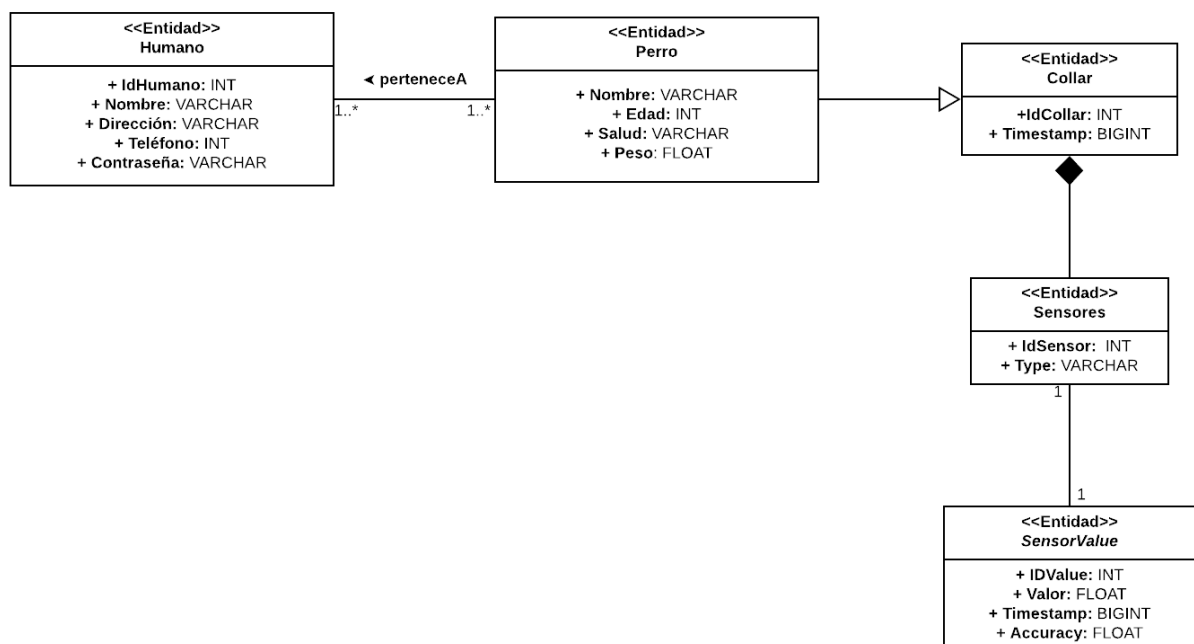
- **Temperatura**: Muestra al usuario la temperatura actual de la mascota y enviará una notificación en el caso de que haya cambios bruscos o circunstancias extremas.
- **Ritmo cardíaco**: Al igual que la temperatura se le mostrará en la pantalla notificando al usuario de valores peligrosos.
- **Ubicación**: Permite al usuario conocer la ubicación de su mascota.
- **Aviso si el perro sale de casa**: Si su mascota se escapa, será avisado por una notificación indicando la fecha aproximada de la fuga.
- **Detector de ladridos**: Cuando el animal comience a emitir muchos sonidos se le será notificado por la aplicación y tendrá opciones para actuar con un estímulo auditivo, en concreto dispondrá de dos modos:
 - **Modo educación**: Se recriminará una mala acción o se premiará mediante dos grabaciones que usted mismo podrá editar.
 - **Modo guardián**: Se notificará cuando se escuchen muchos ladridos, también se da opción de premiar o calmar al animal con estímulos sonoros.
- **Modo nocturno**: Dispondrá de este modo el cual constará de una tira de led RGB que podrás activarla para no perder de vista a tu perro al salir de paseo. Además, tendrá instalado un fotodiodo que controlará la intensidad de los leds dependiendo de cuanta luminosidad haya.
- **Resistencia al agua y al polvo**.
- **Controlado a través de Telegram**: Desarrollaremos una aplicación a través de la aplicación Telegram, que mediante comandos nos permitirá controlar todas las funciones del iPet.
- **Reproducción de mensajes**: Si usted quiere darle cariño a su mascota mediante un audio podrá hacerlo mediante una previa grabación y una función de reproducción.
- **Captación de sonidos**: Con esta función podrá oír lo que su mascota está escuchando en ese momento, es ideal si quiere saber si le saltó una alarma en casa, se dejó la cafetera encendida ...
- **Modo bajo consumo**: Se reducen algunas funcionalidades del dispositivo con el objetivo de que se produzca una larga duración de la batería. Es una buena opción si va a dejar sola a su mascota un periodo más largo de tiempo.
- **Cámara incorporada**: Podrá ver todo lo que su mascota ve, con esta función se le brinda seguridad de que su perro no hace ninguna travesura sin que usted lo

sepa o simplemente disfrutar del placer de observar sus movimientos en primera persona.

- **Calorías quemadas y pasos dados:** ¿Su mascota tiene problemas de obesidad? Si es así esta función es ideal para llevar un seguimiento de su ejercicio y mantener en forma a su mascota de manera saludable y controlada.
- **NFC:** En todo momento cualquier persona podrá saber cómo se llama su mascota, dónde vive, quienes son sus dueños y un teléfono de contacto. Si su mascota se pierde cualquier persona que lo encuentre podrá devolverlo a su hogar.
- **Batería recargable.**
- **Alimentarlo a distancia:** Podrás tener un seguimiento estricto de su dieta controlando cuando come, producto puede incluir un comedero que dispensa las raciones adaptadas a sus necesidades. (Se podría añadir un timer para controlar cuanta cantidad de comida suministrar).
- **Puerta inteligente:** Tan sólo tu mascota será capaz de abrir la escotilla para entrar en casa evitando sorpresas inesperadas.

Diagrama UML

Nuestra base de datos está creada a partir de este **diagrama UML**, en el que podemos ver tanto las diferentes tablas que hemos implementado con sus atributos como sus herencias.



Base de datos

Como bien se muestra en el diagrama anterior, la **base de datos** consta de **5 tablas** en las cuales iremos almacenando los datos necesarios para el funcionamiento de nuestra aplicación distribuida. A continuación, iremos mostrando una a una los atributos de cada una de ellas.

Tabla de humano.







Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 idHumano	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 nombre	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 direccion	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 telefono	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 contraseña	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 idPerro	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Tabla de perro.






Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 nombre	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 edad	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 salud	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 idCollar	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 peso	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Tabla de collar.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 idCollar	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 timestamp	BIGINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Tabla de sensor.









Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 idSensor	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 idCollar	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 type	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Tabla de valores del sensor.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 idValue	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 valor	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 timestamp	BIGINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 accuracy	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 idSensor	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

API Rest

Hemos realizado una serie de métodos que se encargarán de trabajar con los datos en nuestra base de datos. Las operaciones más importantes relacionadas con los datos en los sistemas REST son: **POST** que los usaremos para actualizar datos de las tablas. **GET** para realizar consultas y obtener resultados con búsquedas. El **PUT** con el que añadiremos nuevos elementos en la base de datos, y por último el **DELETE** para eliminar las entradas. Estos métodos los podremos usar siempre a través de la URL, y estas son las que usaremos.

```
// -----[USUARIO]-----
router.put("/ipet/user/newUser").handler(this::addUser);
router.get("/ipet/users").handler(this::getAllUsers);
router.get("/ipet/user/:idHumano").handler(this::getUser);
router.get("/ipet/user/dog/:idHumano").handler(this::getDogByUser);
router.delete("/ipet/user/deleteUser/:idHumano").handler(this::deleteUser);
router.post("/ipet/user/updatePass").handler(this::updatePass);
router.post("/ipet/user/updateAddress").handler(this::updateAddress);
router.post("/ipet/user/updatePhone/").handler(this::updatePhone);

// -----[PERRO]-----
router.put("/ipet/dog/newdog").handler(this::addDog);
router.get("/ipet/dogs").handler(this::getAllDogs);
router.get("/ipet/dog/:idPerro").handler(this::getDog);
router.delete("/ipet/dog/deleteDog/:idCollar").handler(this::deleteDog);
router.post("/ipet/dog/updateAge").handler(this::updateAge);
router.post("/ipet/dog/updateWeight").handler(this::updateWeight);
router.post("/ipet/dog/updateHealth").handler(this::updateHealth);

// -----[COLLAR]-----
router.put("/ipet/collar/newCollar").handler(this::addCollar);
router.get("/ipet/collars").handler(this::getAllCollars);
router.get("/ipet/collar/:idCollar").handler(this::getCollar);
router.delete("/ipet/collar/deleteCollar/:idCollar").handler(this::deleteCollar);

// -----[SENSOR]-----
router.put("/ipet/sensor/newSensor").handler(this::addSensor);
router.get("/ipet/sensors/:idCollar").handler(this::getAllSensorsByCollar);
router.delete("/ipet/sensor/deleteSensor/:idSensor").handler(this::deleteSensor);

// -----[SENSORVALUE]-----
router.put("/ipet/sensorValue/newSensor").handler(this::addSensorValue);
router.get("/ipet/sensorValue/:idSensor").handler(this::getValueBySensor);
router.get("/ipet/sensorValue/collar/:idCollar").handler(this::getValueByCollar);
router.delete("/ipet/sensorValue/deleteSensor/:idSensorValue").handler(this::deleteSensorValue);
router.post("/ipet/sensorValue/updateValue").handler(this::updateValue);
router.post("/ipet/sensorValue/updateAccuracy").handler(this::updateAccuracy);
```

Antes de nada, vamos a mostrar el método que se encarga de calcular el **timestamp** para posteriormente poder utilizarlo en la **API Rest**.

```
private long fechaActual() {
    Date fecha = new Date();
    long ts = fecha.getTime();
    System.out.println(ts);

    return ts;
}
```

