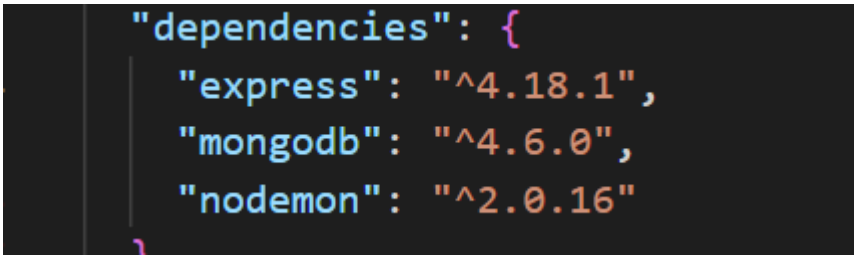
APUNTES NODE.JS - Abel Rios - 27/05/2022

Hoy vamos a crear un servidor en Node que conectaremos con una base de datos de MongoDB.

Creamos una nueva carpeta (ServerMongo), creamos un nuevo servidor como ya sabemos (npm init, npm i express, hacer el index.js, etc, etc) y una vez tenemos el servidor creado instalamos mongo con:

npm i mongodb

(si lo hemos instalado bien deberá aparecer en el package.json como una dependencia)



Ahora, en nuestro index vamos a poner:

```
let db = null;
let MongoClient = require("mongodb").MongoClient;
MongoClient.connect("mongodb://localhost:27017/", (err, client) => {
    if (err) throw err;
    db = client
    const PORT = 3001;
      app.listen(PORT, () => {
        console.log(`Server running on port ${PORT}`);
      });
});
```

De esta forma estamos creando ‘db’ que es nuestro conector y conectamos nuestro servidor a la base de datos desde el principio, para que no vaya conectando y desconectando en cada una de las distintas llamadas que realizaremos y así evitar sobrecargar el socket (número de entradas de conexiones).

Ejercicio:

Crear una base de datos en MongoDB con 3 o 4 personas que tendrán como propiedades:

- Nombre
- Apellido
- Edad

Y sobre ella hacer un Post, Get, Put y Delete.

— — — — — — — —

Primero crearemos una base de datos en Mongodb de prueba (en este caso se llama ‘basedeprueba’) y una colección (‘personas’) en la que añadiremos 4 personas distintas. Para ello he creado un archivo js con 4 objetos (personas) y lo he exportado como json e importado desde el Compass. (también se podría haber insertado desde la consola).

Ahora ya en nuestro archivo index podemos empezar a hacer las llamadas a la base de datos. Nuestro esqueleto para los endpoints será así:

```
17  // Esqueleto del endpoint con una query de MongoDB dentro
18
19  app.get("/", async function (request, response){
20
21      let database = client.db("basedeprueba");
22
23      await database.collection("personas").find().toArray((err, results) => {
24        if (err) throw err;
25        response.json(results);
26      });
27  });
28
29
```

pero, como nosotros al cliente (client) lo hemos guardado en la variable ‘db’ anteriormente, en lugar de client, debemos decir ‘db’.

- **Haciendo distintos GET**

```
17  // Haciendo un GET de menores de la edad que nos llegue por params
18
19  app.get("/menoresde/edad/:edad", async function (request, response){
20
21      let database = db.db("basedeprueba");
22
23      await database.collection("personas").find({ edad: { $lt: Number(request.params.edad) } })
24      // Hacemos el Number() porque lo que nos llega por params es un string
25      .toArray((err, results) => {
26        if (err) throw err;
27        response.json(results);
28      });
29
30  }); // FUNCIONA
```

```
33  // Haciendo un GET del nombre que nos llegue por params
34
35  app.get("/menoresde/nombre/:nombre", async function (request, response){
36
37      let database = db.db("basedeprueba");
38
39      await database.collection("personas").find({ nombre: { $eq: request.params.nombre } })
40      // Hacemos el Number() porque lo que nos llega por params es un string
41      .toArray((err, results) => {
42        if (err) throw err;
43        response.json(results);
44      });
45
46  }); // FUNCIONA
47
```

