



# **PRESENTACIÓN FINAL ING. DATOS**

Juan Arroyo  
Juan Arevalo  
William Ramirez



# CONTEXTO

- ¿Quién es nuestro cliente?
- ¿Qué problemas tenía?
- ¿Cómo nos comunicábamos con él?

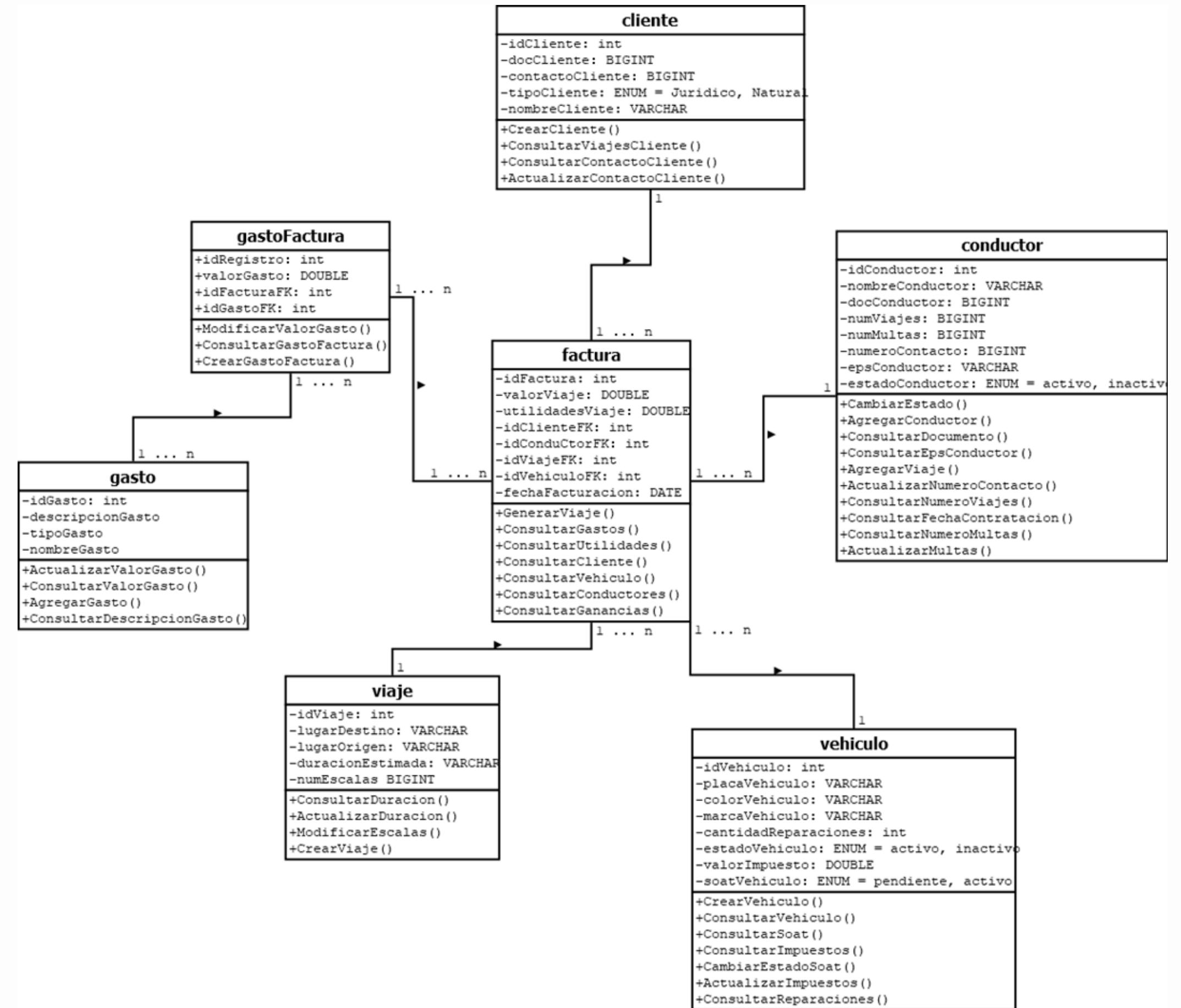


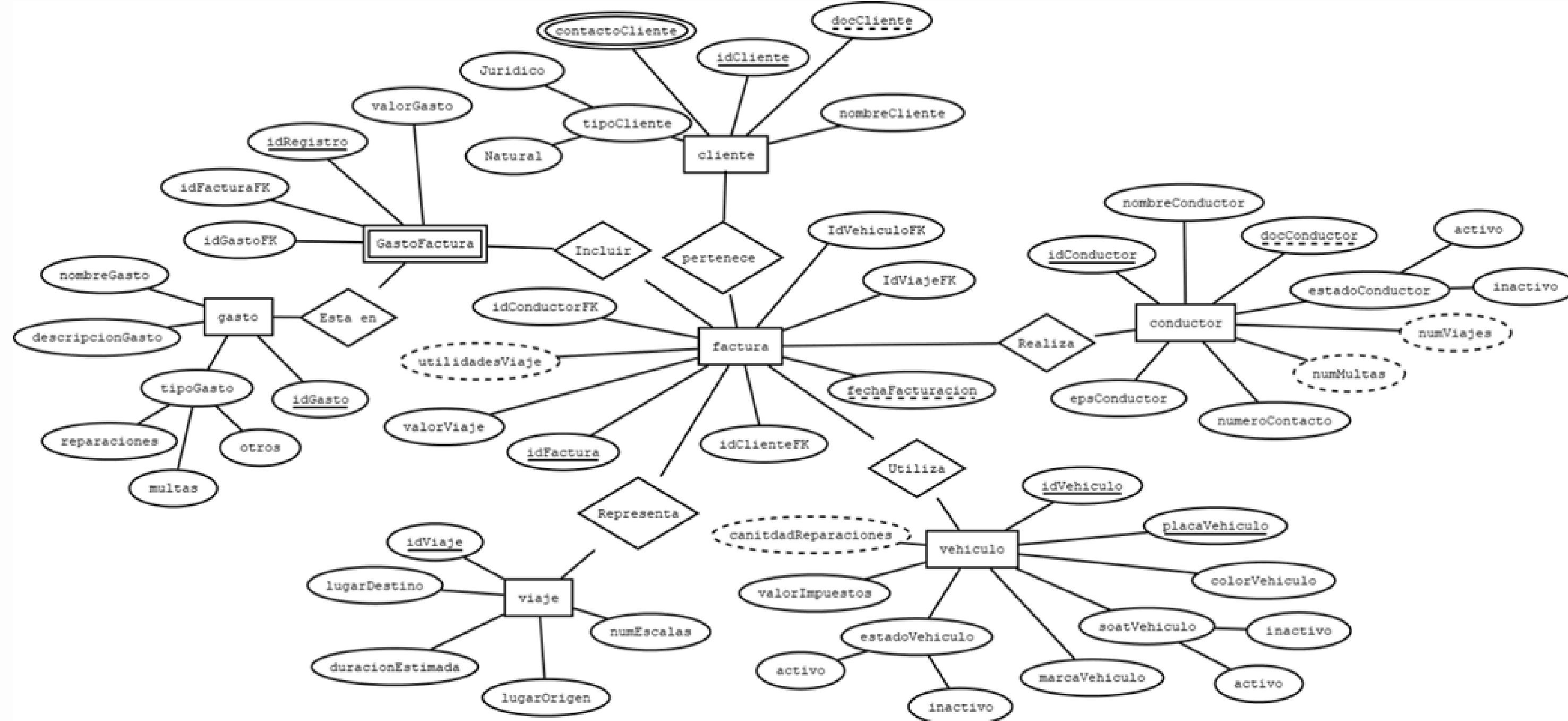
# IDEA DE LA SOLUCION

Para llegar a la solución que implementamos unimos las ideas que teníamos y las pulimos con las retroalimentaciones de la docente. La solución final constaba de 7 clases cada una con información que ayudaría a nuestro cliente a tener una mejor administración de su empresa