Métodos de usuario.

```
// -----[MÉTODO PARA AÑADIR UN USUARIO]-----
private void addUser(RoutingContext routingContext) {
    Humano humano = Json.decodeValue(routingContext.getBodyAsString(), Humano.class);
    mySQLPool.preparedQuery(
        "INSERT INTO ipet_db.humano (nombre, direccion, telefono, contraseña, idPerro) VALUES (?, ?, ?, ?, ?)",
        Tuple.of(humano.getNombre(), humano.getDireccion(), humano.getTelefono(), humano.getContraseña(),
            humano.getIdPerro()),
        handler -> {
            if (handler.succeeded()) {
                System.out.println("El usuario se ha añadido con éxito.");
                long id = handler.result().property(MySQLClient.LAST_INSERTED_ID);
                humano.setIdHumano((int) id);
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(humano).encodePrettily());
            } else {
                System.out.println("Error al añadir usuario.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePrettily());
            }
        });
}

// -----[MÉTODO PARA OBTENER TODOS LOS USUARIOS]-----
private void getAllUsers(RoutingContext routingContext) {
    mySQLPool.query("SELECT * FROM ipet_db.humano", res -> {
        if (res.succeeded()) {
            RowSet<Row> resultSet = res.result();
            System.out.println("Se han obtenido un total de " + resultSet.size() + " usuarios.");
            JSONArray result = new JSONArray();
            for (Row row : resultSet) {
                result.add(JsonObject.mapFrom(new Humano(row.getInteger("idHumano"), row.getString("nombre"),
                    row.getString("direccion"), row.getInteger("telefono"), row.getString("contraseña"),
                    row.getInteger("idPerro"))));
            }
            routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                .end(result.encodePrettily());
        } else {
            routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                .end(JsonObject.mapFrom(res.cause()).encodePrettily());
            System.out.println("Error al obtener los usuarios.");
            System.out.println(res.cause().toString());
        }
    });
}

// -----[MÉTODO PARA OBTENER UN USUARIO]-----
private void getUser(RoutingContext routingContext) {
    mySQLPool.query("SELECT * FROM ipet_db.humano WHERE idHumano = " + routingContext.request().getParam("idHumano"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();
                System.out.println("Se ha obtenido el usuario asociado al id. "
                    + routingContext.request().getParam("idHumano") + ".");
                JSONArray result = new JSONArray();
                for (Row row : resultSet) {
                    result.add(JsonObject.mapFrom(new Humano(row.getInteger("idHumano"),
                        row.getString("nombre"), row.getString("direccion"), row.getInteger("telefono"),
                        row.getString("contraseña"), row.getInteger("idPerro"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePrettily());
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al obtener el usuario.");
                System.out.println(res.cause().toString());
            }
        });
}

// -----[MÉTODO PARA ELIMINAR UN USUARIO]-----
private void deleteUser(RoutingContext routingContext) {
    mySQLPool.query("DELETE FROM ipet_db.humano WHERE idHumano = " + routingContext.request().getParam("idHumano"),
        res -> {
            if (res.succeeded()) {
                System.out.println("El usuario con id. " + routingContext.request().getParam("idHumano")
                    + " eliminado correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El usuario con id. " + routingContext.request().getParam("idHumano")
                    + " eliminado correctamente.");
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al eliminar el usuario.");
                System.out.println(res.cause().toString());
            }
        });
}
```



```
// -----[MÉTODO PARA OBTENER EL PERRO DE UN USUARIO]-----
private void getDogByUser(RoutingContext routingContext) {

    mySQLPool.query(
        "SELECT * FROM ipet_db.perro LEFT JOIN ipet_db.humano ON perro.idCollar = humano.idPerro WHERE idHumano = "
        + routingContext.request().getParam("idHumano"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();
                System.out.println("Se ha obtenido el perro asociado al id. del usuario "
                    + routingContext.request().getParam("idHumano") + ".");
                JSONArray result = new JSONArray();
                for (Row row : resultSet) {
                    result.add(JsonObject.mapFrom(new Perro(row.getString("nombre"), row.getInteger("edad"),
                        row.getString("salud"), row.getInteger("idCollar"), row.getFloat("peso"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePrettily());
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al obtener el perro.");
                System.out.println(res.cause().toString());
            }
        }
    );
}

// -----[MÉTODO PARA ACTUALIZAR LA CONTRASEÑA DE UN USUARIO]-----
private void updatePass(RoutingContext routingContext) {
    Humano humano = Json.decodeValue(routingContext.getBodyAsString(), Humano.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.humano SET contraseña = ? WHERE idHumano = ?",
        Tuple.of(humano.getContraseña(), humano.getIdHumano()), res -> {
            if (res.succeeded()) {
                System.out.println("La contraseña del id. usuario " + humano.getIdHumano()
                    + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("La contraseña del id. usuario " + humano.getIdHumano()
                        + " actualizada correctamente.");
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al actualizar la contraseña.");
                System.out.println(res.cause().toString());
            }
        }
    );
}

// -----[MÉTODO PARA ACTUALIZAR LA DIRECCION DE UN USUARIO]-----
private void updateAddress(RoutingContext routingContext) {
    Humano humano = Json.decodeValue(routingContext.getBodyAsString(), Humano.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.humano SET direccion = ? WHERE idHumano = ?",
        Tuple.of(humano.getDireccion(), humano.getIdHumano()), res -> {
            if (res.succeeded()) {
                System.out.println(
                    "La dirección del id. usuario " + humano.getIdHumano() + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json").end(
                    "La dirección del id. usuario " + humano.getIdHumano() + " actualizada correctamente.");
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al obtener la dirección.");
                System.out.println(res.cause().toString());
            }
        }
    );
}

// -----[MÉTODO PARA ACTUALIZAR EL TELEFONO DE UN USUARIO]-----
private void updatePhone(RoutingContext routingContext) {
    Humano humano = Json.decodeValue(routingContext.getBodyAsString(), Humano.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.humano SET telefono = ? WHERE idHumano = ?",
        Tuple.of(humano.getTelefono(), humano.getIdHumano()), res -> {
            if (res.succeeded()) {
                System.out.println(
                    "El teléfono del id. usuario " + humano.getIdHumano() + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json").end(
                    "El teléfono del id. usuario " + humano.getIdHumano() + " actualizada correctamente.");
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al actualizar el teléfono.");
                System.out.println(res.cause().toString());
            }
        }
    );
}
}

```


Métodos de perro.

```
// -----[MÉTODO PARA AÑADIR UN PERRO]-----
private void addDog(RoutingContext routingContext) {
    Perro perro = Json.decodeValue(routingContext.getBodyAsString(), Perro.class);
    mySQLPool.query("INSERT INTO ipet_db.perro (nombre, edad, salud, idCollar, peso) VALUES (?, ?, ?, ?, ?)",
        Tuple.of(perro.getNombre(), perro.getEdad(), perro.getSalud(), perro.getIdCollar(), perro.getPeso()),
        handler -> {
            if (handler.succeeded()) {
                System.out.println("El perro se ha añadido con éxito.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(perro).encodePretty());
            } else {
                System.out.println("Error al añadir el perro.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        });
}

// -----[MÉTODO PARA OBTENER UN PERRO]-----
private void getDog(RoutingContext routingContext) {
    mySQLPool.query("SELECT * FROM ipet_db.perro WHERE idCollar = " + routingContext.request().getParam("idPerro"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();
                System.out.println("Se ha obtenido el perro asociado al id. "
                    + routingContext.request().getParam("idPerro") + ".");
                JSONArray result = new JSONArray();
                for (Row row : resultSet) {
                    result.add(JsonObject.mapFrom(new Perro(row.getString("nombre"), row.getInteger("edad"),
                        row.getString("salud"), row.getInteger("idCollar"), row.getFloat("peso"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePretty());
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePretty());
                System.out.println("Error al obtener el perro.");
                System.out.println(res.cause().toString());
            }
        });
}

// -----[MÉTODO PARA OBTENER TODOS LOS PERROS]-----
private void getAllDogs(RoutingContext routingContext) {
    mySQLPool.query("SELECT * FROM ipet_db.perro", res -> {
        if (res.succeeded()) {
            RowSet<Row> resultSet = res.result();
            System.out.println("Se han obtenido un total de " + resultSet.size() + " perros.");
            JSONArray result = new JSONArray();
            for (Row row : resultSet) {
                result.add(JsonObject.mapFrom(new Perro(row.getString("nombre"), row.getInteger("edad"),
                    row.getString("salud"), row.getInteger("idCollar"), row.getFloat("peso"))));
            }
            routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                .end(result.encodePretty());
        } else {
            routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                .end(JsonObject.mapFrom(res.cause()).encodePretty());
            System.out.println("Error al obtener los perros.");
            System.out.println(res.cause().toString());
        }
    });
}

// -----[MÉTODO PARA ELIMINAR UN PERRO]-----
private void deleteDog(RoutingContext routingContext) {
    mySQLPool.query("DELETE FROM ipet_db.perro WHERE idCollar = " + routingContext.request().getParam("idCollar"),
        handler -> {
            if (handler.succeeded()) {
                System.out.println("El perro con id. " + routingContext.request().getParam("idCollar")
                    + " eliminado correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El perro con id. " + routingContext.request().getParam("idCollar")
                        + " eliminado correctamente.");
            } else {
                System.out.println("Error al eliminar el perro.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        });
}
```

```
// -----[MÉTODO PARA ACTUALIZAR EL PESO UN PERRO]-----
private void updateWeight(RoutingContext routingContext) {
    Perro perro = Json.decodeValue(routingContext.getBodyAsString(), Perro.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.perro SET peso = ? WHERE idCollar = ?",
        Tuple.of(perro.getPeso(), perro.getIdCollar()), handler -> {
            if (handler.succeeded()) {
                System.out
                    .println("El peso del id. perro " + perro.getIdCollar() + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El peso del id. perro " + perro.getIdCollar() + " actualizada correctamente.");
            } else {
                System.out.println("Error al actualizar el peso.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        });
}

// -----[MÉTODO PARA ACTUALIZAR LA EDAD DE UN PERRO]-----
private void updateAge(RoutingContext routingContext) {
    Perro perro = Json.decodeValue(routingContext.getBodyAsString(), Perro.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.perro SET edad = ? WHERE idCollar = ?",
        Tuple.of(perro.getEdad(), perro.getIdCollar()), handler -> {
            if (handler.succeeded()) {
                System.out.println(
                    "La edad del id. perro " + perro.getIdCollar() + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("La edad del id. perro " + perro.getIdCollar() + " actualizada correctamente.");
            } else {
                System.out.println("Error al actualizar la edad.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        });
}

// -----[MÉTODO PARA ACTUALIZAR LA SALUD UN PERRO]-----
private void updateHealth(RoutingContext routingContext) {
    Perro perro = Json.decodeValue(routingContext.getBodyAsString(), Perro.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.perro SET salud = ? WHERE idCollar = ?",
        Tuple.of(perro.getSalud(), perro.getIdCollar()), handler -> {
            if (handler.succeeded()) {
                System.out.println(
                    "La salud del id. perro " + perro.getIdCollar() + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("La salud del id. perro " + perro.getIdCollar() + " actualizada correctamente.");
            } else {
                System.out.println("Error al actualizar la salud.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        });
}

```

Métodos de collar.

```
// -----[MÉTODO PARA AÑADIR UN COLLAR]-----
private void addCollar(RoutingContext routingContext) {
    Collar collar = Json.decodeValue(routingContext.getBodyAsString(), Collar.class);
    mySQLPool.query("INSERT INTO ipet_db.collar (timestamp) VALUES (" + fechaActual() + ")", handler -> {
        if (handler.succeeded()) {
            System.out.println("El collar se ha añadido con éxito.");
            long id = handler.result().property(MySQLClient.LAST_INSERTED_ID);
            collar.setIdCollar((int) id);

            routingContext.response().setStatusCode(200).putHeader("content-type", "application/json").end();
        } else {
            System.out.println("Error al añadir el collar.");
            System.out.println(handler.cause().toString());
            routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                .end(JsonObject.mapFrom(handler.cause()).encodePretty());
        }
    });
}

// -----[MÉTODO PARA OBTENER TODOS LOS COLLARES]-----
private void getAllCollars(RoutingContext routingContext) {
    mySQLPool.query("SELECT * FROM ipet_db.collar", res -> {
        if (res.succeeded()) {
            RowSet<Row> resultSet = res.result();
            System.out.println("Se han obtenido un total de " + resultSet.size() + " collares.");
            JSONArray result = new JSONArray();
            for (Row row : resultSet) {
                result.add(JsonObject.mapFrom(new Collar(row.getInteger("idCollar"), row.getLong("timestamp"))));
            }
            routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                .end(result.encodePretty());
        } else {
            System.out.println("Error al obtener los collares.");
            System.out.println(res.cause().toString());
            routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                .end(JsonObject.mapFrom(res.cause()).encodePretty());
        }
    });
}

// -----[MÉTODO PARA OBTENER UN COLLAR]-----
private void getCollar(RoutingContext routingContext) {
    mySQLPool.query(
        "SELECT * FROM ipet_db.collar WHERE idCollar = " + routingContext.request().getParam("idCollar"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();
                System.out.println("Se ha obtenido el collar asociado al id. "
                    + routingContext.request().getParam("idCollar") + ".");
                JSONArray result = new JSONArray();

                for (Row row : resultSet) {
                    result.add(JsonObject
                        .mapFrom(new Collar(row.getInteger("idCollar"), row.getLong("timestamp"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePretty());
            } else {
                System.out.println("Error al obtener el collar.");
                System.out.println(res.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePretty());
            }
        }
    );
}

// -----[MÉTODO PARA ELIMINAR UN COLLAR]-----
private void deleteCollar(RoutingContext routingContext) {
    mySQLPool.query("DELETE FROM ipet_db.collar WHERE idCollar = " + routingContext.request().getParam("idCollar"),
        res -> {
            if (res.succeeded()) {
                System.out.println("El collar con id. " + routingContext.request().getParam("idCollar")
                    + " eliminado correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El collar con id. " + routingContext.request().getParam("idCollar")
                        + " eliminado correctamente.");
            } else {
                System.out.println("Error al eliminar el collar.");
                System.out.println(res.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePretty());
            }
        }
    );
}
```