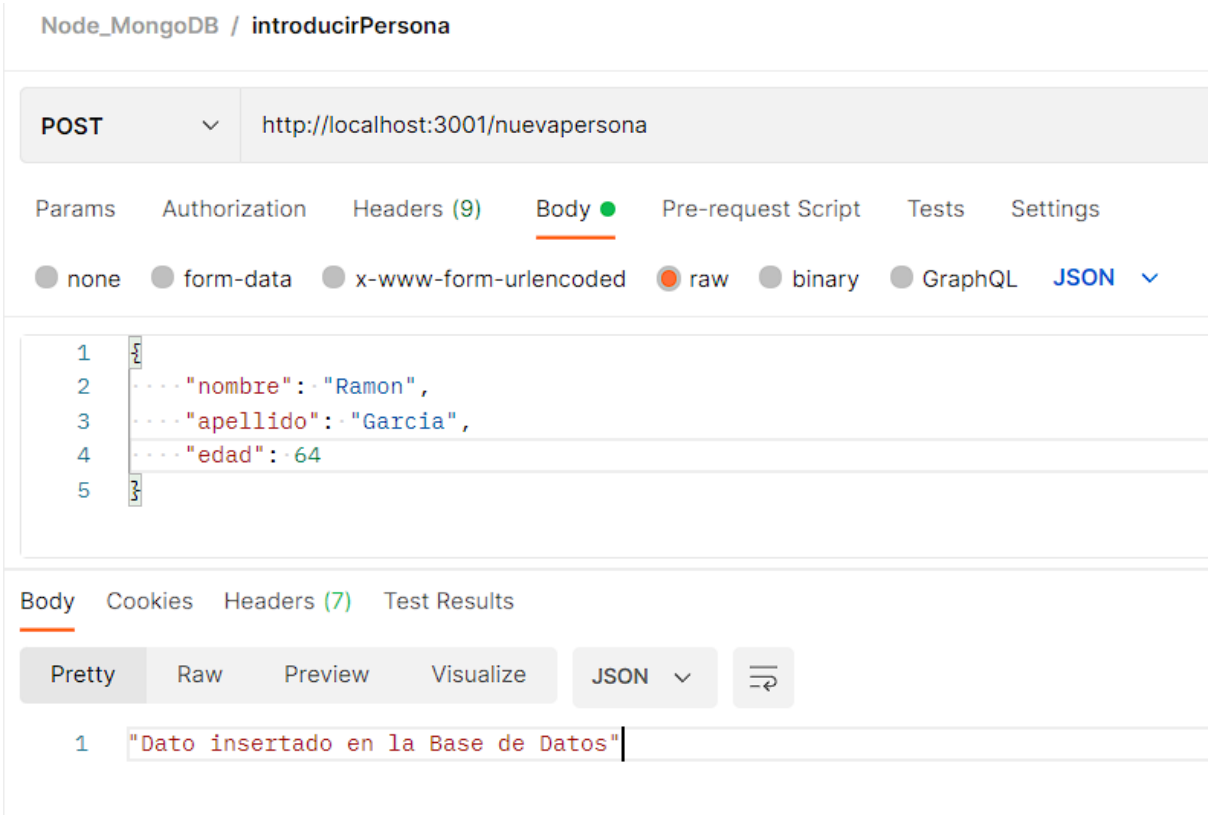
para estas cosas tenemos la documentación de Mongo con sus operadores y métodos:

https://www.mongodb.com/docs/manual/reference/operator/query/

- **Haciendo un POST (con Postman)**

```
// Haciendo un POST de una persona

app.post("/nuevapersona", async function (request, response){

    let database = db.db("basedeprueba");

    await database.collection("personas").insertOne(request.body);

    response.json("Dato insertado en la Base de Datos");
});
```

Postman:



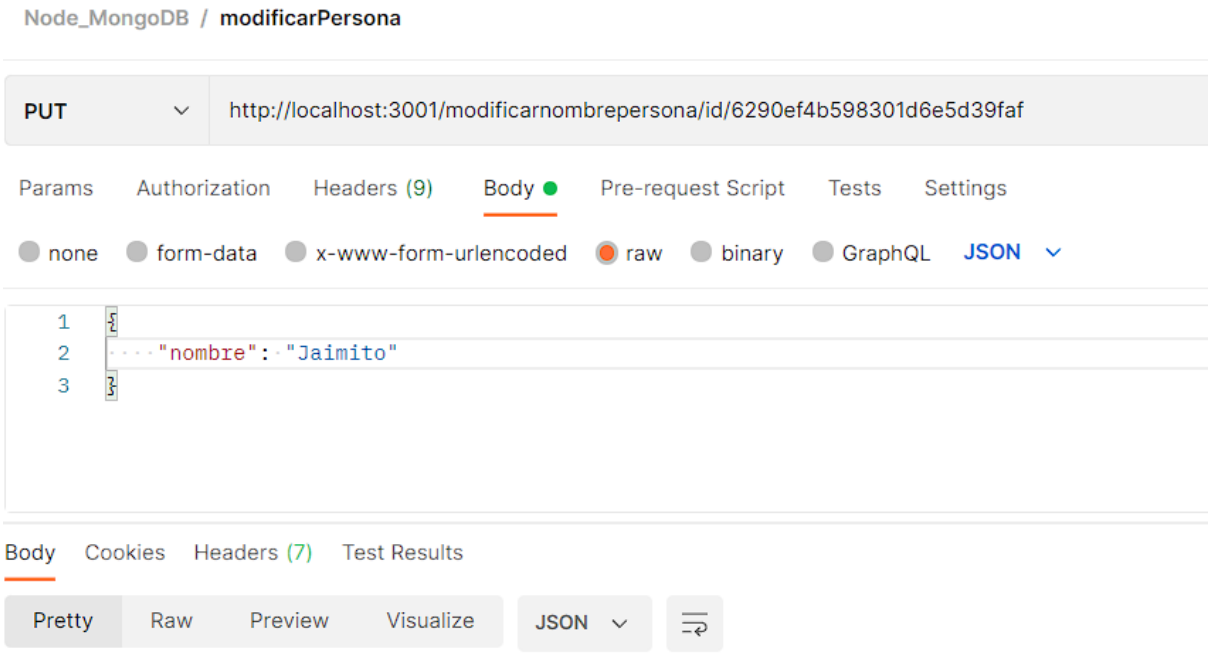
- **Haciendo un PUT (modificar con body y params)**

```
// Haciendo un PUT que modifique una persona existente (usando body+params)

app.put("/modificarnombrepersona/id/:id", async function (request, response){

    let database = db.db("basedeprueba");

    await database.collection("personas").updateOne(
        {_id:ObjectID(request.params.id)},
        {$set: {nombre: request.body.nombre}}
    );

    response.json("Dato modificado en la Base de Datos");

});

```

OJO: Para usar la función ObjectID tenemos que hacer un 'require' al principio del documento:

```
const { ObjectID } = require("mongodb");
```

