

Miriam Guerra Guerra

Tarea NODE.

Conexión y esquema tienda (pop up) y cliente:

```
op_upjs / main
//conectamos

const mongoose = require('mongoose'); //paquete mongoose
const express = require('express');

const app = express(); //creamos la app
const port = 3000;
//arranca el servidor
app.listen(port, () => {console.log(`puerto ${port}`)}});

app.use(express.urlencoded({ extended: true}));
// app.use(express.static('htdocs'));

main();

async function main(){
  await mongoose.connect ('mongodb+srv://usermm:zLj0Ux2lCmJYeKS1@majadabae.egiyan.mongodb.
    net/popup?retryWrites=true&w=majority&appName=MajadaBAE')
  console.log("conectado");

  // Datos de las tablas

  const popupSchema = new mongoose.Schema({
    nombre: String,
    precio: Number,
    tipoPago: String,
    fecha: Date,
    alquilado: Boolean,
    lugar: String,
    tipo: Array
  });

  const clienteSchema = new mongoose.Schema({
    nombre: String,
    apellidos: String,
    tarifa: Array, //tipo de suscripción, para saber la confianza en el cliente
    contacto: String
  });
```

Tablas y guardado de datos:

```

// creamos la colección de las tiendas
const Tienda = mongoose.model('popups', popupSchema);

// creamos la colección de los clientes

const Cliente = mongoose.model('clientes', clienteSchema);

// metemos los datos en las tablas

const latienda =
  new Tienda ({
    nombre: 'Atletico de altura',
    precio: 500,
    tipoPago: 'mensual',
    fecha: new Date ('2024-08-16'),
    alquilado: true,
    lugar: 'C/Toreto, 15',
    tipo: ["edificio con vistas", "planta baja", "almacén", "camión"]
  });

> /* ...

const losclientes =
  new Cliente({
    nombre: 'Lorenzo',
    apellidos: 'Bermudez Moreno',
    tarifa: [
      { tipo: 'premium', constancia: 'ininterrumpida' }
    ],
    contacto: 'lorenzo@gmail.com'
  });

// guardamos los datos

await latienda.save();
await losclientes.save();
console.log(latienda.nombre);
console.log(losclientes.nombre);

```

Find TIENDA:

```

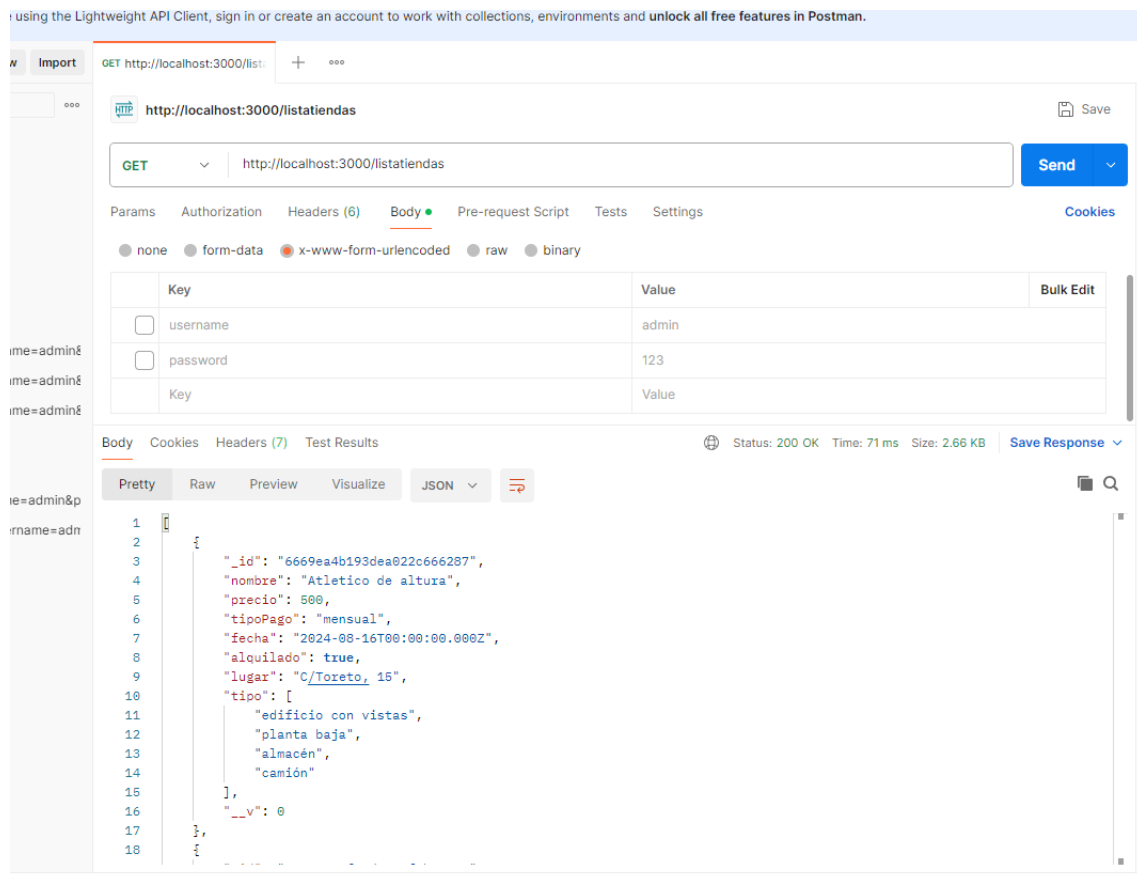
//FIND/SELECT

app.get('/listatiendas', async (req,res)=>{
  let shop = await Tienda.find();
  res.send(shop);
})

const tiendita = await Tienda.find();
console.log(tiendita);