Métodos de sensor.

```
// -----[MÉTODO PARA AÑADIR UN SENSOR]-----
private void addSensor(RoutingContext routingContext) {
    Sensor sensor = Json.decodeValue(routingContext.getBodyAsString(), Sensor.class);
    mySQLPool.preparedQuery("INSERT INTO ipet_db.Sensores (idCollar, type) VALUES (?,?)",
        Tuple.of(sensor.getIdCollar(), sensor.getTipo()), handler -> {
            if (handler.succeeded()) {
                System.out.println("El sensor se ha añadido con éxito.");
                long id = handler.result().property(MySQLClient.LAST_INSERTED_ID);
                sensor.setIdSensor((int) id);
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(sensor).encodePrettily());
            } else {
                System.out.println("Error al añadir el sensor.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePrettily());
            }
        });
}

// -----[MÉTODO PARA OBTENER TODOS LOS SENSORES DE UN COLLAR]-----
private void getAllSensorsByCollar(RoutingContext routingContext) {
    mySQLPool.query(
        "SELECT * FROM ipet_db.sensores WHERE idCollar = " + routingContext.request().getParam("idCollar"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();
                System.out.println(
                    "Se han obtenido un total de " + resultSet.size() + " sensores del collar con id. "
                    + routingContext.request().getParam("idCollar") + ".");
                JSONArray result = new JSONArray();

                for (Row row : resultSet) {
                    result.add(JsonObject.mapFrom(new Sensor(row.getInteger("idSensor"),
                        row.getInteger("idCollar"), row.getString("type"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePrettily());
            } else {
                System.out.println("Error al obtener los sensores del collar.");
                System.out.println(res.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
            }
        });
}

// -----[MÉTODO PARA ELIMINAR UN SENSOR]-----
private void deleteSensor(RoutingContext routingContext) {
    mySQLPool.query(
        "DELETE FROM ipet_db.sensores WHERE idSensor = " + routingContext.request().getParam("idSensor"),
        res -> {
            if (res.succeeded()) {
                System.out.println("El sensor con id. " + routingContext.request().getParam("idSensor")
                    + " ha sido eliminado con éxito.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El sensor con id. " + routingContext.request().getParam("idSensor")
                    + " ha sido eliminado con éxito.");
            } else {
                System.out.println("Error al eliminar el sensor.");
                System.out.println(res.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
            }
        });
}
```

Métodos de valor del sensor.

```
// -----[MÉTODO PARA AÑADIR UN VALOR AL SENSOR]-----
private void addSensorValue(RoutingContext routingContext) {
    SensorValue sensorvalue = Json.decodeValue(routingContext.getBodyAsString(), SensorValue.class);
    long ts = fechaActual();
    MySQLPool.preparedQuery(
        "INSERT INTO ipet_db.sensorvalue (valor, timestamp, accuracy, idSensor) VALUES (?, ?, ?, ?)",
        Tuple.of(sensorvalue.getValor(), sensorvalue.getAccuracy(), sensorvalue.getIdSensor(), handler -> {
            if (handler.succeeded()) {
                System.out.println("El valor del sensor se ha añadido con éxito.");
                long id = handler.result().property(MySQLClient.LAST_INSERTED_ID);
                sensorvalue.setIdValue((int) id);
                sensorvalue.setTimestamp(ts);
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(sensorvalue).encodePrettily());
            } else {
                System.out.println("Error al añadir un sensor.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePrettily());
            }
        }
    ));
}

// -----[MÉTODO PARA OBTENER TODOS LOS VALORES DE LOS SENSORES DE UN COLLAR]-----
private void getValueByCollar(RoutingContext routingContext) {
    MySQLPool.query(
        "SELECT sensorvalue.idValue, sensorvalue.valor, sensorvalue.timestamp, sensorvalue.accuracy, sensorvalue.idSensorValue FROM ipet.sensorvalue "
        + "LEFT JOIN ipet.sensores ON sensores.idSensor = sensorvalue.idSensorValue WHERE sensores.idCollar = "
        + routingContext.request().getParam("idCollar"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();

                System.out.println("Se han obtenido los valores de los sensores del collar con id. "
                    + routingContext.request().getParam("idCollar") + ".");
                JsonArray result = new JsonArray();
                for (Row row : resultSet) {
                    result.add(JsonObject.mapFrom(new SensorValue(row.getInteger("idValue"),
                        row.getFloat("valor"), row.getLong("timestamp"), row.getFloat("accuracy"),
                        row.getInteger("idSensorValue"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePrettily());
            } else {
                System.out.println("Error al obtener valores del collar.");
                System.out.println(res.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
            }
        }
    ));
}

// -----[MÉTODO PARA OBTENER LOS VALORES DE UN SENSOR]-----
private void getValueBySensor(RoutingContext routingContext) {
    MySQLPool.query(
        "SELECT * FROM ipet_db.sensorvalue WHERE idSensor = " + routingContext.request().getParam("idSensor"),
        res -> {
            if (res.succeeded()) {
                RowSet<Row> resultSet = res.result();
                System.out.println("Se ha obtenido correctamente el valor del sensor.");
                JsonArray result = new JsonArray();
                for (Row row : resultSet) {
                    result.add(JsonObject.mapFrom(new SensorValue(row.getInteger("idValue"),
                        row.getFloat("valor"), row.getLong("timestamp"), row.getFloat("accuracy"),
                        row.getInteger("idSensor"))));
                }
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end(result.encodePrettily());
            } else {
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(res.cause()).encodePrettily());
                System.out.println("Error al obtener los valores del sensor.");
                System.out.println(res.cause().toString());
            }
        }
    ));
}
```



```
// -----[MÉTODO PARA ELIMINAR LOS VALORES DE UN SENSOR]-----
private void deleteSensorValue(RoutingContext routingContext) {
    mySQLPool.query(
        "DELETE FROM ipet_db.sensorvalue WHERE idValue = " + routingContext.request().getParam("idValue"),
        handler -> {
            if (handler.succeeded()) {
                System.out.println("El valor del sensor con id. " + routingContext.request().getParam("idValue")
                    + " eliminado correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El valor del sensor con id. " + routingContext.request().getParam("idValue")
                        + " eliminado correctamente.");
            } else {
                System.out.println("Error al eliminar el valor del sensor.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        }
    );
}

// -----[MÉTODO PARA ACTUALIZAR EL VALOR DE UN SENSOR]-----
private void updateValue(RoutingContext routingContext) {
    SensorValue sensorValue = Json.decodeValue(routingContext.getBodyAsString(), SensorValue.class);
    mySQLPool.preparedQuery(
        "UPDATE ipet_db.sensorvalue SET valor = ?, timestamp = " + fechaActual() + " WHERE idValue = ?",
        Tuple.of(sensorValue.getValor(), sensorValue.getIdValue(), handler -> {
            if (handler.succeeded()) {
                System.out.println("El valor del sensor con id." + sensorValue.getIdValue()
                    + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("El valor del sensor con id." + sensorValue.getIdValue()
                        + " actualizada correctamente.");
            } else {
                System.out.println("Error al actualizar el valor del sensor.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        }
    );
}

// -----[MÉTODO PARA ACTUALIZAR EL ACCURACY DE UN SENSOR]-----
private void updateAccuracy(RoutingContext routingContext) {
    SensorValue sensorValue = Json.decodeValue(routingContext.getBodyAsString(), SensorValue.class);
    mySQLPool.preparedQuery("UPDATE ipet_db.sensorvalue SET accuracy = ? WHERE idValue = ?",
        Tuple.of(sensorValue.getAccuracy(), sensorValue.getIdValue(), handler -> {
            if (handler.succeeded()) {
                System.out.println("La precisión del sensor con id." + sensorValue.getIdValue()
                    + " actualizada correctamente.");
                routingContext.response().setStatusCode(200).putHeader("content-type", "application/json")
                    .end("La precisión del sensor con id." + sensorValue.getIdValue()
                        + " actualizada correctamente.");
            } else {
                System.out.println("Error al actualizar la precisión del sensor.");
                System.out.println(handler.cause().toString());
                routingContext.response().setStatusCode(401).putHeader("content-type", "application/json")
                    .end(JsonObject.mapFrom(handler.cause()).encodePretty());
            }
        }
    );
}

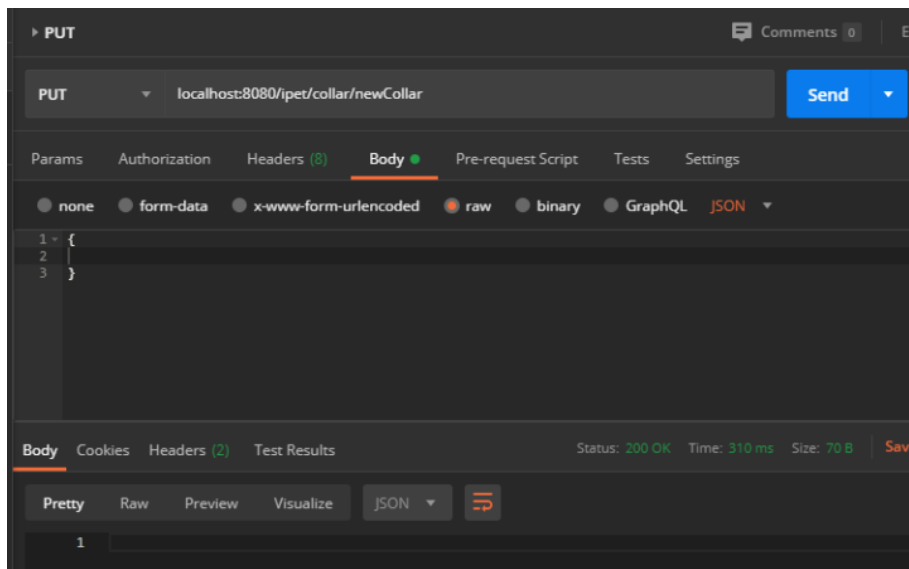
```

Pruebas de API Rest

PUT

Insertar nuevo collar.

Postman:

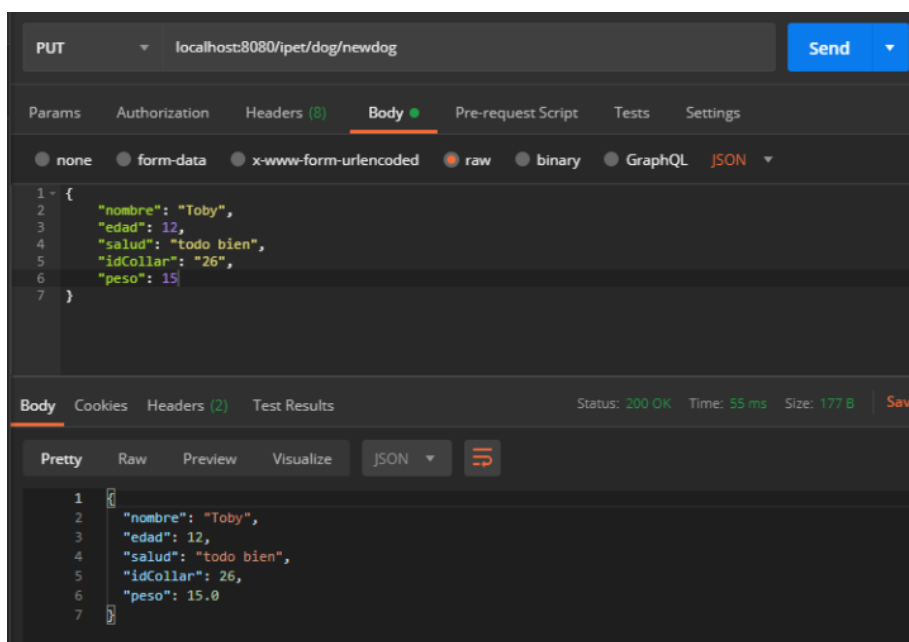


Consola de Java:

```
El collar se ha añadido con éxito.
```

Insertar nuevo perro.