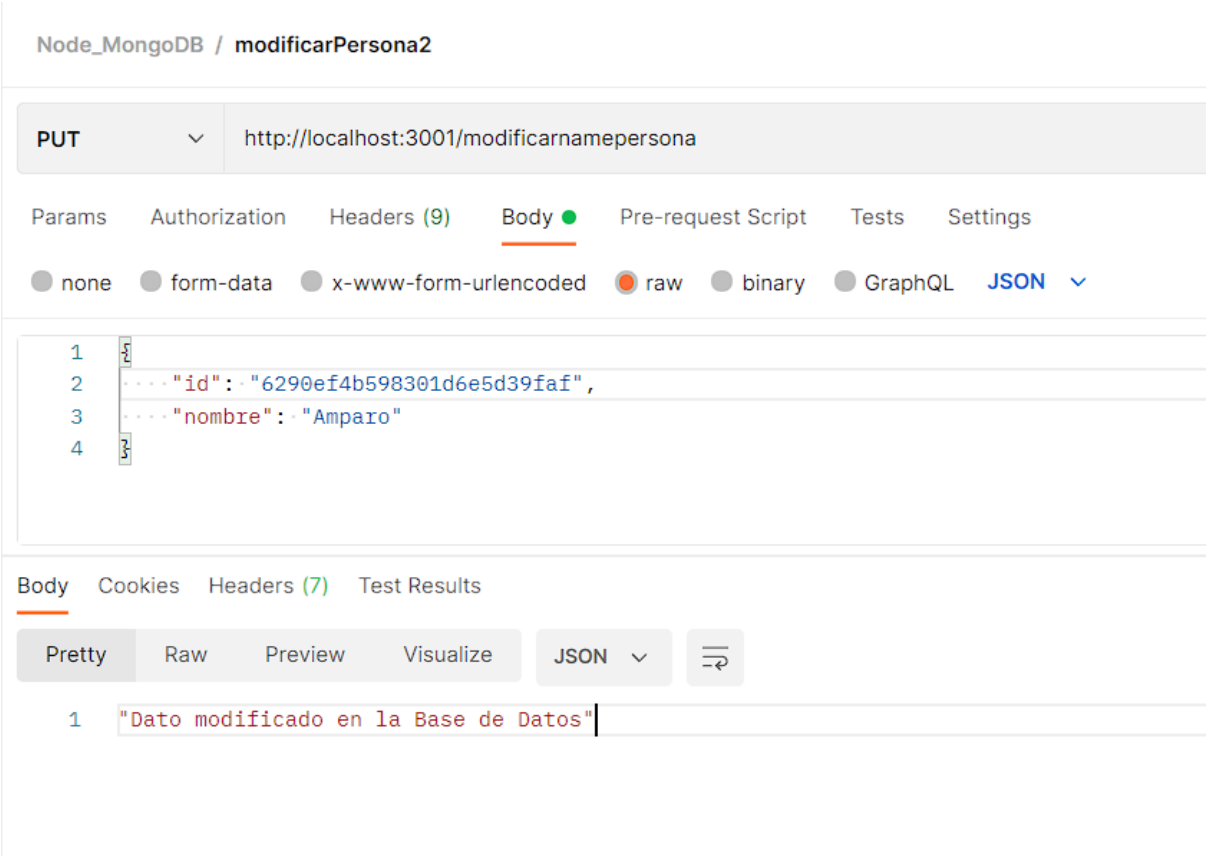
Postman:



- **Haciendo un PUT (modificar con body sólo)**

```
// Haciendo un PUT que modifique una persona existente (usando solo body)

app.put("/modificarnamepersona", async function (request, response){

    let database = db.db("basedeprueba");

    await database.collection("personas").updateOne(
        {_id: ObjectID(request.body.id)},
        {$set: {nombre: request.body.nombre}}
    );

    response.json("Dato modificado en la Base de Datos");

});
```