```

Resultado find:



delete TIENDA:

```
app.get('/borrartienda/:id', async (req,res)=>{
  let borrashop = await Tienda.findByIdAndDelete(req.params.id);
  res.send(borrashop);
})
```

Resultado delete:

GET http://localhost:3000/bor + ...

HTTP http://localhost:3000/borrtienda/6669edc6a8d8671297e4e677 Save

GET http://localhost:3000/borrtienda/6669edc6a8d8671297e4e677 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary

Key	Value	Bulk Edit
<input type="checkbox"/> username	admin	
<input type="checkbox"/> password	123	
Key	Value	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 73 ms Size: 483 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "6669edc6a8d8671297e4e677",
3   "nombre": "Atletico de altura",
4   "precio": 500,
5   "tipoPago": "mensual",
6   "fecha": "2024-08-16T00:00:00.000Z",
7   "alquilado": true,
8   "lugar": "C/Toreto, 15",
9   "tipo": [
10    "edificio con vistas",
11    "planta baja",
12    "almacén",
13    "camión"
14  ],
15   "__v": 0
16 }
```

Insert TIENDA:

```

167
168 // insertar
169
170 // datos del formulario
171 app.post('/insertatienda', async (req,res)=>{
172   let nombre = req.body.nombre;
173   let precio = req.body.precio;
174   let tipoPago = req.body.tipoPago;
175   let fecha = req.body.fecha;
176   let alquilado = req.body.alquilado;
177   let lugar = req.body.lugar;
178   let tipo = req.body.tipo;
179   // crea la nueva tienda
180   const nuevaTienda =
181     new Tienda ({
182       nombre: nombre,
183       precio: precio,
184       tipoPago: tipoPago,
185       fecha: fecha,
186       alquilado: alquilado,
187       lugar: lugar,
188       tipo: tipo
189     });
190   // guarda los datos
191   await nuevaTienda.save();
192   // muestra que todo salió bien
193   res.send(nuevaTienda.id+' se ha añadido correctamente');
194
195
196   ;
197

```

Resultado insert:

HTTP http://localhost:3000/insertatienda Save

POST http://localhost:3000/insertatienda Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

<input checked="" type="checkbox"/>	alquilado	false
<input checked="" type="checkbox"/>	lugar	C/ Almibar, 156 🗑
<input checked="" type="checkbox"/>	tipo	tienda con escarparte
	Key	Value

Body Cookies Headers (7) Test Results 🌐 Status: 200 OK Time: 85 ms Size: 281 B Save Response

Pretty Raw Preview Visualize **HTML** 🔍

```

1 6669fc06b4bc236945c07af9 se ha añadido correctamente

```

```
197
198   const tiendita = await Tienda.find();
199   console.log(tiendita);
200
201
202
203
204
205
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
    lugar: 'C/Toreto, 15',
    tipo: [ 'edificio con vistas', 'planta baja', 'almacén', 'camión' ],
    __v: 0
  },
  {
    _id: new ObjectId('6669fc06b4bc236945c07af9'),
    nombre: 'Perfumes naturales',
    precio: 564,
    tipoPago: 'semanal',
    alquilado: false,
    lugar: 'C/ Almíbar, 156',
    tipo: [ 'tienda con escaparate' ],
    __v: 0
  },
```

Update TIENDA:

```
196   });
197
198   // update
199
200   app.post('/actualizaTienda', async function (req,res){
201     // datos del formulario
202     let id = req.body.id;
203     let nombre = req.body.nombre;
204     let precio = req.body.precio;
205     let tipoPago = req.body.tipoPago;
206     let fecha = req.body.fecha;
207     let alquilado = req.body.alquilado;
208     let lugar = req.body.lugar;
209     let tipo = req.body.tipo;
210
211     // buscar por id y cambiar datos
212     let shop = await Tienda.findByIdAndUpdate(id, {
213       nombre: nombre,
214       precio: precio,
215       tipoPago: tipoPago,
216       fecha: fecha,
217       alquilado: alquilado,
218       lugar: lugar,
219       tipo: tipo
220     });
221     res.send (shop.id+' se ha actualizado');
222   });
```

Resultado update:

POST http://localhost:3000/actualizaTienda

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary

Key	Value
nombre	Perfumes de flores
precio	564
tipoPago	semanal

Body Cookies Headers (7) Test Results Status: 200 OK Time: 89 ms Size: 270 B

Pretty Raw Preview Visualize HTML

```
1 6669fc06b4bc236945c07af9 se ha actualizado
```

```
227
228
229 const tiendita = await Tienda.find();
230 console.log(tiendita);
231
232
233
234
235
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
alquilado: true,
lugar: 'C/Toreto, 15',
tipo: [ 'edificio con vistas', 'planta baja', 'almacén', 'camión' ],
__v: 0
},
{
  _id: new ObjectId('6669fc06b4bc236945c07af9'),
  nombre: 'Perfumes de flores',
  precio: 564,
  tipoPago: 'semanal',
  alquilado: false,
  lugar: 'C/ Frambuesa, 156',
  tipo: [ '"edificio con vistas", "almacén"' ],
  __v: 0
}
```

Ln 220, Col 4 Spaces: 2 UTF-

Find CLIENTE:

```
18
19 app.get('/listaclientes', async (req,res)=>{
20   let cliente = await Cliente.find();
21   res.send(cliente);
22 })
```

Resultado:

The screenshot shows a web browser interface for testing HTTP requests. The URL bar shows `http://localhost:3000/listaclientes`. The request method is `GET`. The response body is displayed in JSON format, showing a single client object with the following details:

Key	Value
nombre	Perfumes de flores
precio	564
tipoPago	semanal

The JSON response is as follows:

```
{
  "_id": "6669ec01f66b64b2c47eb538",
  "nombre": "Lorenzo",
  "apellidos": "Bermudez Moreno",
  "tarifa": [
    {
      "tipo": "premium",
      "constancia": "ininterrumpida"
    }
  ],
  "contacto": "lorenzo@gmail.com",
  "__v": 0
}
```

delete CLIENTE:

```
0
1
2 app.get('/borrarcliente/:id', async (req,res)=>{
3   let borraCliente = await Cliente.findByIdAndDelete(req.params.id);
4   res.send(borraCliente + 'Borrado correctamente');
5 })
6
```

Resultado delete:

GET http://localhost:3000/bor

http://localhost:3000/borrarcliente/6669ec01f66b64b2c47eb538

GET http://localhost:3000/borrarcliente/6669ec01f66b64b2c47eb538

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	nombre	Perfumes de flores	
<input checked="" type="checkbox"/>	precio	564	
<input checked="" type="checkbox"/>	tipoPago	semanal	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 85 ms Size: 462 B Save Response

Pretty Raw Preview Visualize HTML

```
1 {
2   _id: new ObjectId('6669ec01f66b64b2c47eb538'),
3   nombre: 'Lorenzo',
4   apellidos: 'Bermudez Moreno',
5   tarifa: [ { tipo: 'premium', constancia: 'ininterrumpida' } ],
6   contacto: 'lorenzo@gmail.com',
7   __v: 0
8 } Borrado correctamente
```

Insert CLIENTE:

```
69
70 app.post('/insertacliente', async (req,res)=>{
71   let nombre = req.body.nombre;
72   let apellidos = req.body.apellidos;
73   let tarifa = JSON.parse(req.body.tarifa);
74   let contacto = req.body.contacto;
75
76   // crea nuevo cliente
77   const nuevoCliente =
78     new Cliente ({
79     nombre: nombre,
80     apellidos: apellidos,
81     tarifa: tarifa,
82     contacto: contacto,
83   });
84   // guarda los datos
85   await nuevoCliente.save();
86   💡 muestra que todo salió bien
87   res.send(nuevoCliente.id+' se ha añadido correctamente');
88 });
89
90
91
```

Resultado insert:

POST http://localhost:3000/insertacliente

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> nombre	Lola	
<input checked="" type="checkbox"/> apellidos	Garrido Sams	
<input checked="" type="checkbox"/> tarifa	[{"tipo": "Básico", "constancia": "fines de semana"}]	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 82 ms Size: 281 B Save Response

Pretty Raw Preview Visualize HTML