Postman:

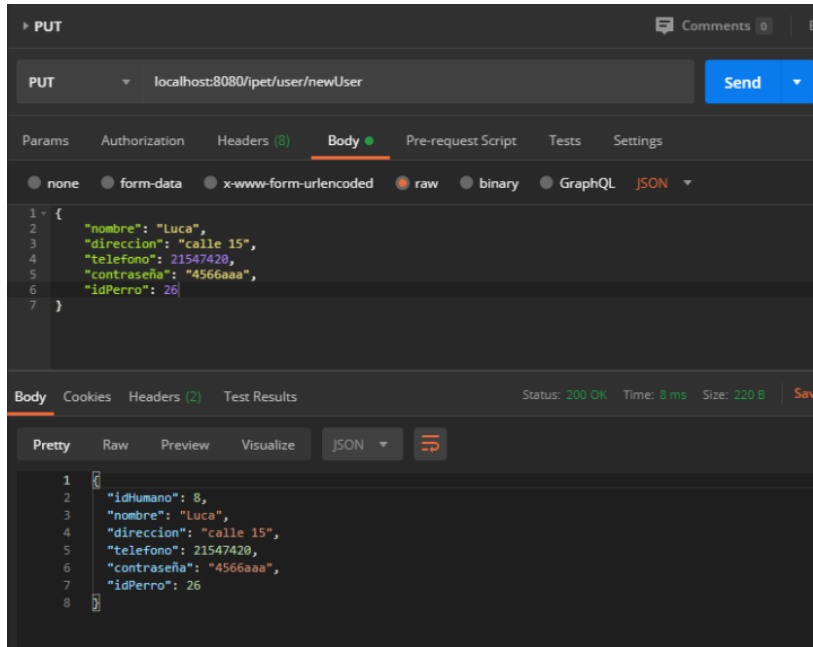


[Consola de Java:](#)

```
El perro se ha añadido con éxito.
```

[Insertar nuevo usuario.](#)

[Postman:](#)

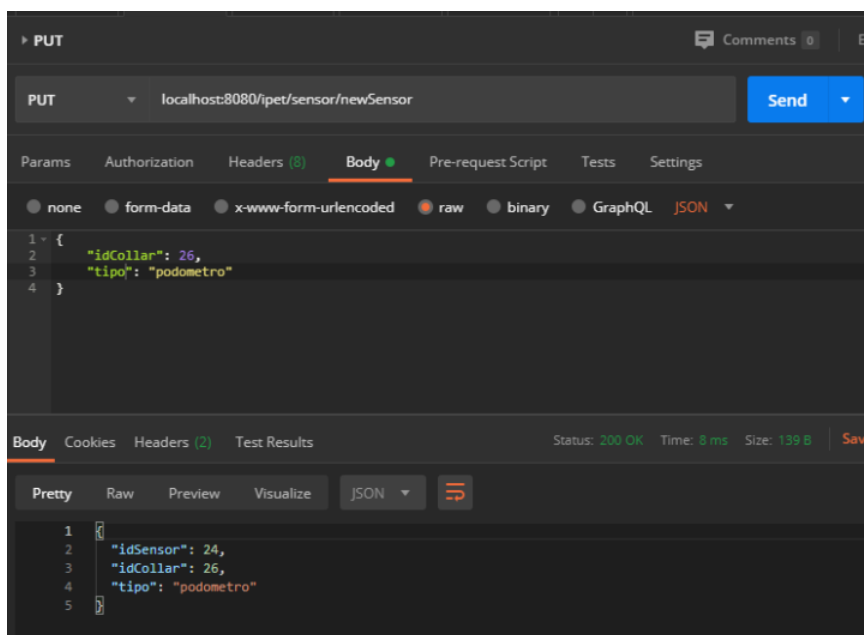


[Consola de Java:](#)

```
El usuario se ha añadido con éxito.
```

[Insertar nuevo sensor.](#)

[Postman:](#)

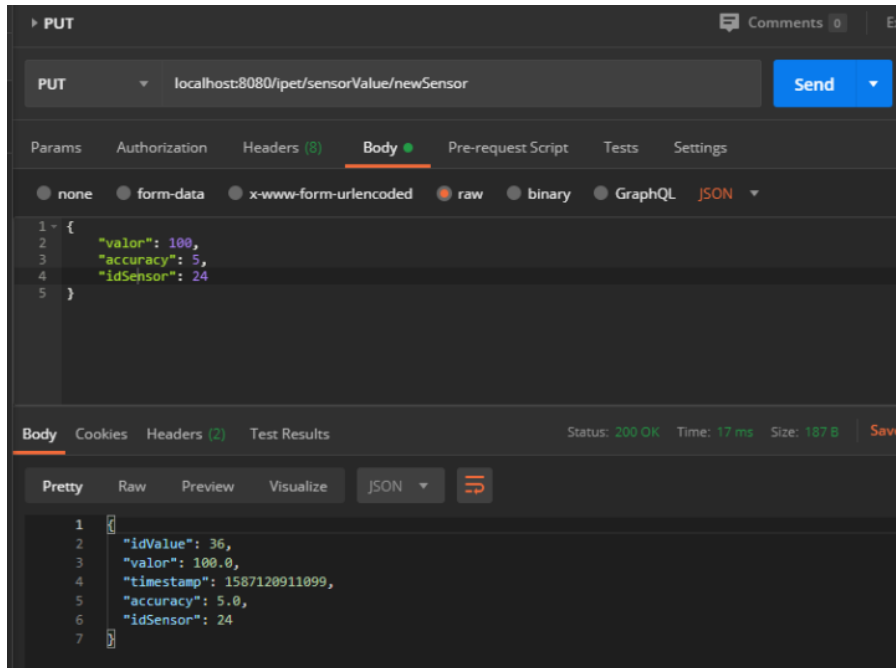


[Consola de Java:](#)

```
El sensor se ha añadido con éxito.
```

[Insertar nuevo valor del sensor.](#)

[Postman:](#)



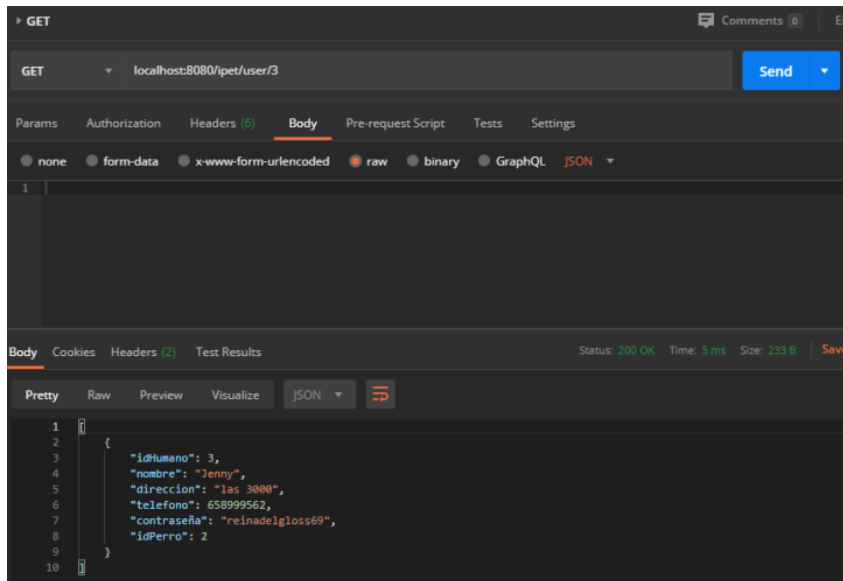
[Consola de Java:](#)

```
El valor del sensor se ha añadido con éxito.
```

GET

Obtener un usuario.

Postman:

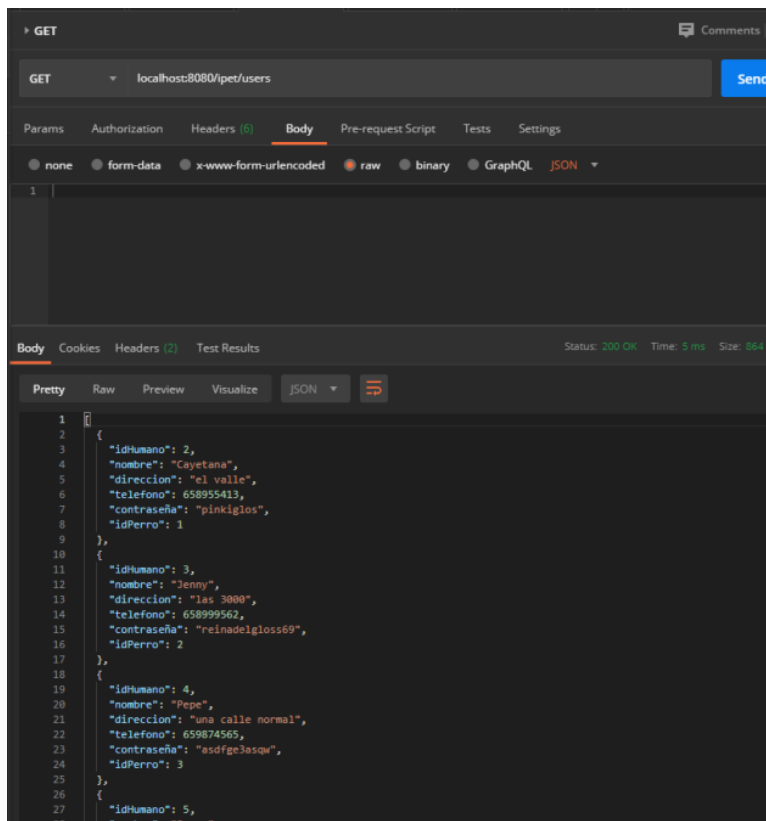


Consola de Java:

Se ha obtenido el usuario asociado al id. 3.

Obtener todos los usuarios.