# IDEA DE LA SOLUCIÓN

Para llegar a la solución que implementamos unimos las ideas que teníamos y las pulimos con las retroalimentaciones de la docente. La solución final constaba de 7 clases cada una con información que ayudaría a nuestro cliente a tener una mejor administración de su empresa





# IDEA DE LA SOLUCIÓN

<b>idConductor</b>	<b>nombreConductor</b>	<b>docConductor</b>	<b>numViajes</b>	<b>numMultas</b>	<b>numeroContacto</b>	<b>EpsConductor</b>	<b>estadoConductor</b>
1	Mario Mendoza	1031650258	0	0	3053524931	compensar	inactivo
2	Carlos Pérez	1029876543	5	1	3204567890	Sanitas	activo
3	Jorge Ramírez	1013456789	21	3	3109876543	Compensar	activo
4	Luis Torres	1002345678	8	8	3182345678	Nueva EPS	inactivo
5	Andrés Martínez	1009871234	23	3	3012345678	Sura	activo
6	Pedro Herrera	1034567891	13	5	3119876543	Coomeva	activo
7	Juan Ríos	1011122233	15	4	3198765432	Sanitas	inactivo
8	David López	1056789012	8	3	3145678901	Compensar	activo
9	Óscar Díaz	1001234567	5	1	3151234567	Sura	activo

# CREACIÓN DE LAS TABLAS

<b>idVehiculo</b>	<b>placaVehiculo</b>	<b>colorVehiculo</b>	<b>marcaVehiculo</b>	<b>cantidadReparaciones</b>	<b>estadoVehiculo</b>	<b>valorImpuesto</b>	<b>SoatVehiculo</b>
1	SLF799	Rojo	KIA	1	inactivo	95000.000	pendiente
2	SLF044	Negro	Mazda	0	activo	87000.000	activo
3	SJP614	Blanco	Chevrolet	3	inactivo	120000.000	pendiente
4	SDN758	Gris	Hyundai	2	activo	110000.000	activo
5	BRL123	Rojo	Renault	0	activo	83000.000	activo
6	FGH456	Azul	Toyota	0	inactivo	105000.000	pendiente
7	MNB789	Negro	Nissan	4	activo	99000.000	activo
8	QWE234	Verde	Ford	1	activo	98000.000	pendiente
9	ASD567	Blanco	Volkswagen	9	inactivo	89000.000	activo
10	ZXC890	Gris	KIA	5	activo	102000.000	pendiente

# CREACIÓN DE LAS TABLAS

<b>idCliente</b>	<b>docCliente</b>	<b>contactoCliente</b>	<b>tipoCliente</b>	<b>nombreCliente</b>
1	1023456789	3124567890	juridico	TransRisaralda
2	2234567890	3135678901	juridico	CCM Ingenieros
3	3345678901	3146789012	juridico	INVERTRANS
4	4456789012	3157890123	juridico	OLT
5	5567890123	3168901234	juridico	ALITRANS
6	6678901234	3179012345	juridico	TRANSPCOL
7	7789012345	3180123456	juridico	LOGYSTEL
8	8890123456	3191234567	juridico	SERVIENTREGA
9	9901234567	3202345678	juridico	DEPRISA
10	1012345678	3213456789	juridico	JYM
11	1345678901	3246789012	natural	Sergio Peña

# CREACIÓN DE LAS TABLAS

<b>idViaje</b>	<b>lugarDestino</b>	<b>lugarOrigen</b>	<b>duracionEstimada</b>	<b>numEscalas</b>
1	Bogotá	Pasto	12 horas	2
2	Medellín	Barranquilla	2 días	1
3	Cali	Armenia	6 horas	0
4	Tunja	Villavicencio	1 día y 2 horas	2
5	Cartagena	Montería	5 horas y 15 min	0
6	Manizales	Ibagué	3 horas	1
7	Bucaramanga	Neiva	2 días y 6 horas	2
8	Pereira	Santa Marta	1 día	1
9	Cúcuta	Bogotá	16 horas	3
10	Quibdó	Turbo	10 horas y 20 min	1
11	Barrancaber...	Cali	18 horas	2

# CREACIÓN DE LAS TABLAS

<b>idGasto</b>	<b>descripcionGasto</b>	<b>tipoGasto</b>	<b>nombreGasto</b>
1	Cambio de aceite y filtro	1	Aceite
2	Reparacion de frenos delanteros	1	Frenos Delante
3	Cambio de correa de distribucion	1	Correa
4	Alineacion y balanceo completo	1	Alineacion
5	Reparacion de suspension trasera	1	Suspension
6	Reparacion del sistema de escape	1	Escape
7	Cambio de bujias desgastadas	1	Bujias
8	Mantenimiento general del motor	1	Mantenimiento
9	Reparacion del sistema electrico	1	Electrico
10	Reemplazo de amortiguadores	1	Amortiguad

# CREACIÓN DE LAS TABLAS

<b>idFactura</b>	<b>valorViaje</b>	<b>utilidadesViaje</b>	<b>idClienteFK</b>	<b>idConductorFK</b>	<b>idViajeFK</b>	<b>idVehiculoFK</b>	<b>FechaFacturacion</b>
1	180000.000	15000.000	1	3	6	9	2024-04-08
2	250000.000	20000.000	2	4	6	10	2025-01-19
3	320000.000	28000.000	3	5	8	9	2024-11-20
4	150000.000	12000.000	4	3	9	12	2025-02-08
5	400000.000	35000.000	5	3	10	13	2024-12-13
6	220000.000	18000.000	6	6	11	9	2023-11-12
7	270000.000	22000.000	6	6	12	15	2024-09-01
8	310000.000	29000.000	3	4	13	16	2024-08-21
9	190000.000	13000.000	9	7	14	17	2025-03-15

# CREACIÓN DE LAS TABLAS

<b>idRegistro</b>	<b>valorGasto</b>	<b>idFacturaFK</b>	<b>idGastoFK</b>
1	50000.000	1	1
2	60000.000	1	2
3	55000.000	1	3
4	70000.000	2	4
5	80000.000	2	5
6	80000.000	2	6
7	90000.000	3	7
8	100000.000	3	8
9	102000.000	3	9
10	40000.000	4	10