```
1 666a05fbae98d4418a7e6a96 se ha añadido correctamente
```

GET http://localhost:3000/lista:

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Co

none form-data x-www-form-urlencoded raw binary

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> nombre	Lola	
<input checked="" type="checkbox"/> apellidos	Garrido Sams	
<input checked="" type="checkbox"/> tarifa	[{"tipo": "Básico", "constancia": "fines de semana"}]	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 99 ms Size: 5.77 KB Save Respo

Pretty Raw Preview Visualize JSON

```
377     "id": "666a05fbae98d4418a7e6a96",
378   },
379   {
380     "id": "666a05fbae98d4418a7e6a96",
381     "nombre": "Lola",
382     "apellidos": "Garrido Sams",
383     "tarifa": [
384       {
385         "tipo": "Básico",
386         "constancia": "fines de semana"
387       }
388     ],
389     "contacto": "lola@gmail.com",
390     "_v": 0
391   },
392   {
```

Update CLIENTE:

```
5
6 app.post('/actualizaCliente', async function (req,res){
7   // datos del formulario
8   let id = req.body.id;
9   let nombre = req.body.nombre;
10  let apellidos = req.body.apellidos;
11  let tarifa = JSON.parse(req.body.tarifa);
12  let contacto = req.body.contacto;
13
14  // buscar por id y cambiar datos
15  let cliente = await Cliente.findByIdAndUpdate(id, {
16    nombre: nombre,
17    apellidos: apellidos,
18    tarifa: tarifa,
19    contacto: contacto,
20  });
21  res.send (cliente.id+' se ha actualizado');
22  });
23
24
25
```

Resultado update:

POST http://localhost:3000/actualizaCliente Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary

<input checked="" type="checkbox"/>	apellidos	Garrido Moreno
<input checked="" type="checkbox"/>	tarifa	[{"tipo": "Premium", "constancia": "fines de semana"}]
<input checked="" type="checkbox"/>	contacto	lola@gmail.com
<input checked="" type="checkbox"/>	id	666a05fbae98d4418a7e6a96

ody Cookies Headers (7) Test Results Status: 200 OK Time: 73 ms Size: 270 B Save Resp

Pretty Raw Preview Visualize HTML

```
1 666a05fbae98d4418a7e6a96 se ha actualizado
```

HTTP http://localhost:3000/listaclientes

GET http://localhost:3000/listaclientes

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary

apellidos	Garrido Moreno
tarifa	[{"tipo": "Premium", "constancia": "fines de semana"}]
contacto	lola@gmail.com
id	666a05fbae98d4418a7e6a96

Body Cookies Headers (7) Test Results Status: 200 OK Time: 65 ms Size: 6.13 KB Save Response

Pretty Raw Preview Visualize JSON

```
378 },
379 {
380   "_id": "666a05fbae98d4418a7e6a96",
381   "nombre": "Lola",
382   "apellidos": "Garrido Moreno",
383   "tarifa": [
384     {
385       "tipo": "Premium",
386       "constancia": "fines de semana"
387     }
388   ],
389   "contacto": "lola@gmail.com",
390   "__v": 0
391 },
392 }
```

Todo correcto en ATLAS:

ATLAS

DATABASES: 8 COLLECTIONS: 16

+ Create Database

Search Namespaces

- Inventario
- Musica
- PRO2324
- Spotify
- media
- popup**
 - clientes
 - popup
 - popups**
 - sample_mflix
 - tienda

popup.popups

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 940B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

Filter Type a query: { field: 'value' } Reset Apply Op

QUERY RESULTS: 1-20 OF MANY

```
{
  "_id": ObjectId('6669ea4b193dea022c666287'),
  "nombre": "Atletico de altura",
  "precio": 500,
  "tipoPago": "mensual",
  "fecha": "2024-08-16T00:00:00.000+00:00",
  "alquilado": true,
  "lugar": "C/Toreto, 15",
  "tipo": Array (4),
  "__v": 0
}
```

```
{
  "_id": ObjectId('6669ea61f28d9e73f5b3e71a'),
  "nombre": "Atletico de altura",
  "precio": 500,
  "tipoPago": "mensual",
  "fecha": "2024-08-16T00:00:00.000+00:00",
  "alquilado": true,
  "lugar": "C/Toreto, 15"
}
```