Postman:

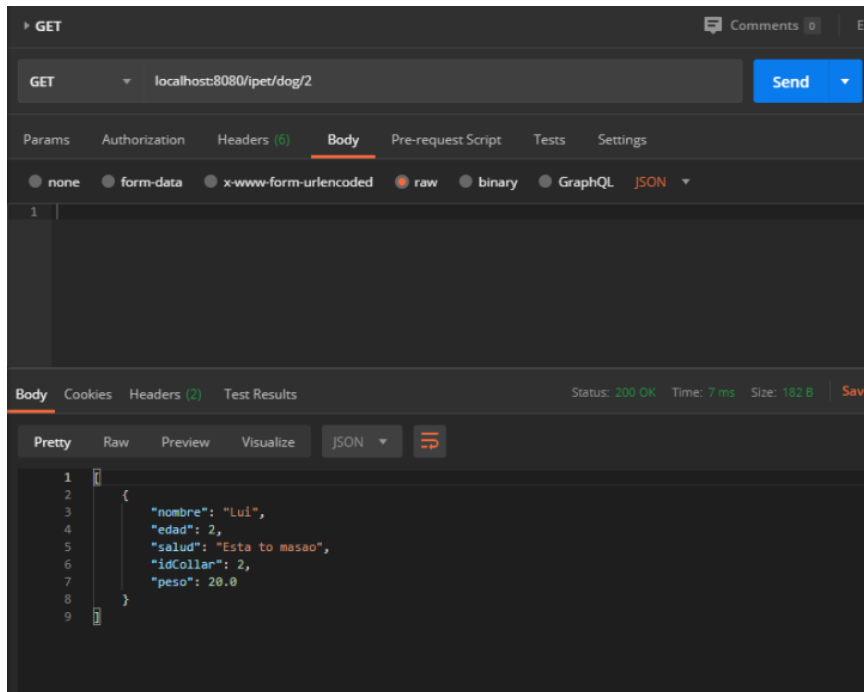


[Consola de Java:](#)

```
Se han obtenido un total de 5 usuarios.
```

[Obtener un perro.](#)

[Postman:](#)

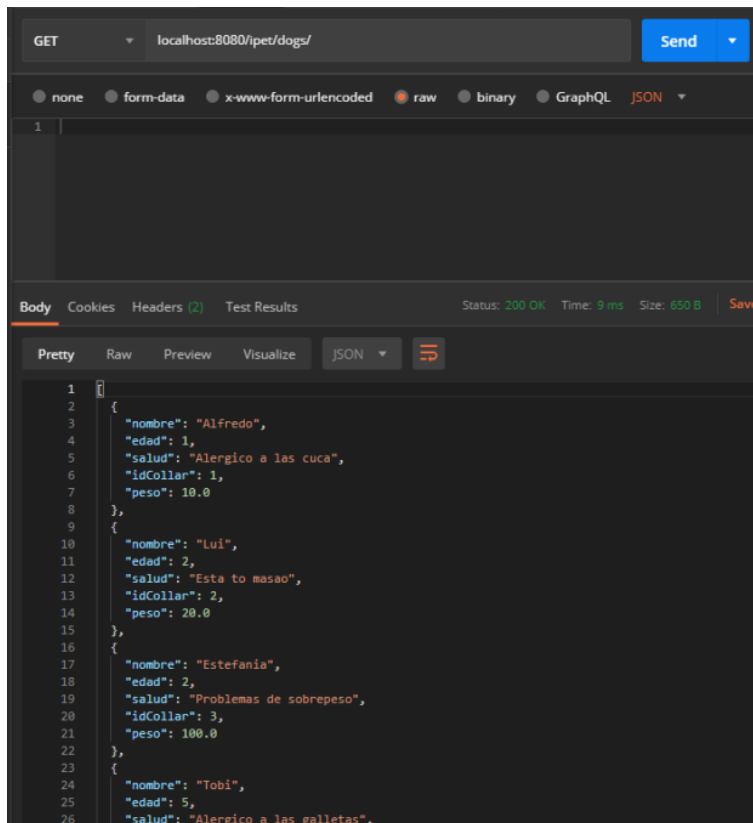


[Consola de Java:](#)

```
Se ha obtenido el perro asociado al id. 2.
```

Obtener todos los perros.

Postman:

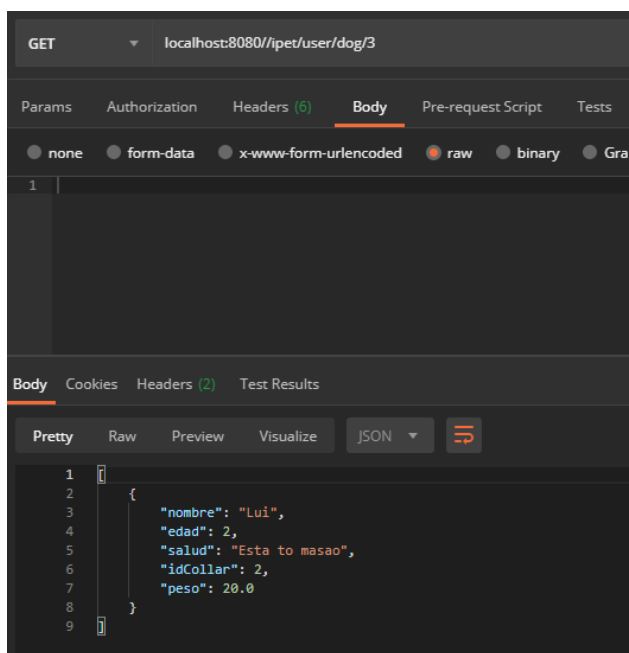


Consola de Java:

Se ha obtenido el perro asociado al id. 2.

Obtener el perro de un usuario.

Postman:

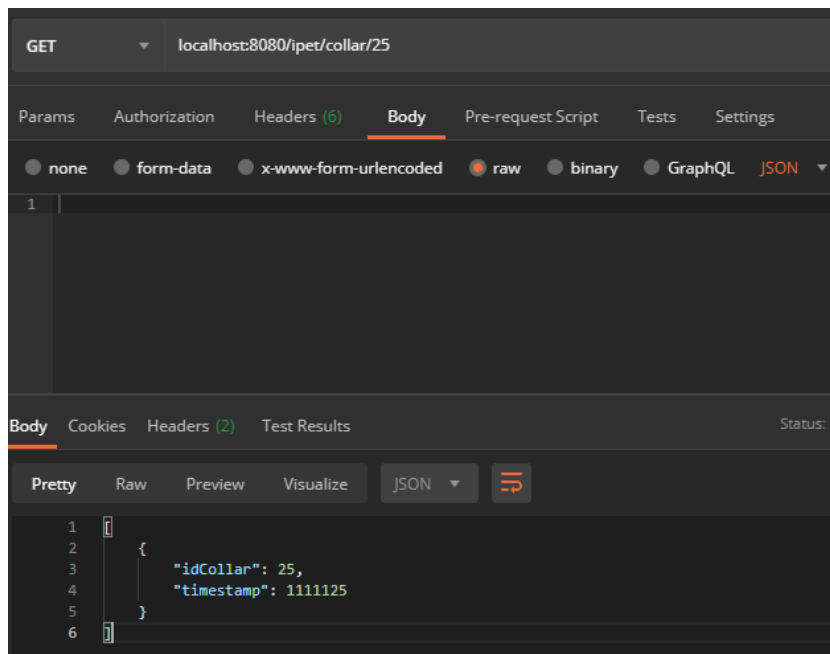


[Consola de Java:](#)

Se ha obtenido el perro asociado al id. del usuario 3.

[Obtener un collar.](#)

[Postman:](#)

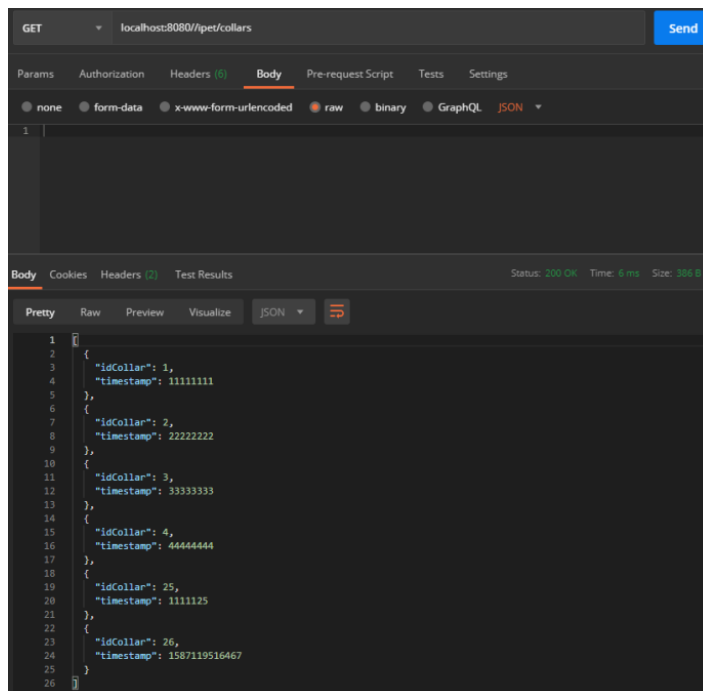


[Consola de Java:](#)

Se han obtenido un total de 6 collares.

[Obtener todos los collares.](#)

[Postman:](#)

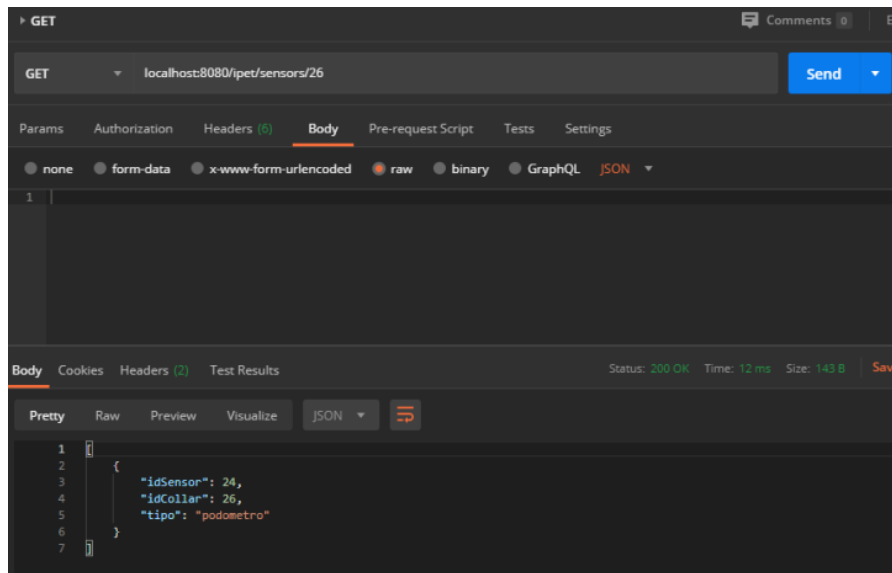


Consola de Java:

```
Se han obtenido un total de 6 collares.
```

Obtener todos sensores de un collar.

Postman:

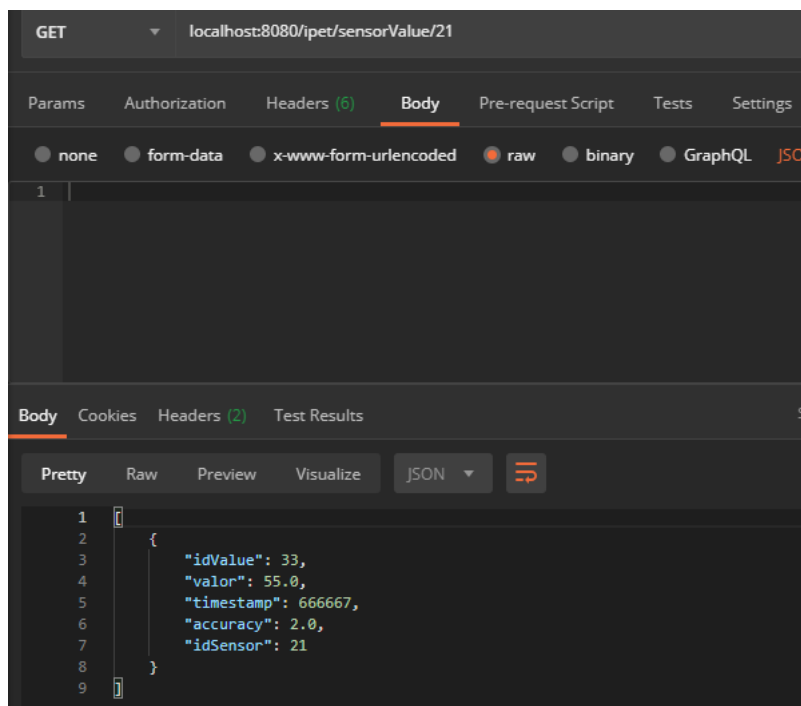


Consola de Java:

```
Se han obtenido un total de 1 sensores del collar con id. 26.
```

Obtener el valor del sensor.