# CREACIÓN DE LAS TABLAS

```
CREATE PROCEDURE actualizarEPS (IN idModif int, nuevaEPs varchar(20))
BEGIN
UPDATE conductor
SET EpsConductor=nuevaEPs WHERE idConductor=idModif;
END $$

DELIMITER ;
```

# SCRIPTS IMPORTANTES

```
SELECT valorGasto,nombreConductor,descripcionGasto FROM GASTOFACTURA  
INNER JOIN GASTO on idGasto=idGastoFK  
INNER JOIN FACTURA ON idFacturaFK=idFactura  
INNER JOIN CONDUCTOR ON idConductorFK=idConductor  
WHERE idConductorFK=5 and tipoGasto=2;
```

valorGasto	nombreConductor	descripcionGasto
200000.000	Andrés Martínez	Multa por obstrucción de paso peatonal

# SCRIPTS IMPORTANTES

```

CREATE VIEW facturaCompleta AS
SELECT f.valorViaje,f.utilidadesViaje,v.lugarOrigen,v.lugarDestino,c.nombreConductor AS "conductor",
cl.nombreCliente ,ve.marcaVehiculo,
ve.placaVehiculo FROM factura f
INNER JOIN viaje v ON v.idViaje=f.idViajeFK
INNER JOIN conductor c ON c.idConductor=f.idConductorFK
INNER JOIN vehiculo ve ON ve.idVehiculo=f.idVehiculoFK
INNER JOIN cliente cl ON cl.idCliente=f.idClienteFK;

```

valorViaje	utilidadesViaje	lugarOrigen	lugarDestino	conductor	nombreCliente	marcaVehiculo	placaVehiculo
1800000.000	15000.000	Ibagué	Manizales	Jorge Ramírez	TransRisaralda	Volkswagen	ASD567
2500000.000	20000.000	Ibagué	Manizales	Luis Torres	CCM Ingenieros	KIA	ZXC890
3200000.000	28000.000	Santa Marta	Pereira	Andrés Martínez	INVERTTRANS	Volkswagen	ASD567
1500000.000	12000.000	Bogotá	Cúcuta	Jorge Ramírez	OLT	Mazda	UIO654
4000000.000	35000.000	Turbo	Quibdó	Jorge Ramírez	ALITRANS	Chevrolet	BNM987
2200000.000	18000.000	Cali	Barrancabermeja	Pedro Herrera	TRANSPCOL	Volkswagen	ASD567
2700000.000	22000.000	Florencia	Popayán	Pedro Herrera	TRANSPCOL	Toyota	YHN753
3100000.000	29000.000	Puerto Nariño	Leticia	Luis Torres	INVERTTRANS	Nissan	EDC852

# SCRIPTS IMPORTANTES

```
CREATE TRIGGER añadirReparaciones AFTER INSERT ON gastoFactura FOR EACH ROW
BEGIN
IF (SELECT tipoGasto FROM gasto WHERE idGasto = NEW.idGastoFK)=1 THEN
UPDATE VEHICULO
SET cantidadReparaciones=cantidadReparaciones+1 WHERE idVehiculo=(select idVehiculoFK from factura
where idFactura=new.idFacturaFK);
END IF;
END $$

DELIMITER ;
```

# SCRIPTS IMPORTANTES

```
DELIMITER $$  
CREATE TRIGGER añadirMultas AFTER INSERT ON gastoFactura FOR EACH ROW  
BEGIN  
    IF (SELECT tipoGasto FROM gasto WHERE idGasto = NEW.idGastoFK) = 2 THEN  
        UPDATE conductor  
        SET numMultas=numMultas+1 WHERE idConductor=(select idConductorFK from factura  
        where idFactura=new.idFacturaFK);  
    END IF;  
END $$  
DELIMITER ;
```

# SCRIPTS IMPORTANTES

```
CREATE VIEW gastosDeUnaFactura AS  
SELECT idFactura, utilidadesViaje, valorViaje, valorGasto, idGastoFK AS "id gasto" FROM factura  
INNER JOIN gastoFactura ON idFacturaFK=idFactura;
```

<b>idFactura</b>	<b>utilidadesViaje</b>	<b>valorViaje</b>	<b>valorGasto</b>	<b>id gasto</b>
1	15000.000	180000.000	50000.000	1
1	15000.000	180000.000	60000.000	2
1	15000.000	180000.000	55000.000	3
2	20000.000	250000.000	70000.000	4
2	20000.000	250000.000	80000.000	5
2	20000.000	250000.000	80000.000	6
3	28000.000	320000.000	90000.000	7
3	28000.000	320000.000	100000.000	8

# SCRIPTS IMPORTANTES

# BASE DE DATOS NOSQL

Para la base de datos Nosql creamos modulos con los mismos nombres de las tablas, le creamos su respectiva ruta a cada modelo. En estas rutas creamos las urls para las peticiones

```
const ClienteSchema=new mongoose.Schema({
  idCliente:Number,
  docCliente:{type:Number,require:true},
  contactoCliente:String,
  tipoCliente:String,
  nombreCliente:String,
  creadoEn:{type:Date,default:Date.now}
});
```

The screenshot shows a POST request to the URL `http://localhost:3000/api/Cliente`. The request body contains an array of eight client documents. The response shows a single document from the array.

Request Headers:

- POST
- Content-Type: application/json
- Accept: application/json

Request Body (Raw JSON):

```
[{"idCliente": 1, "docCliente": 1023456789, "contactoCliente": "3124567890", "tipoCliente": "juridico", "nombreCliente": "TransRisaralda"}, {"idCliente": 2, "docCliente": 2234567890, "contactoCliente": "3135678901", "tipoCliente": "juridico", "nombreCliente": "CCM Ingenieros"}, {"idCliente": 3, "docCliente": 3345678901, "contactoCliente": "3146789012", "tipoCliente": "juridico", "nombreCliente": "INVERTTRANS"}, {"idCliente": 4, "docCliente": 4456789012, "contactoCliente": "3157890123", "tipoCliente": "juridico", "nombreCliente": "OLT"}, {"idCliente": 5, "docCliente": 5567890123, "contactoCliente": "3168901234", "tipoCliente": "juridico", "nombreCliente": "ALITRANS"}, {"idCliente": 6, "docCliente": 6678901234, "contactoCliente": "3179012345", "tipoCliente": "juridico", "nombreCliente": "TRANSPCOL"}, {"idCliente": 7, "docCliente": 7789012345, "contactoCliente": "3180123456", "tipoCliente": "juridico", "nombreCliente": "LOGYSTELE"}, {"idCliente": 8, "docCliente": 8890123456, "contactoCliente": "3191234567", "tipoCliente": "juridico", "nombreCliente": "SERVENTREGA"}]
```

Response Headers:

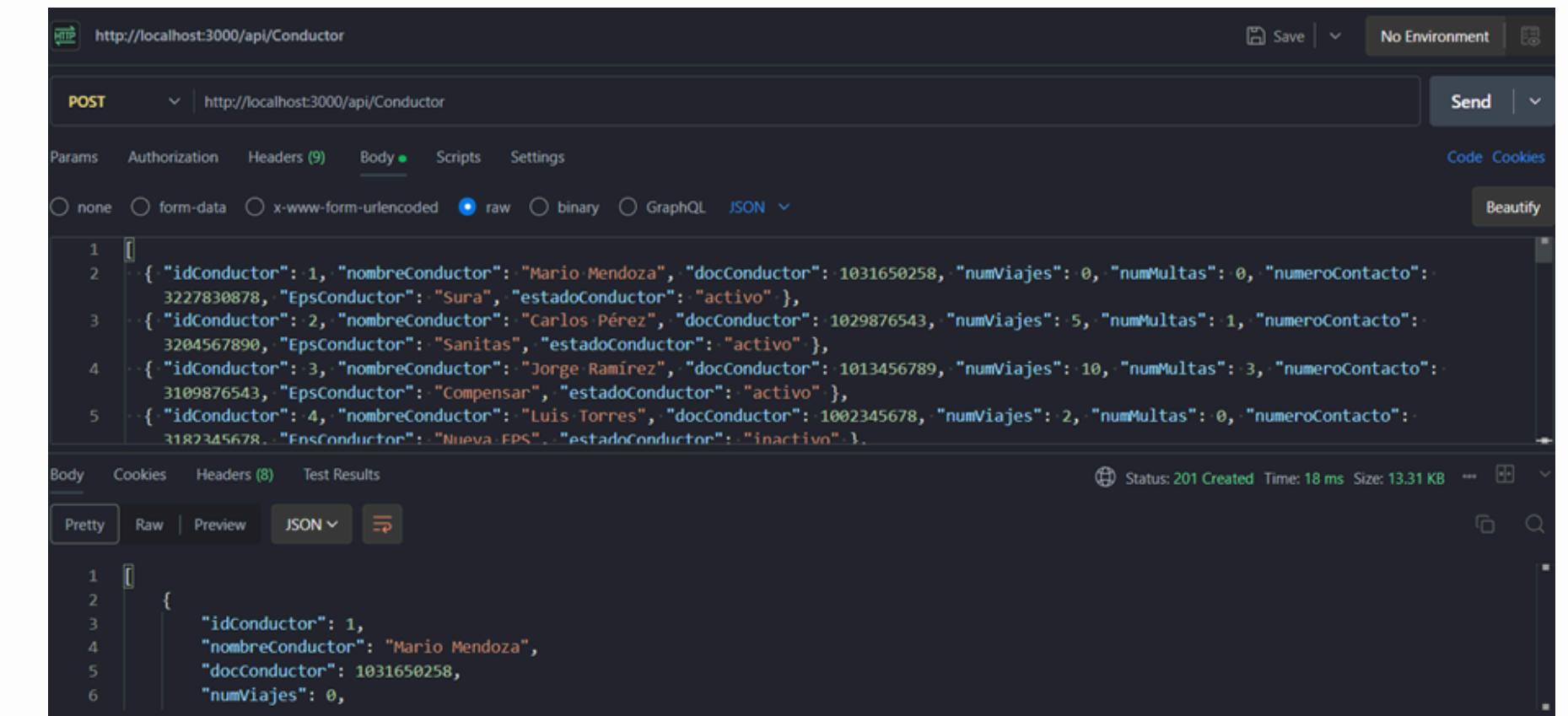
- Status: 201 Created
- Time: 62 ms
- Size: 9.94 KB

Response Body (Pretty JSON):

```
[{"idCliente": 1, "docCliente": 1023456789, "contactoCliente": "3124567890", "tipoCliente": "juridico", "nombreCliente": "TransRisaralda"}]
```

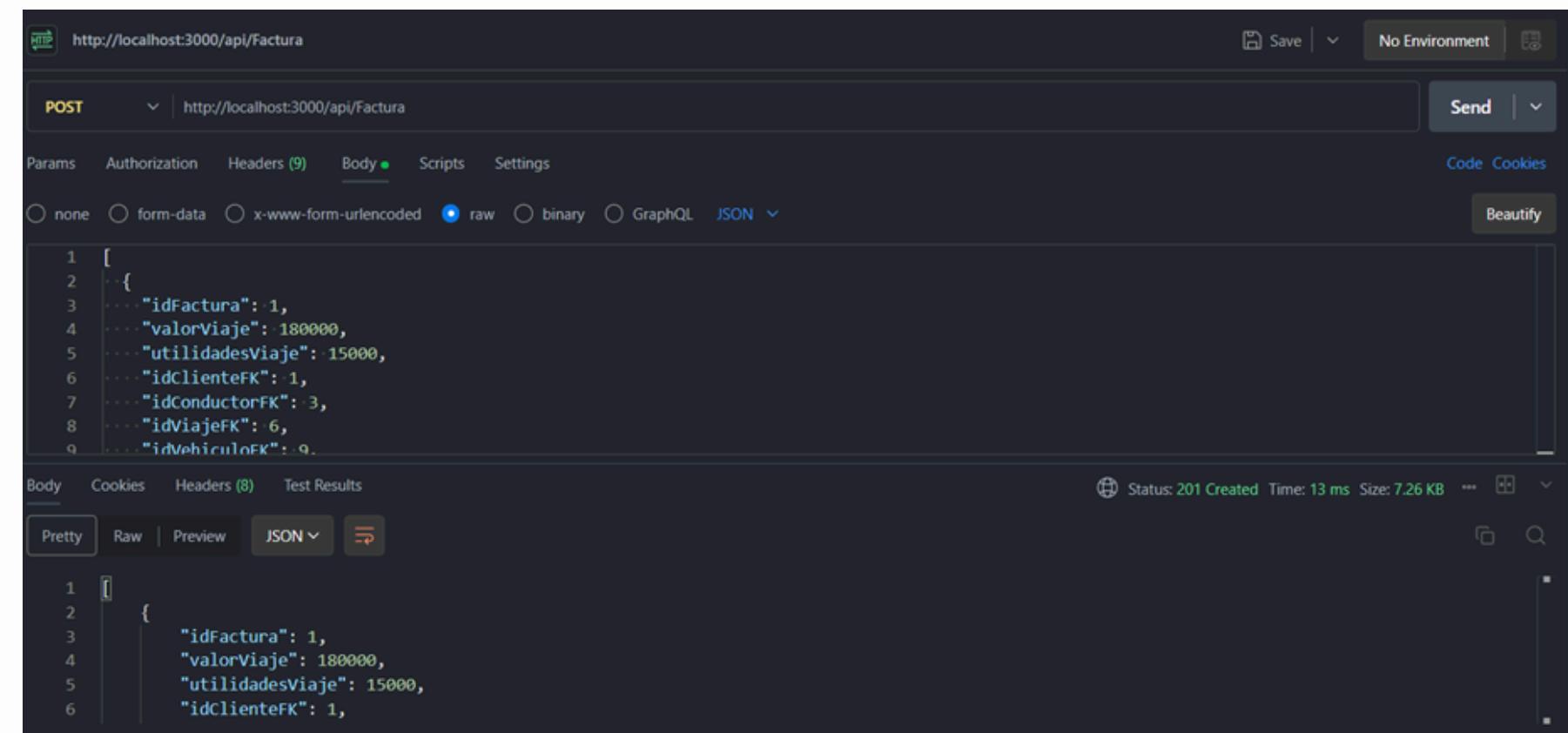
# BASE DE DATOS NOSQL

```
const conductorSchema=new mongoose.Schema({
  idConductor:Number,
  nombreConductor:{type:String,require:true},
  docConductor:Number,
  numViajes:Number,
  numMultas:Number,
  numeroContacto:Number,
  EpsConductor:String,
  estadoConductor:String,
  creadoEn:{type:Date,default:Date.now}
});
```



