

PROGRAMACIÓN 1º DAW 2022-23

Objetivo:

Realizar una **API Rest para el CRUD** de una base de datos Mongo usando Mongoose y Express.js

Requisitos:

Mínimos (5p):

Operaciones CRUD de una colección.

Creación de un Schema con variedad de tipos campos: String, Number, Date, Boolean, Arrays.

Ampliaciones (hasta 10p):

- Otras operaciones: Por ejemplo: Filtro de búsqueda (1p)
Filtro por marca en las fundas
- Otra colección: (2p)
Cristales
- Relaciones entre colecciones: (1p)
- Frontend web (2p)
- Otras: Control y Gestión de errores (1p) , confirmación de borrado (1p).

Entrega:

Archivos de la API

Lista de endpoints creados con capturas de pantalla demostrando que funcionan.

Usar Postman o extensiones de Visual Studio.

Daniela Aragón Risso

Capturas Funcionamiento Schema

1.- Schema

*/

```
const fundaSchema = new mongoose.Schema({
  marca: String,
  modelo: String,
  cantidad: Decimal128,
  precio: Decimal128,
  fecha_envio: Date,
  descripcion_producto: String,
  cristal: String,
  seguro: Boolean,
  add: Array
});
```

```
const cristalesSchema = new mongoose.Schema({
  tipo: String,
  descripcion: String,
  garantia: Boolean,
});
```

Para buscar en la base de datos mediante GET

```
//GET fundas/marca : Listado de fundas por nombre
app.get('/fundas/:marca?', async function (req, res) {
  let marca = req.params.marca || '';
  let fundas;

  if (marca === '') {
    fundas = await funda.find();
  } else {
    fundas = await funda.find({ marca: marca });
  }

  res.send(fundas);
});

//GET /cristales : Listado de cristales
app.get('/cristales', async function (req, res) {
  let cristal;

  cristal = await cristales.find();

  res.send(cristal);
});
```

Daniela Aragón Risso

The first screenshot shows a GET request to `http://localhost:3000/fundas`. The response is a JSON array of three objects, each representing a fundas item with fields like `_id`, `marca`, `modelo`, `cantidad`, `precio`, `descripcion_producto`, `seguro`, `fecha_envio`, `add`, and `cristal`.

The second screenshot shows a GET request to `http://localhost:3000/cristales`. The response is a JSON array of three objects, each representing a cristal item with fields like `_id`, `tipo`, `descripcion`, `garantia`, and `__v`.

Para inserta mediante POST:

The screenshot shows a POST request to `http://localhost:3000/insertafundas`. The request body is form-encoded with the following data:

Field	Value
marca	prueba100
modelo	prueba100
cantidad	100
name	value

The response is a 200 OK status with a message: `646bcf371e0349d5c3dbc696 a sido insertado`.

```
//POST insertafundas

app.post('/insertafundas', async function (req, res) {

  //Recoge datos formulario

  let marca = req.body.marca;
  let modelo = req.body.modelo;
  let cantidad = req.body.cantidad;
  let precio = req.body.precio;
  let fecha_envio = req.body.fecha_envio;
  let descripcion_producto = req.body.descripcion_producto;
  let cristal = req.body.cristal;
  let seguro = req.body.seguro;
  let add = req.body.add;

  //crea una instancia de usuarios con esos datos

  let nuevafunda = new funda(
    {
      marca: marca,
      modelo: modelo,
      cantidad: cantidad,
      precio: precio,
      fecha_envio: fecha_envio,
      descripcion_producto: descripcion_producto,
      cristal: cristal,
      seguro: seguro,
      add: add
    }
  );
```

```
    //hace un 'save' de esa instancia

    await nuevafunda.save();

    //informa al cliente

    res.send(nuevafunda.id+' a sido insertado');

  });

  //POST insertacristales

  app.post('/insertacristal', async function (req, res) {

    //Recoge datos formulario

    let tipo = req.body.tipo;
    let descripcion = req.body.descripcion;
    let garantia = req.body.garantia;

    //crea una instancia de usuarios con esos datos

    let nuevocristal = new cristales(
      {
        tipo: tipo,
        descripcion: descripcion,
        garantia: garantia
      }
    );
```

GET ⌵ http://localhost:3000/fundad Send

Query

Headers ²

Auth

Body

Tests

Pre Run

Query Parameters

☐ parameter

value

Status: 200 OK Size: 1.3 KB Time: 507 ms

Response

Headers ⁶

Cookies

Results

Docs

{}

⋮

59

"prueba2"