Postman:

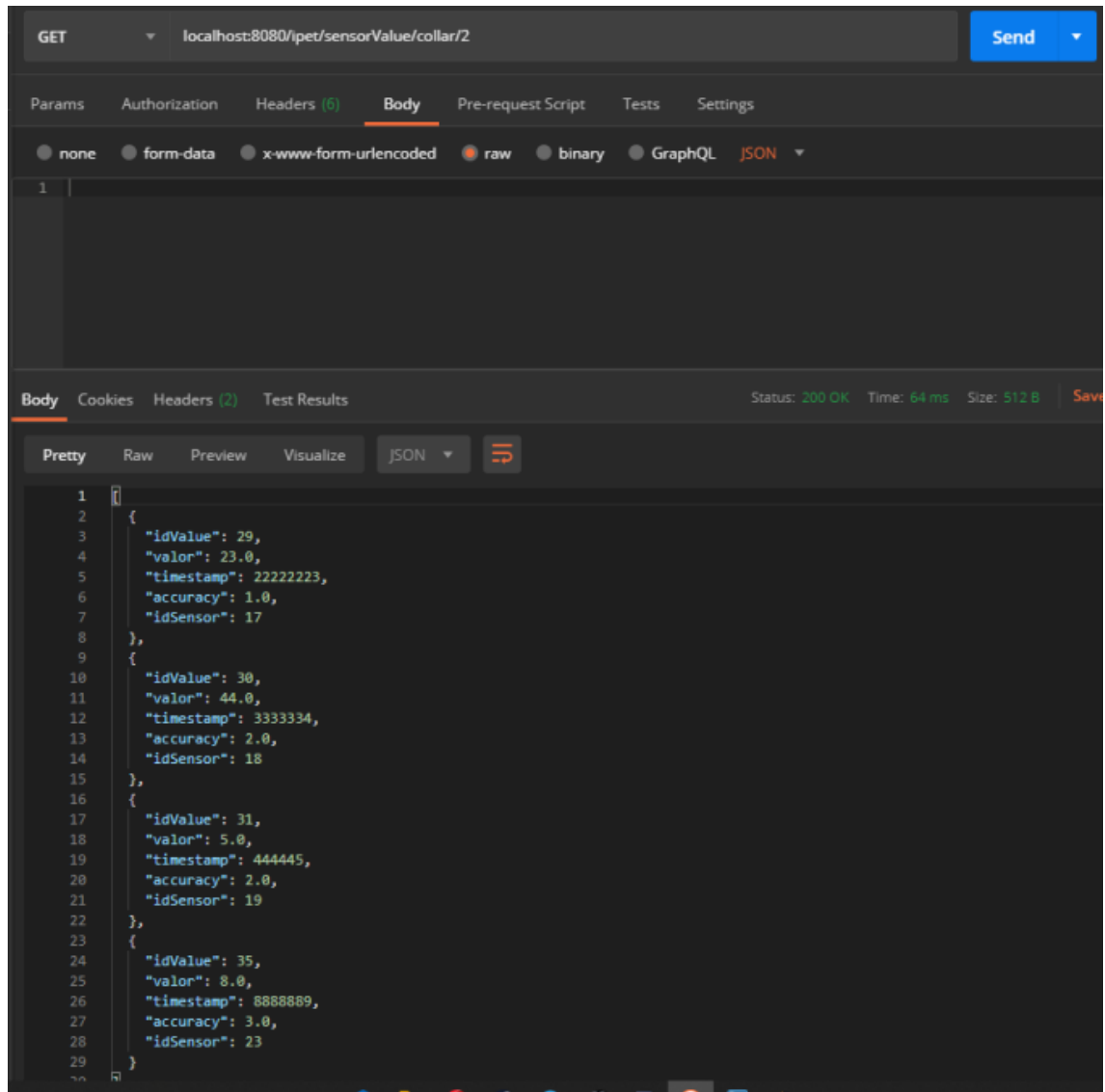


[Consola de Java:](#)

```
Se ha obtenido correctamente el valor del sensor.
```

[Obtener los valores de los sensores de un collar.](#)

[Postman:](#)



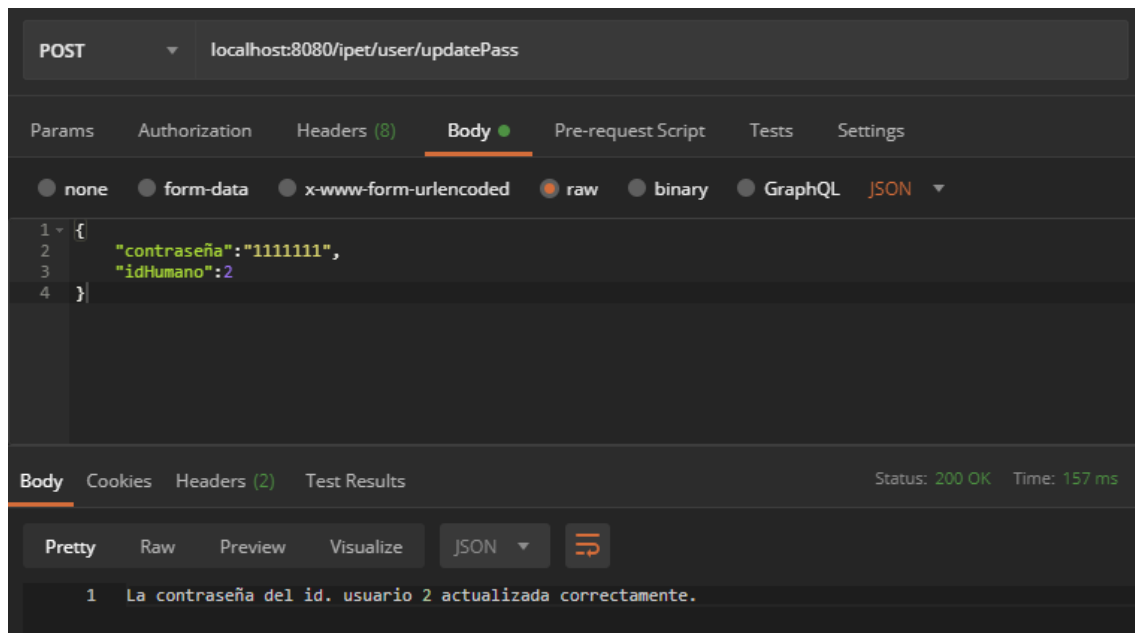
[Consola de Java:](#)

```
Se han obtenido los valores de los sensores del collar con id. 2.
```


POST

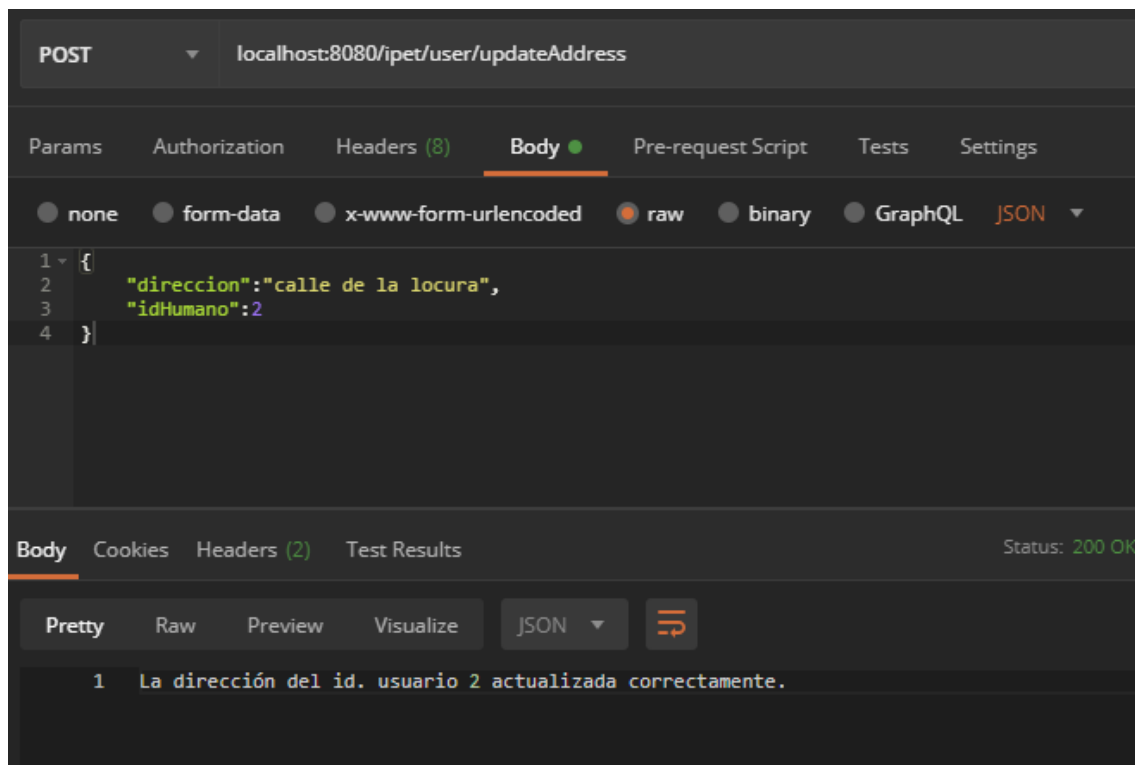
Actualizar contraseña de usuario.

Postman:



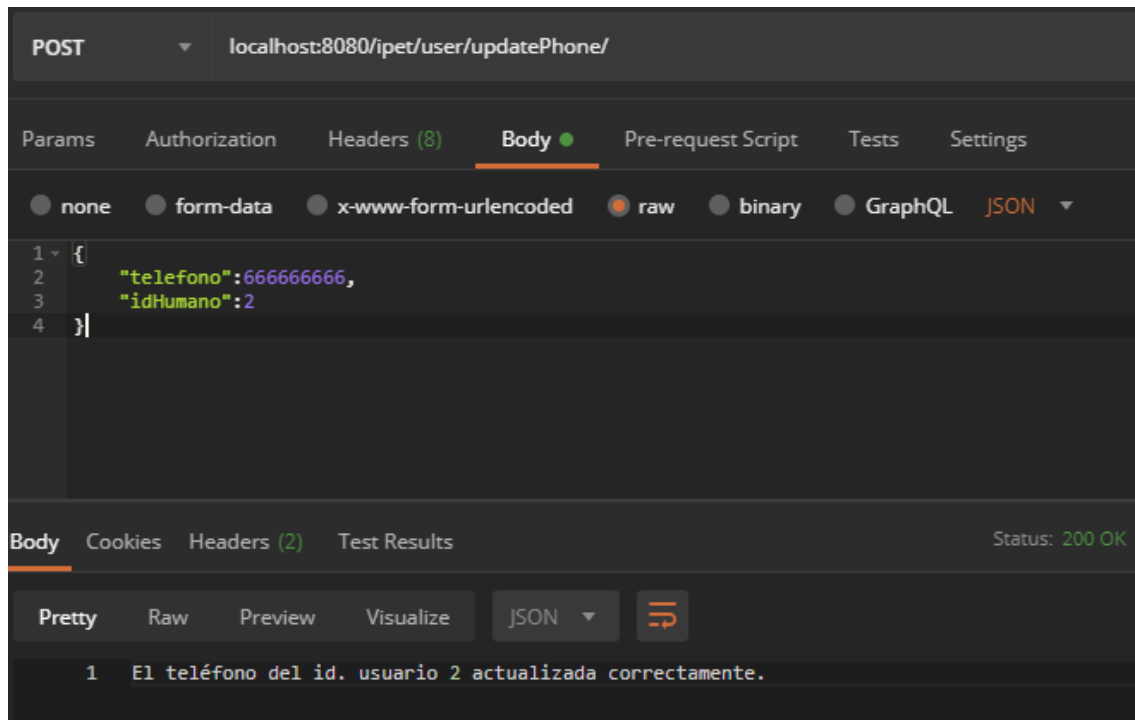
Actualizar dirección de usuario.