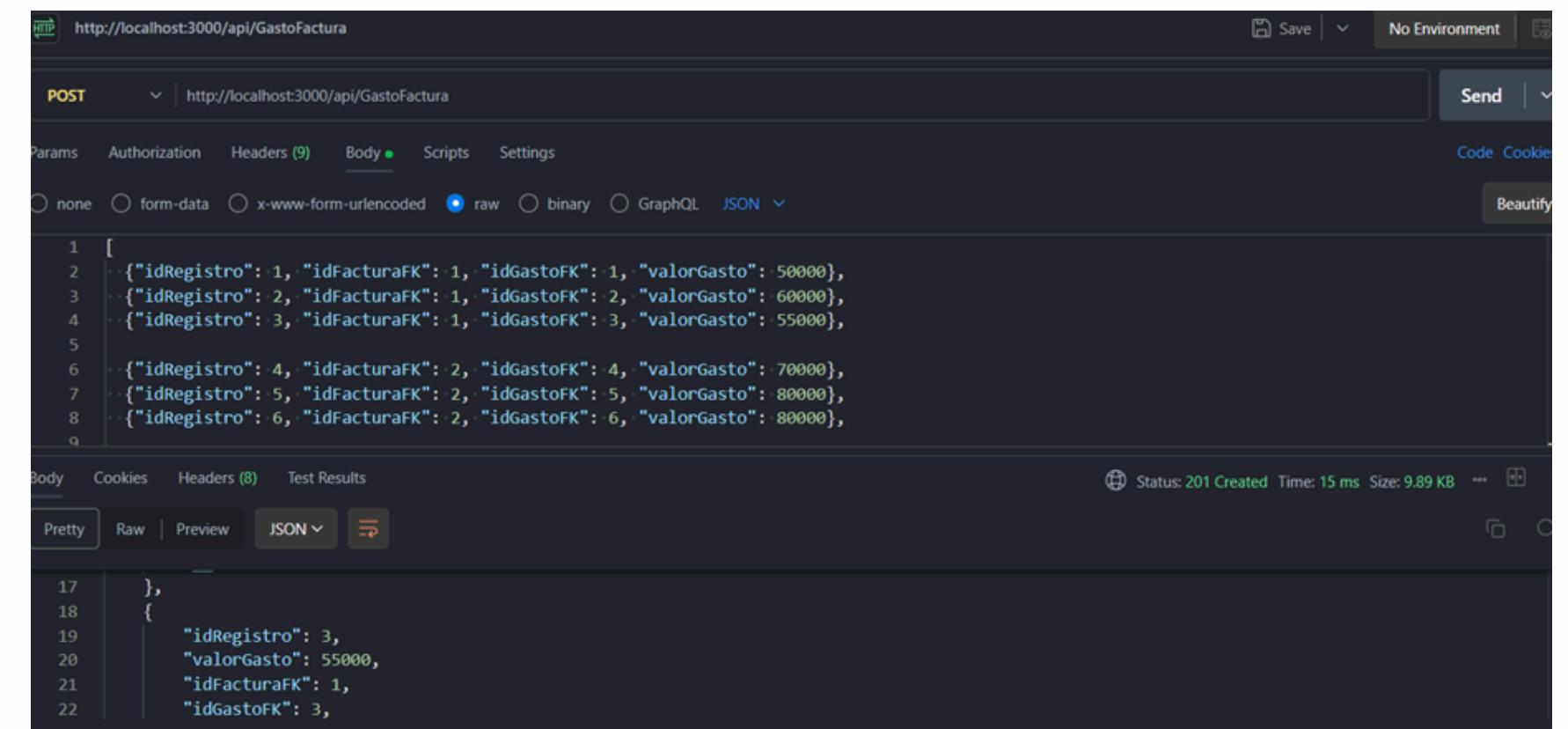
# BASE DE DATOS NOSQL

```
const facturasSchema=new mongoose.Schema({
  idFactura:Number,
  valorViaje:Number,
  utilidadesViaje:Number,
  idClienteFK:Number,
  idConductorFK:Number,
  idViajeFK:Number,
  idVehiculoFK:Number,
  FechaFacturacion:String,
  creadoEn:{type:Date,default:Date.now}
});
```



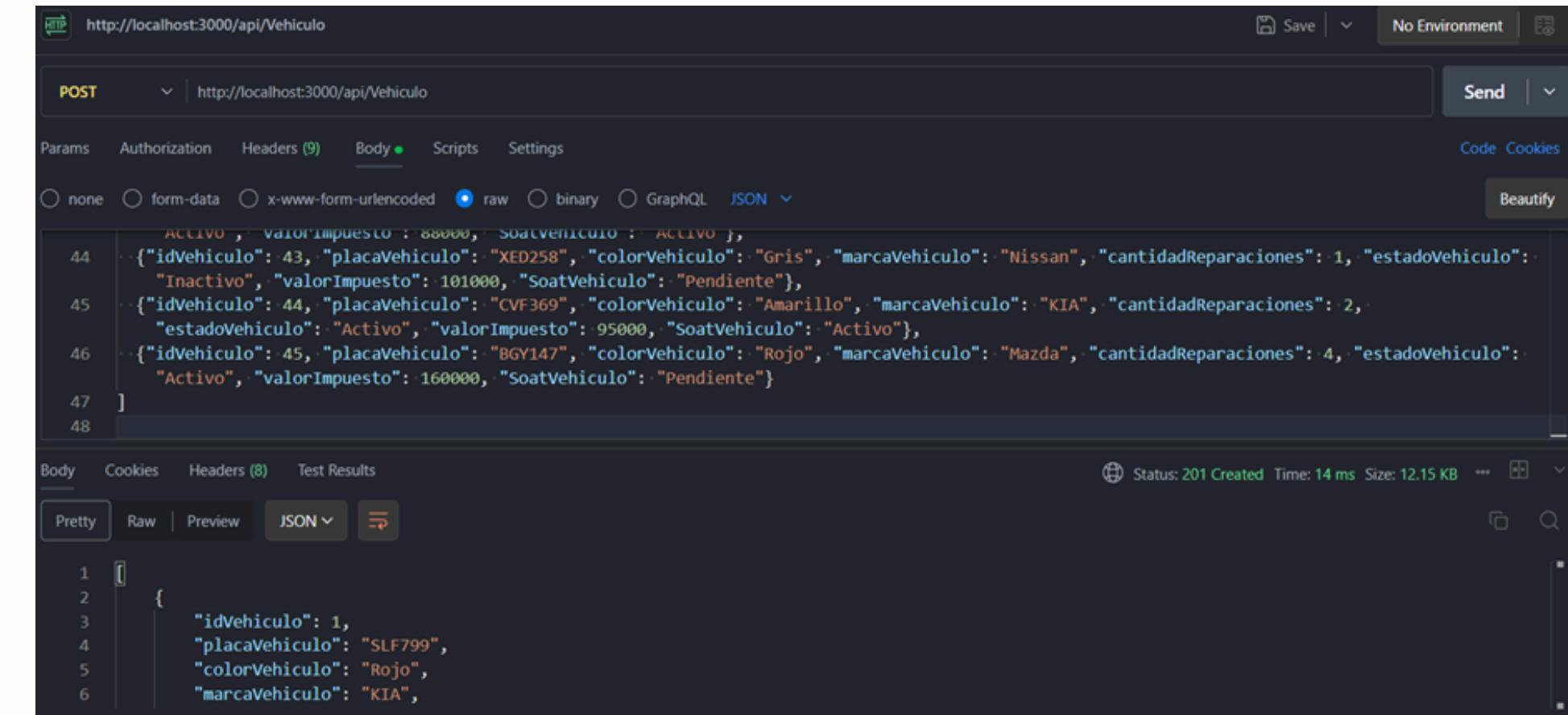
# BASE DE DATOS NOSQL

```
const GastoFacturaSchema=new mongoose.Schema({
  idRegistro:Number,
  valorGasto:Number,
  idFacturaFK:Number,
  idGastoFK:Number
});
```