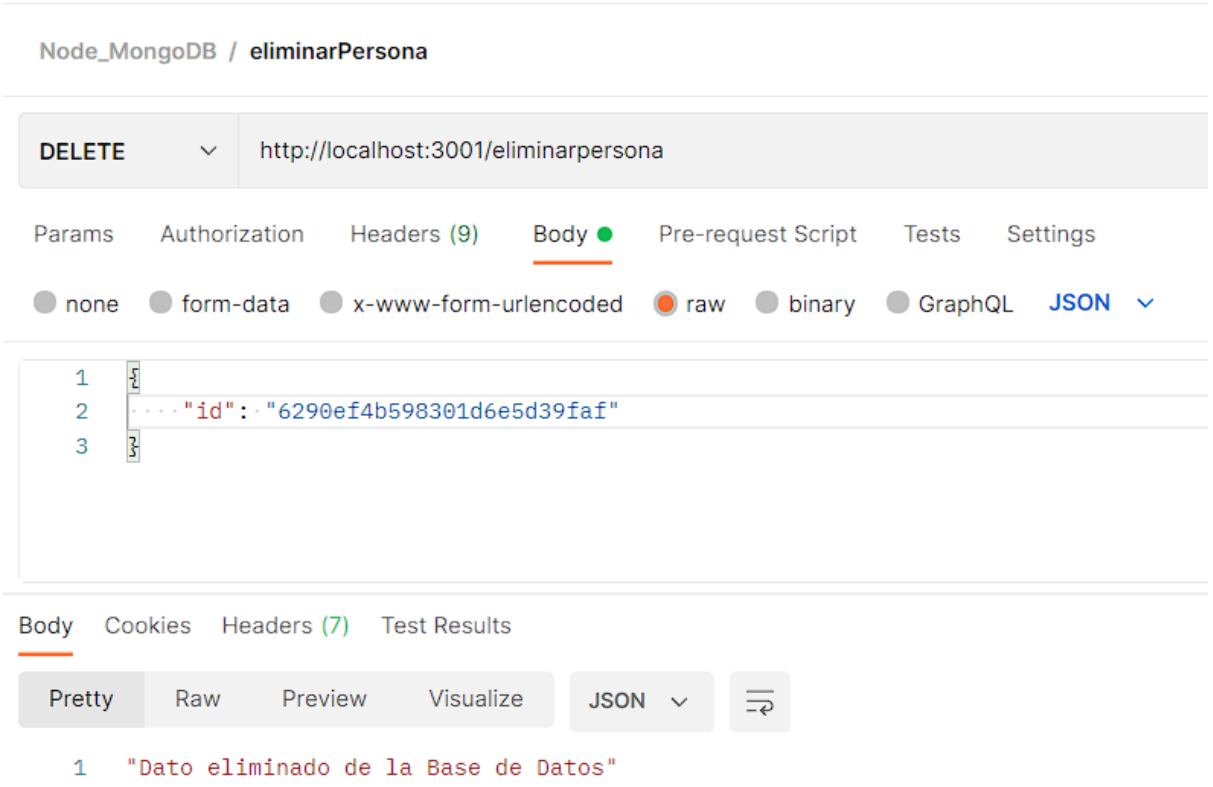
Postman:



- **Haciendo un Delete (con body)**

```
// Haciendo el Delete

app.delete("/eliminarpersona", async function (request, response){

    let database = db.db("basedeprueba");

    await database.collection("personas").deleteOne(
        {_id: ObjectID(request.body.id)}
        );

    response.json("Dato eliminado de la Base de Datos");

});

```

Postman:



Ahora vamos a hacer el mismo ejercicio que hicimos la semana pasada, con la colección “Alumnos” (guardar un array de objetos en formato JSON e importarlo a una nueva colección con el Compass)

1 - Sacar todos los alumnos que tienen 23 años

```
app.get("/edad/:edad", async function (request, response){

    let database = db.db("basedeprueba");

    await database.collection("alumnos").find({ edad: { $eq: Number(request.params.edad) } })
    // Hacemos el Number() porque lo que nos llega por params es un string
    .toArray((err, results) => {
      if (err) throw err;
      response.json(results);
    });

});
```

2 - Sacar aquellas personas que tienen el sexo femenino y no tienen 23 años

```
125  app.get("/sexoyedad/:sexo/:edad", async function (request, response){
126
127      let database = db.db("basedeprueba");
128
129      await database.collection("alumnos").find({
130          edad: { $ne: Number(request.params.edad) },
131          sexo: {$eq: request.params.sexo}
132      })
133      // Hacemos el Number() porque lo que nos llega por params es un string
134      .toArray((err, results) => {
135        if (err) throw err;
136        response.json(results);
137      });
138
139  });
```

3 - Introducir un alumno nuevo, si su DNI ya existe, entonces debe de sacar un mensaje de error y no crear el alumno
(revisar si funciona de verdad)

```
144  app.post("/introduciralumno", async function (request, response){
145
146      let result;
147
148      let database = db.db("basedeprueba");
149
150      let existe = await database.collection("alumnos").findOne({
151          dni: { $eq: request.body.dni },
152      });
153
154      if(!existe){
155          await database.collection("alumnos").insertOne(request.body);
156          result="Dato insertado en la Base de Datos";
157      } else {
158          result="No se ha podido insertar."
159      }
160      response.json(result);
161  });
```