Postman:



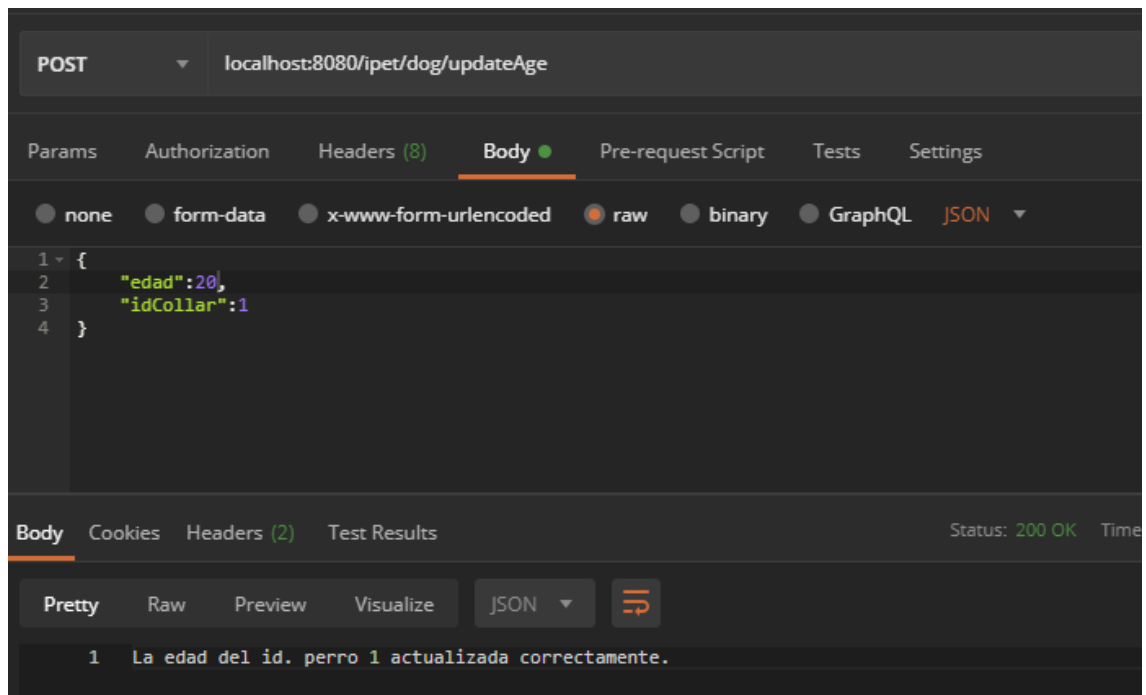
Actualizar teléfono de usuario.

Postman:



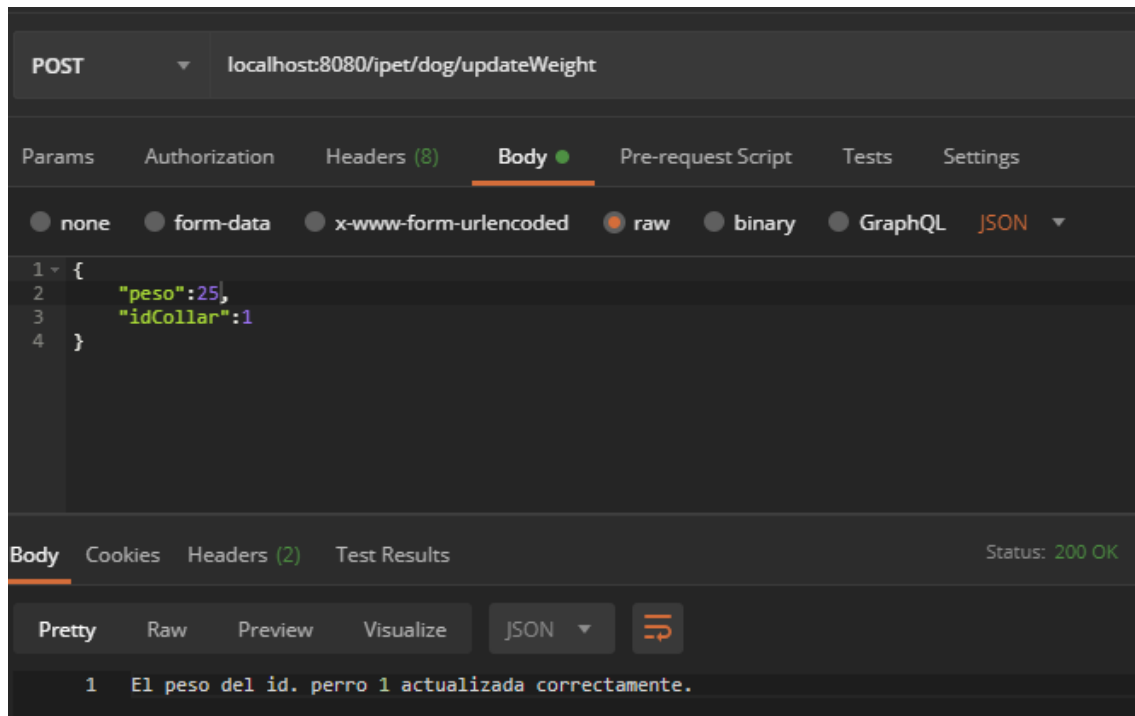
Actualizar edad de perro.

Postman:



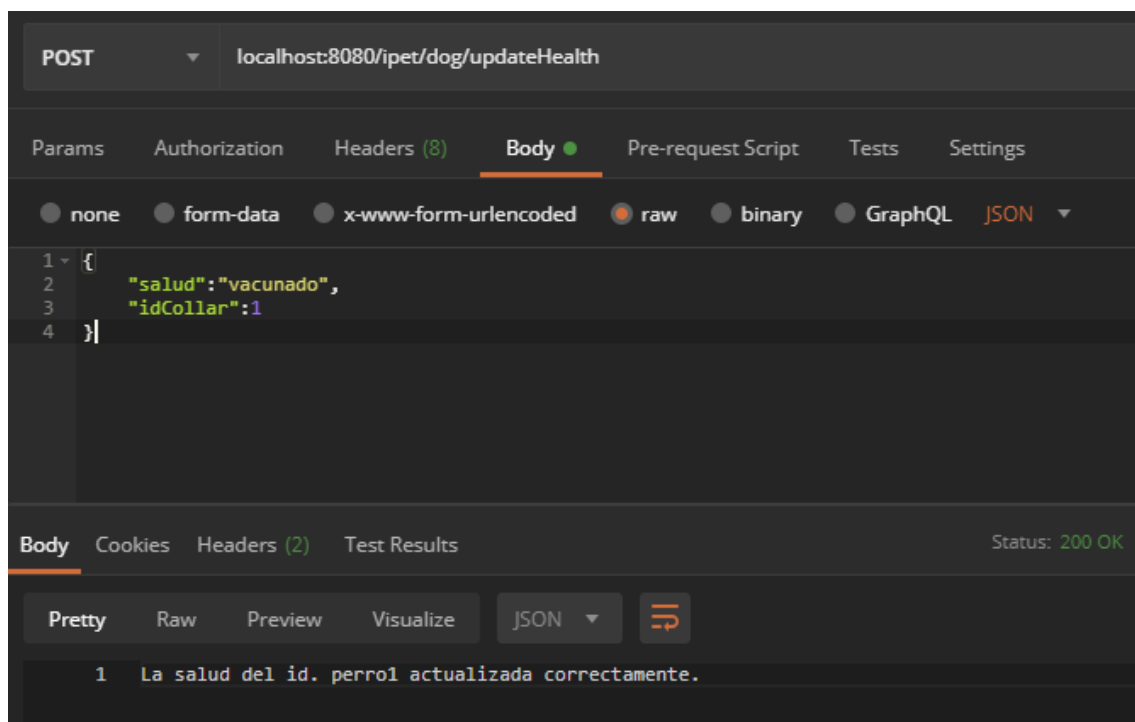
Actualizar peso de perro.

Postman:



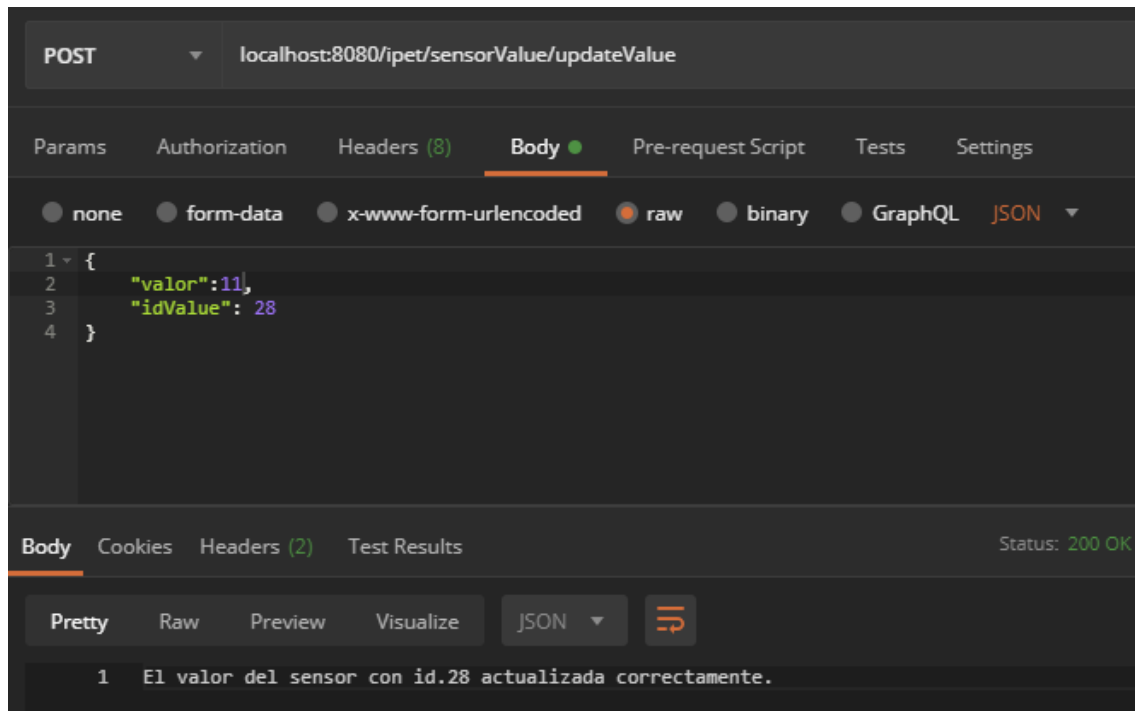
Actualizar salud de perro.