# BASE DE DATOS NOSQL

```
const vehiculoSchema=new mongoose.Schema({
    idVehiculo:Number,
    placaVehiculo:{type:String,require:true},
    colorVehiculo:String,
    marcaVehiculo:String,
    cantidadReparaciones:Number,
    estadoVehiculo:String,
    valorImpuesto:Number,
    SoatVehiculo:String,
    creadoEn:{type:Date,default:Date.now}
});
```



# BASE DE DATOS NOSQL

```
const ViajeSchema=new mongoose.Schema({
  idViaje:Number,
  lugarDestino:{type:String,require:true},
  lugarOrigen:String,
  duracionEstimada:String,
  numEscalas:Number,
  creadoEn:{type:Date,default:Date.now}
});
```

The screenshot shows a Postman interface with a POST request to `http://localhost:3000/api/Viaje`. The request body contains an array of eight objects representing travel routes. The response body shows the first object from the array, indicating a successful creation with status 201 Created.

POST | http://localhost:3000/api/Viaje

Params | Authorization | Headers (9) | Body | Scripts | Settings

raw

1 [ { "idViaje": 1, "lugarDestino": "Bogotá", "lugarOrigen": "Pasto", "duracionEstimada": "12 horas", "numEscalas": 2 }, { "idViaje": 2, "lugarDestino": "Medellín", "lugarOrigen": "Barranquilla", "duracionEstimada": "2 días", "numEscalas": 1 }, { "idViaje": 3, "lugarDestino": "Cali", "lugarOrigen": "Armenia", "duracionEstimada": "6 horas", "numEscalas": 0 }, { "idViaje": 4, "lugarDestino": "Tunja", "lugarOrigen": "Villavicencio", "duracionEstimada": "1 día y 2 horas", "numEscalas": 2 }, { "idViaje": 5, "lugarDestino": "Cartagena", "lugarOrigen": "Montería", "duracionEstimada": "5 horas y 15 min", "numEscalas": 0 }, { "idViaje": 6, "lugarDestino": "Manizales", "lugarOrigen": "Ibagué", "duracionEstimada": "3 horas", "numEscalas": 1 }, { "idViaje": 7, "lugarDestino": "Bucaramanga", "lugarOrigen": "Neiva", "duracionEstimada": "2 días y 6 horas", "numEscalas": 2 }, { "idViaje": 8, "lugarDestino": "Pereira", "lugarOrigen": "Santa Marta", "duracionEstimada": "1 día", "numEscalas": 1 } ]

Body | Cookies | Headers (8) | Test Results

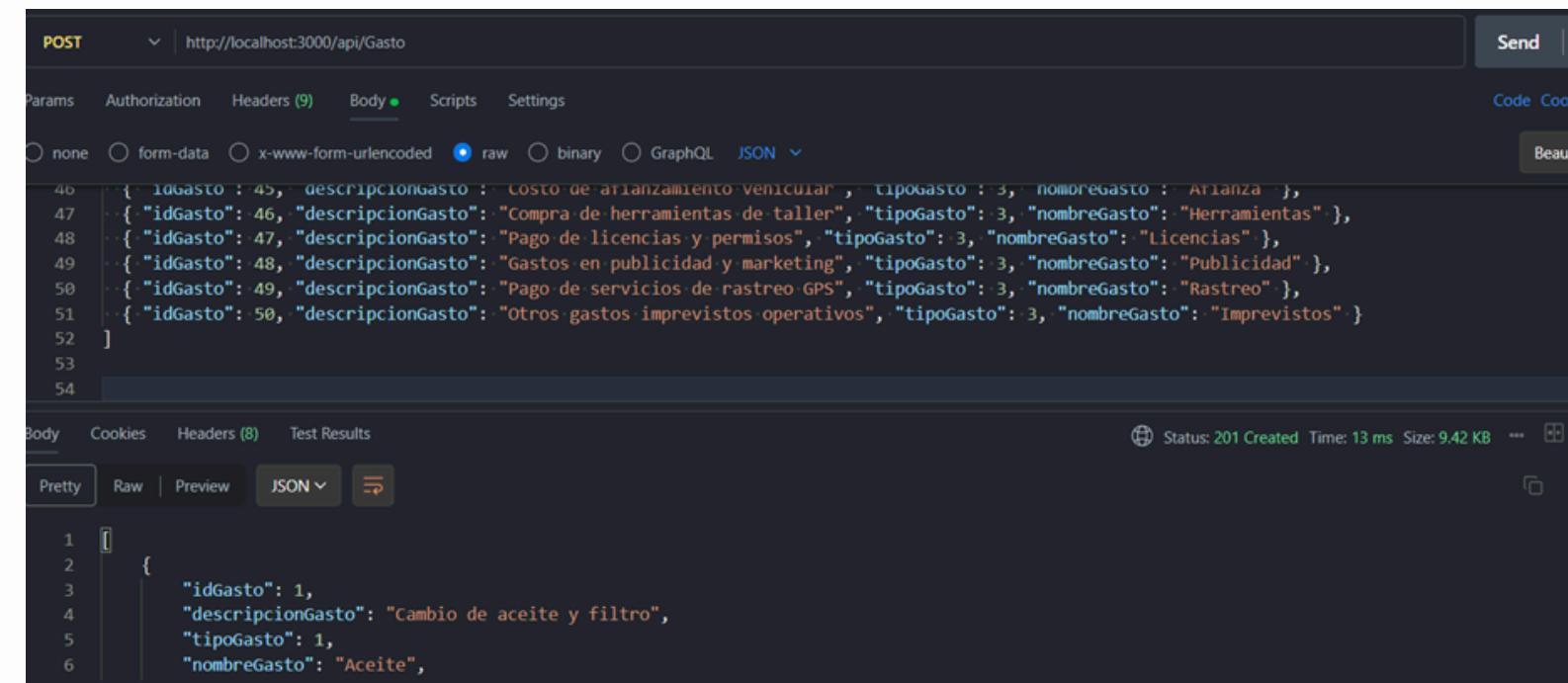
Pretty | Raw | Preview | JSON

1 [ { "idViaje": 1, "lugarDestino": "Bogotá", "lugarOrigen": "Pasto", "duracionEstimada": "12 horas", }

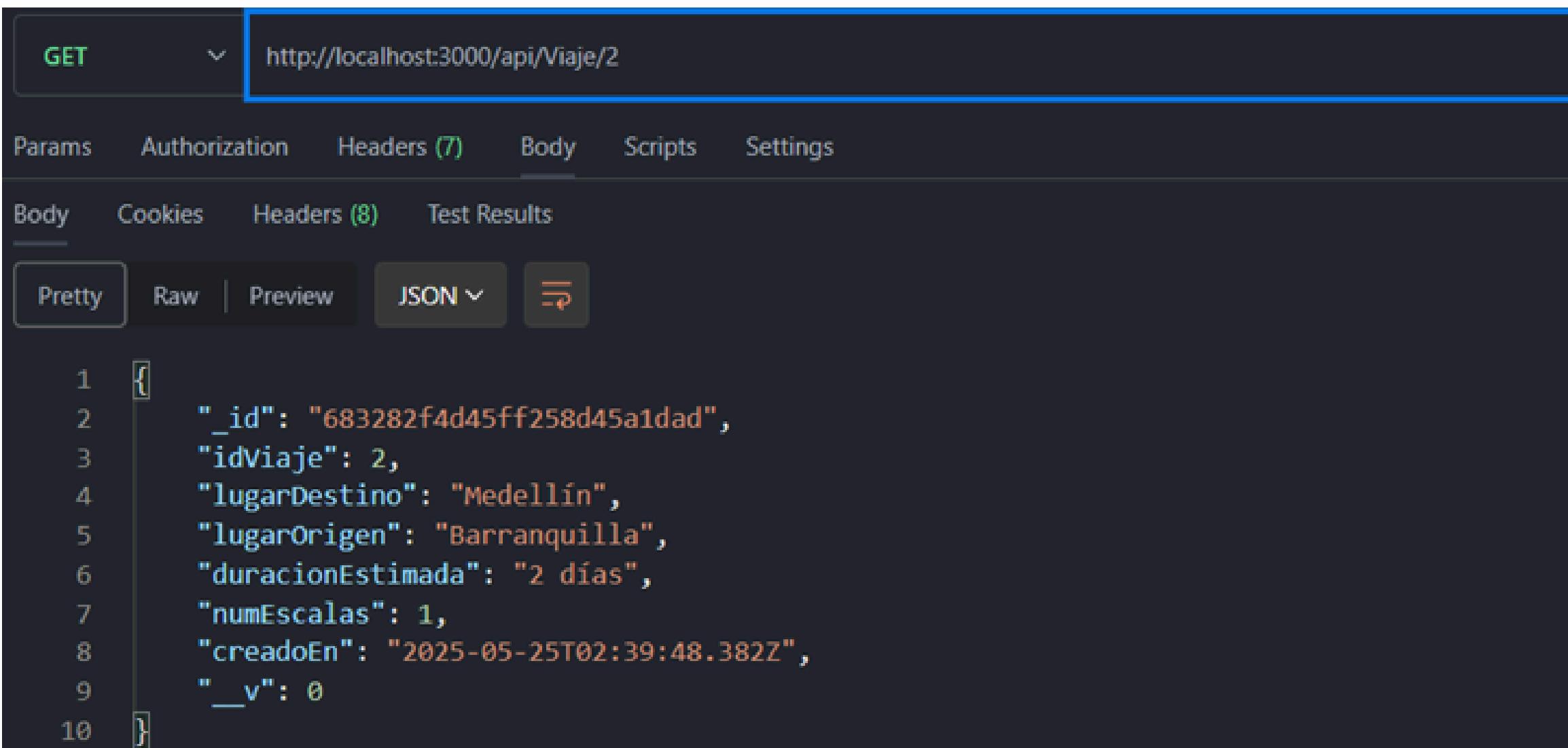
Status: 201 Created Time: 13 ms Size: 9.63 KB

# BASE DE DATOS NOSQL

```
const GastoSchema=new mongoose.Schema({  
    idGasto:Number,  
    descripcionGasto:{type:String,require:true},  
    tipoGasto:Number,  
    nombreGasto:String,  
    creadoEn:{type:Date,default:Date.now}  
});
```



# BASE DE DATOS NOSQL



The screenshot shows a Postman interface with a successful API call. The method is 'GET' and the URL is 'http://localhost:3000/api/Viaje/2'. The response body is displayed in JSON format, showing a single trip document with various fields like \_id, idViaje, lugarDestino, lugarOrigen, duracionEstimada, numEscalas, and creadoEn.

```
1 {  
2   "_id": "683282f4d45ff258d45a1dad",  
3   "idViaje": 2,  
4   "lugarDestino": "Medellin",  
5   "lugarOrigen": "Barranquilla",  
6   "duracionEstimada": "2 días",  
7   "numEscalas": 1,  
8   "creadoEn": "2025-05-25T02:39:48.382Z",  
9   "__v": 0  
10 }
```

# BASE DE DATOS NOSQL

```
router.put('/soat/:idVehiculo', async (req, res) => {
  try {
    const { placaVehiculo, SoatVehiculo } = req.body;

    if (!placaVehiculo && !SoatVehiculo) {
      return res.status(400).json({ error: 'Debe enviar al menos placaVehiculo o SoatVehiculo para actualizar' });
    }

    const updateData = {};
    if (placaVehiculo) updateData.placaVehiculo = placaVehiculo;
    if (SoatVehiculo) updateData.SoatVehiculo = SoatVehiculo;

    const vehiculo = await Vehiculo.findOneAndUpdate(
      { idVehiculo: Number(req.params.idVehiculo) },
      updateData,
      { new: true }
    );

    if (!vehiculo) {
      return res.status(404).json({ error: 'Vehículo no encontrado' });
    }

    res.json(vehiculo);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

The screenshot shows a Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:3000/api/Vehiculo/Soat/1
- Body:** raw JSON [{"SoatVehiculo": "activo"}]
- Headers:** Headers (8)
- Status:** Status: 200 OK
- Body (Pretty):**

4	"placaVehiculo": "SLF799",
5	"colorVehiculo": "Rojo",
6	"marcaVehiculo": "KIA",
7	"cantidadReparaciones": 1,
8	"estadoVehiculo": "Activo",
9	"valorImpuesto": 95000,
10	"SoatVehiculo": "activo",
11	"creadoEn": "2025-05-25T02:38:43.162Z",

# BASE DE DATOS NOSQL

```
router.get('/juridicos', async (req, res) => {
  try {
    const clientesJuridicos = await Cliente.find(
      { tipoCliente: 'juridico' },
      { _id: 0, idCliente: 1, nombreCliente: 1, nitCliente: 1 }
    );

    if (clientesJuridicos.length === 0) {
      return res.status(404).json({ error: 'No se encontraron clientes jurídicos' });
    }

    res.json(clientesJuridicos);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

The screenshot shows a Postman interface with a successful API call. The method is GET, the URL is <http://localhost:3000/api/Cliente/juridicos>, and the response body is displayed in JSON format.

Body (Pretty):

```
[{"idCliente": 1, "nombreCliente": "TransRisaralda"}, {"idCliente": 2, "nombreCliente": "CCM Ingenieros"}, {"idCliente": 3, "nombreCliente": "INVERTRANS"}, {"idCliente": 4, "nombreCliente": "OLT"}]
```

# BASE DE DATOS NOSQL

```
router.put('/duracion/:idViaje', async (req, res) => {
  try {
    const { duracionEstimada } = req.body;
    if (!duracionEstimada) {
      return res.status(400).json({ error: 'Falta el campo duracionEstimada' });
    }

    const actualizado = await Viaje.findOneAndUpdate(
      { idViaje: Number(req.params.idViaje) },
      { duracionEstimada: duracionEstimada },
      { new: true }
    );

    if (!actualizado) {
      return res.status(404).json({ error: 'Viaje no encontrado' });
    }

    res.json(actualizado);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

The screenshot shows a POSTMAN interface with a PUT request to `http://localhost:3000/api/Viaje/duracion/1`. The request body is set to raw JSON with the value `{"duracionEstimada": "10 segundos"}`. The response body is displayed in pretty JSON format, showing a travel document with the updated estimated duration.

```
1 {  
2   "_id": "683282f4d45ff258d45a1dac",  
3   "idViaje": 1,  
4   "lugarDestino": "Bogotá",  
5   "lugarOrigen": "Pasto",  
6   "duracionEstimada": "10 segundos",  
7   "numEscalas": 2,  
8   "creadoEn": "2025-05-25T02:39:48.382Z",  
9   "v": 0  
}
```

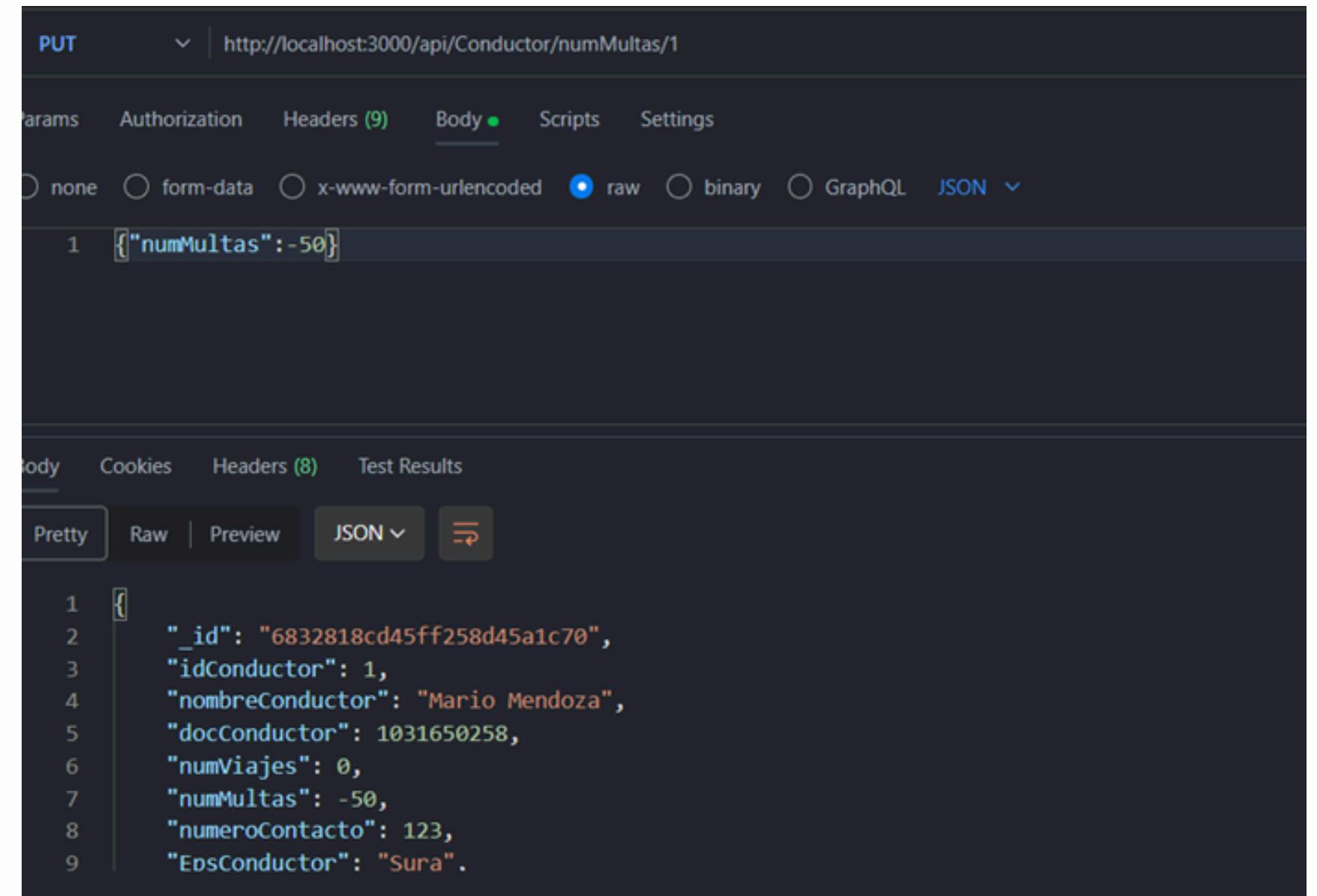
# BASE DE DATOS NOSQL

```
router.put('/numMultas/:idConductor', async (req, res) => {
  try {
    const { numMultas } = req.body;
    if (numMultas === undefined) {
      return res.status(400).json({ error: 'Falta el campo numMultas' });
    }

    const actualizado = await Conductor.findOneAndUpdate(
      { idConductor: Number(req.params.idConductor) },
      { numMultas: numMultas },
      { new: true }
    );

    if (!actualizado) {
      return res.status(404).json({ error: 'Conductor no encontrado' });
    }

    res.json(actualizado);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```



# BASE DE DATOS NOSQL

```
router.delete('/:idCliente', async (req, res) => {
  try {
    const id = Number(req.params.idCliente);
    const deletedCliente = await Cliente.findOneAndDelete({ idCliente: id });

    if (!deletedCliente) {
      return res.status(404).json({ error: 'Cliente no encontrado' });
    }

    res.json({ mensaje: 'Cliente eliminado correctamente', cliente: deletedCliente });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

module.exports=router;

DELETE | http://localhost:3000/api/Cliente/1

Params Authorization Headers (7) Body Scripts Settings

( none ( form-data ( x-www-form-urlencoded ( raw ( binary ( GraphQL JSON

1

Body Cookies Headers (8) Test Results

Status: 200 OK

Pretty Raw Preview JSON

1 {  
2 "mensaje": "Cliente eliminado correctamente",  
3 "cliente": {  
4 "\_id": "68328091d45ff258d45a1c3f",  
5 "idCliente": 1,  
6 "docCliente": 1023456789,  
7 "contactoCliente": "3124567890",  
8 "tipoCliente": "juridico",  
9 "nombreCliente": "TransRisaralda".

**GRACIAS**