60

],

61

"_v": 0

62

},

63

{

64

"_id": "646bccf21e0349d5c3dbc690",

65

"marca": "prueba3",

66

"modelo": "prueba3",

67

"cantidad": {

68

| "\$numberDecimal": "10"

69

},

70

"add": [],

71

"_v": 0

72

},

73

{

74

"_id": "646bcf371e0349d5c3dbc696",

75

"marca": "prueba100",

76

"modelo": "prueba100",

77

"cantidad": {

78

| "\$numberDecimal": "100"

79

},

80

"add": [],

81

"_v": 0

82

}

83

}

POST ⌵ http://localhost:3000/insertacristal Send

Query

Headers ²

Auth

Body ¹

Tests

Pre Run

JSON XML Text Form Form-encode GraphQL Binary

Form Encoded

☒ tipo

prueba200

☒ descripcion

prueba200

☒ garantia

false

Status: 200 OK Size: 41 Bytes Time: 123 ms

Response

Headers ⁶

Cookies

Results

Docs

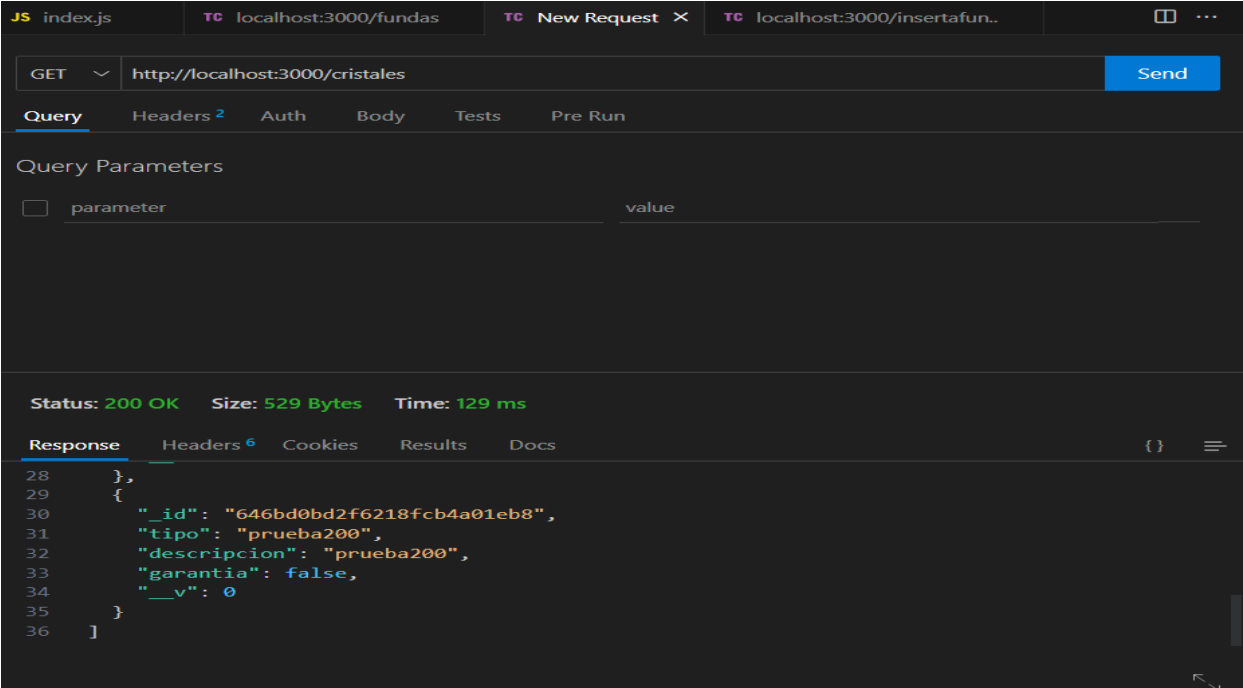
{}

⋮

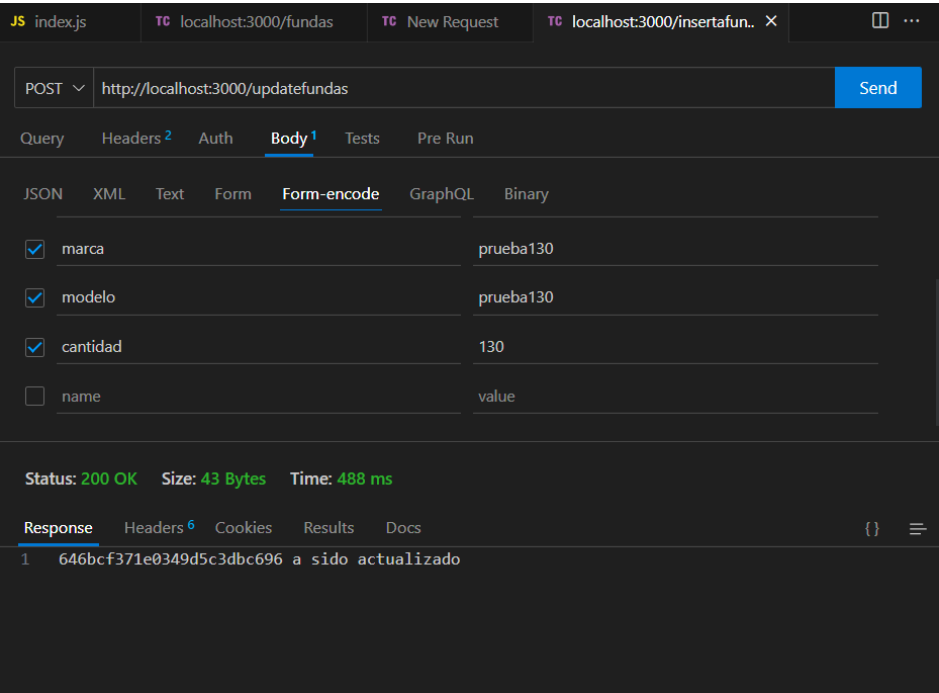
1

646bd0bd2f6218fcb4a01eb8 a sido insertado

Daniela Aragón Risso



Update mediante POST:



```
// update == POST
app.post('/updatefundas', async function (req, res) {
  //coger la id

  let id = req.body.id;

  //coger los datos

  let marca = req.body.marca;
  let modelo = req.body.modelo;
  let cantidad = req.body.cantidad;
  let precio = req.body.precio;
  let fecha_envio = req.body.fecha_envio;
  let descripcion_producto = req.body.descripcion;
  let cristal = req.body.cristal;
  let seguro = req.body.seguro;
  let add = req.body.add;

  //fundasfindByIdAndUpdate
  let id: any

  let fundas = await funda.findByIdAndUpdate(id,

    {
      marca: marca,
      modelo: modelo,
      cantidad: cantidad,
      precio: precio,
      fecha_envio: fecha_envio,
      descripcion_producto: descripcion_producto,
      cristal: cristal,
      seguro: seguro,
      add: add
```

```
      seguro: seguro,
      add: add
    }
  );

  // responder al cliente

  res.send(fundas.id+' a sido actualizado');
});

// update == POST
app.post('/updatecristal', async function (req, res) {
  //coger la id

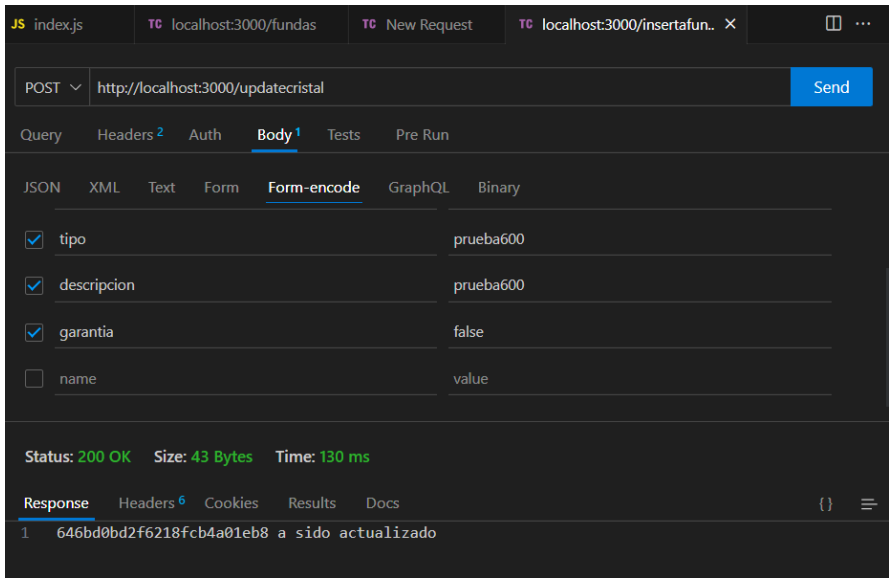
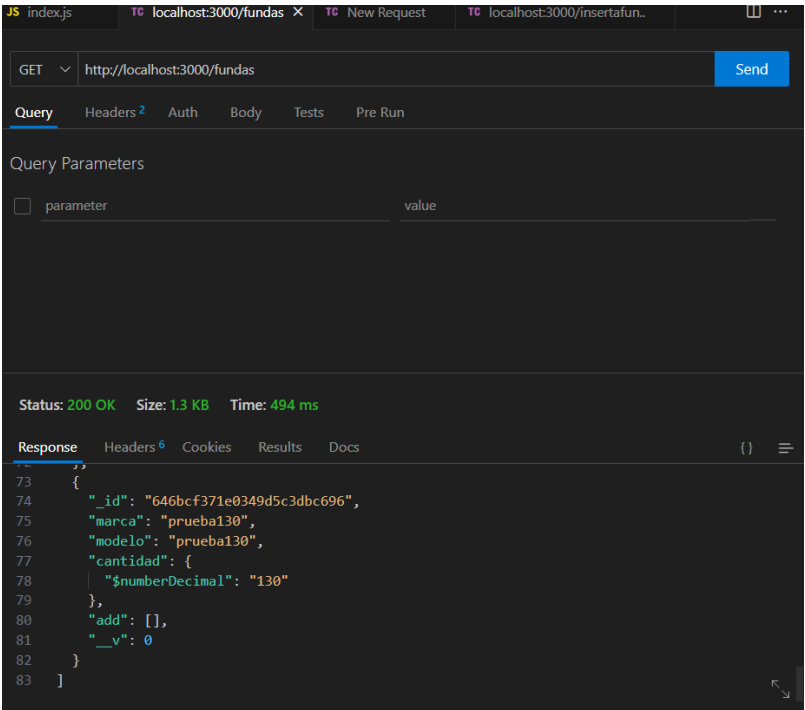
  let id = req.body.id;

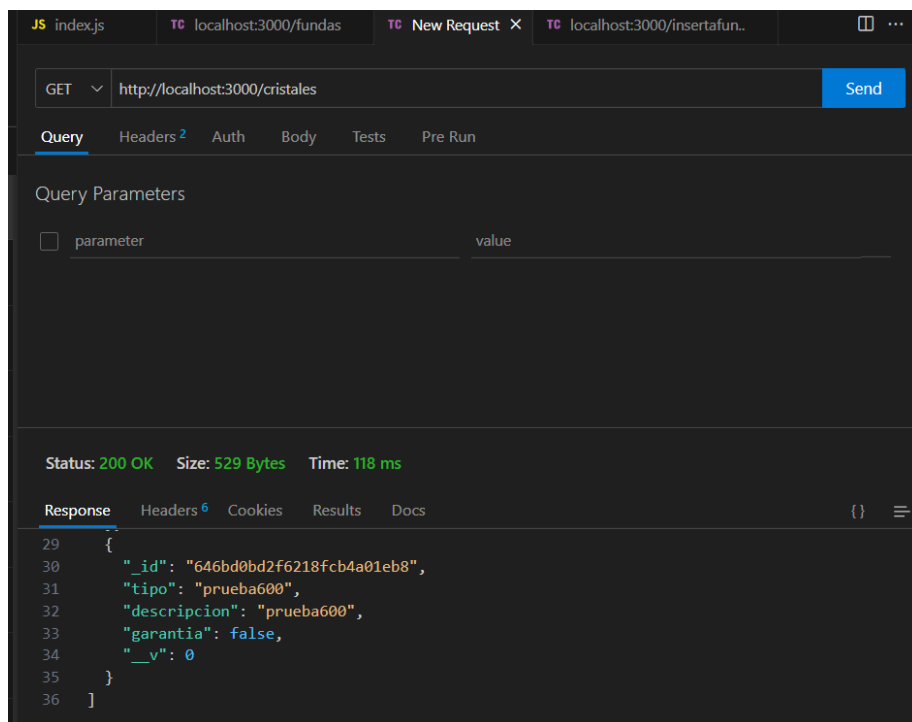
  //coger los datos

  let tipo = req.body.tipo;
  let descripcion = req.body.descripcion;
  let garantia = req.body.garantia;

  //CristalesfindByIdAndUpdate

  let cristal = await cristales.findByIdAndUpdate(id,
```





El borrado mediante DELETE:

```
//DELETE (GET) borrafundas/:id
app.delete('/:id', async function (req, res) {

  let id = req.params.id;

  let borrafundas = await funda.findByIdAndDelete(id);

  res.send(borrafundas.id+' a sido borrado');

});

//DELETE (GET) borracristal/:id

app.delete('/:id', async function (req, res) {

  let id = req.params.id;

  let borracristal = await cristales.findByIdAndDelete(id);

  res.send(borracristal.id+' a sido borrado');

});
```

Daniela Aragón Risso