Postman:



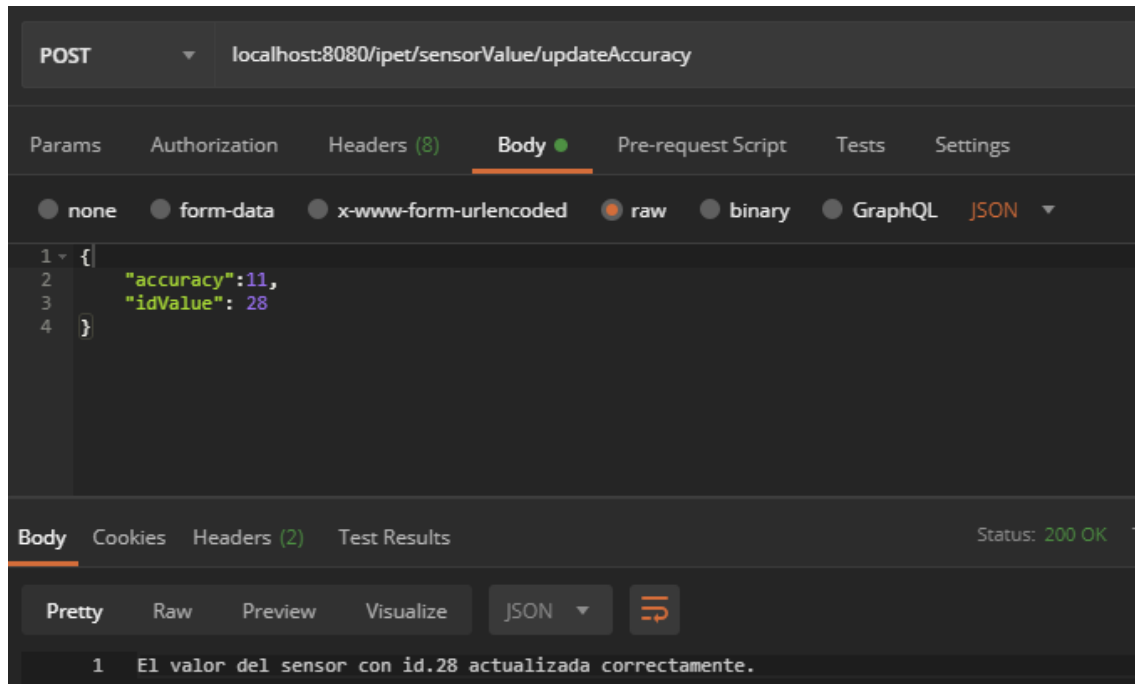
Actualizar valor de sensor.

Postman:



Actualizar precisión de sensor.

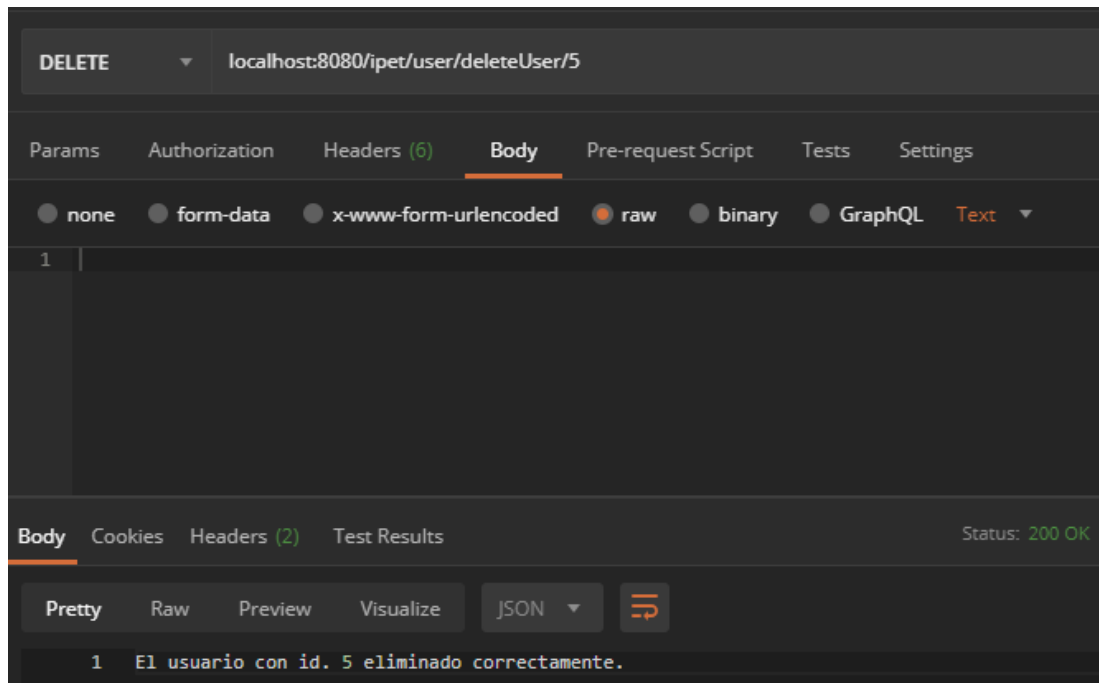
Postman:



DELETE

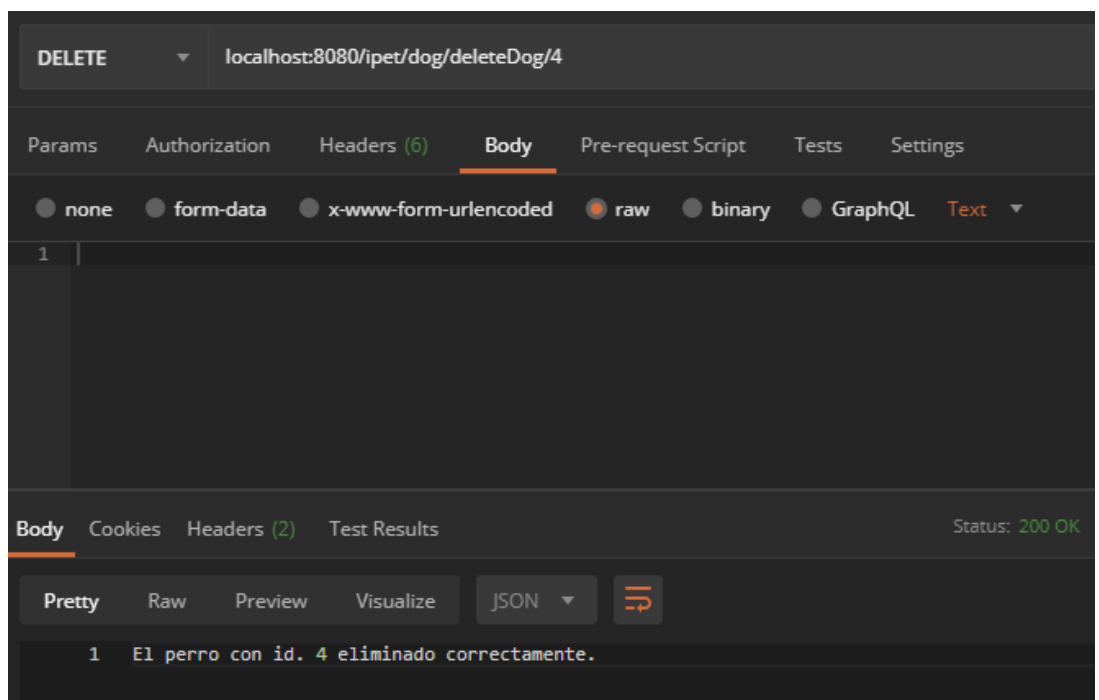
Eliminar usuario.

Postman:



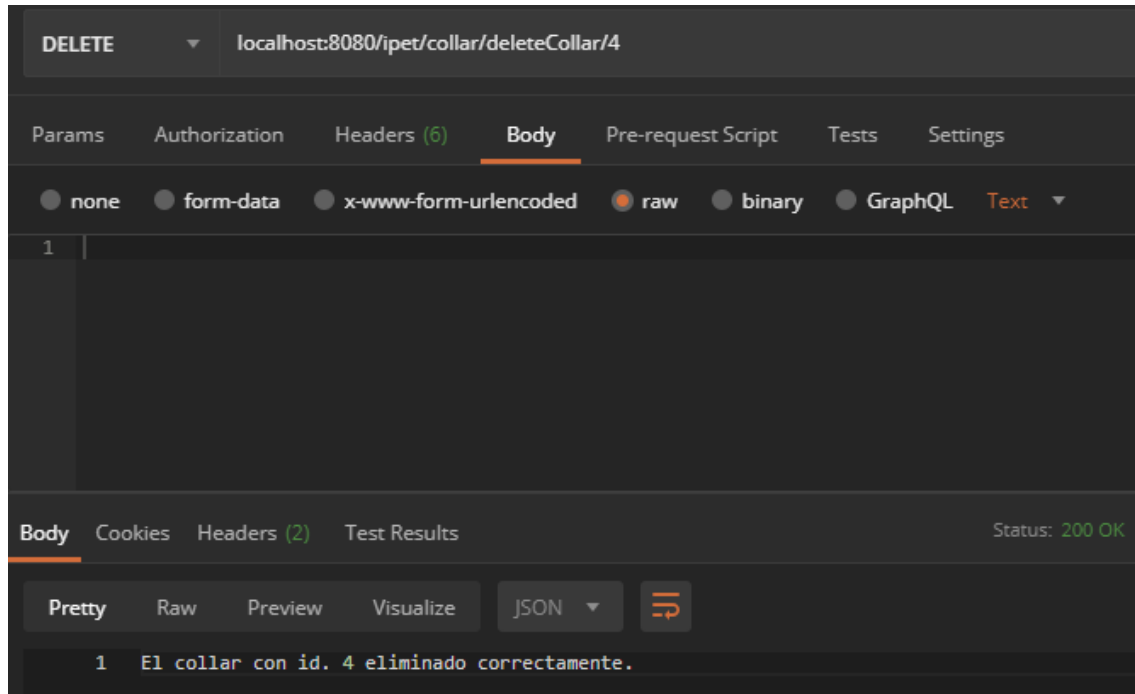
Eliminar perro.

Postman:



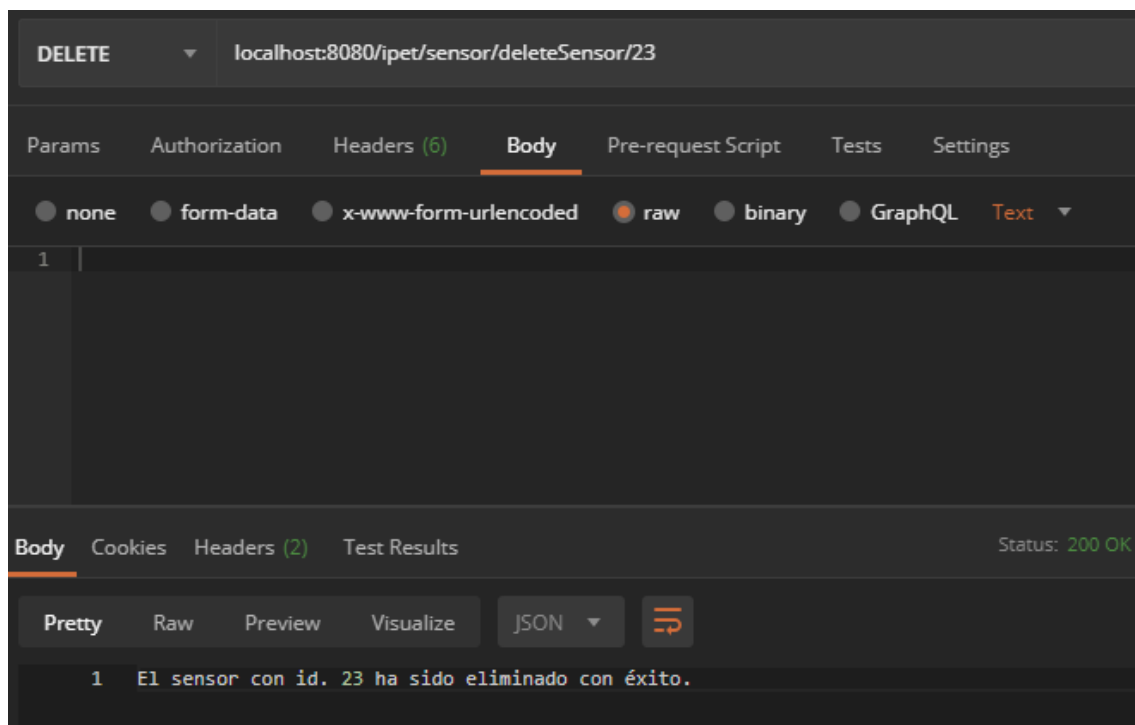
Eliminar collar.

Postman:



Eliminar sensor.

Postman:



Eliminar valor del sensor.

Postman:

