



DESARROLLO DE APLICACIONES WEB

ING. GEOVANNY BRAVO, MG.

ACTIVIDAD AUTÓNOMA # 2

UNIDAD 3: ARQUITECTURA ORIENTADA A MICROSERVICIOS WEB Y
MANEJO DE ESTADOS

ANTONIO JOSÉ VILLÓN YAGUAL
21 DE DICIEMBRE DEL 2024

PRÁCTICA SOBRE UNA API REST PARA INSERTAR, MODIFICAR, ELIMINAR.

1. LA API SERÁ MODIFICADA DEL TRABAJO PRESENTADO ANTERIORMENTE

Configuración de la Base de Datos (app/Config/Database.php):

Las credenciales de PostgreSQL siguen siendo las mismas:

```
27 public $default = [  
28     'DSN' => '',  
29     'hostname' => 'localhost',  
30     'username' => 'postgres', // Usuario de PostgreSQL  
31     'password' => 'postgres', // Contraseña de PostgreSQL  
32     'database' => 'tienda',  
33     'DBDriver' => 'Postgre',  
34     'DBPrefix' => '',  
35     'pConnect' => false,  
36     'DBDebug' => (ENVIRONMENT !== 'production'),  
37     'cacheOn' => false,  
38     'cacheDir' => '',  
39     'charset' => 'utf8',  
40     'DBCollat' => 'utf8_general_ci',  
41     'swapPre' => '',  
42     'encrypt' => false,  
43     'compress' => false,  
44     'strictOn' => false,  
45     'failover' => [],  
46     'port' => 5432  
47 ];  
48
```

Controlador

Se modifica ApiController.php en app/Controllers.

```
app > Controllers > ApiController.php  
1 <?php  
2  
3 namespace App\Controllers;  
4 use CodeIgniter\RESTful\ResourceController;  
5 use App\Models\ClienteModel;  
6 use App\Models\PedidoModel;  
7  
8 class ApiController extends ResourceController  
9 {  
10     protected $modelName = 'App\Models\ClienteModel';  
11     protected $format = 'json';  
12  
13     // Obtener todos los clientes  
14     public function getClientes()  
15     {  
16         $model = new ClienteModel();  
17         $clientes = $model->findAll();  
18         return $this->respond($clientes);  
19     }  
20
```

```

20
21 // Obtener un cliente por ID
22 public function getCliente($id = null)
23 {
24     $model = new ClienteModel();
25     $cliente = $model->find($id);
26     if ($cliente) {
27         return $this->respond($cliente);
28     } else {
29         return $this->failNotFound('Cliente no encontrado');
30     }
31 }
32
33 // Crear un nuevo cliente
34 public function createCliente()
35 {
36     $model = new ClienteModel();
37     $data = $this->request->getJSON();
38
39     if ($model->insert($data)) {
40         return $this->respondCreated(['message' => 'Cliente creado correctamente']);
41     } else {
42         return $this->failValidationErrors($model->errors());
43     }
44 }
45

```

```

46 // Actualizar un cliente existente
47 public function updateCliente($id = null)
48 {
49     $model = new ClienteModel();
50     $data = $this->request->getJSON();
51
52     if ($model->find($id)) {
53         if ($model->update($id, $data)) {
54             return $this->respond(['message' => 'Cliente actualizado correctamente']);
55         } else {
56             return $this->failValidationErrors($model->errors());
57         }
58     } else {
59         return $this->failNotFound('Cliente no encontrado');
60     }
61 }
62
63 // Eliminar un cliente
64 public function deleteCliente($id = null)
65 {
66     $model = new ClienteModel();
67
68     if ($model->find($id)) {
69         $model->delete($id);
70         return $this->respondDeleted(['message' => 'Cliente eliminado correctamente']);
71     } else {
72         return $this->failNotFound('Cliente no encontrado');
73     }
74 }
75
76 // Obtener los pedidos de un cliente
77 public function getPedidos($cliente_id)
78 {
79     $model = new PedidoModel();
80     $pedidos = $model->getPedidosByCliente($cliente_id);
81     return $this->respond($pedidos);
82 }
83 }
84

```

Modelos para Clientes y Pedidos

Siguen siendo los mismos modelos `ClienteModel.php` y `PedidoModel.php` en `app/Models`.

```
app > Models > ClienteModel.php
1  <?php
2
3  namespace App\Models;
4  use CodeIgniter\Model;
5
6  class ClienteModel extends Model
7  {
8      protected $table = 'clientes';
9      protected $primaryKey = 'id';
10     protected $allowedFields = ['nombre', 'correo', 'direccion'];
11 }
12
```

```
app > Models > PedidoModel.php
1  <?php
2
3  namespace App\Models;
4  use CodeIgniter\Model;
5
6  class PedidoModel extends Model
7  {
8      protected $table = 'pedidos';
9      protected $primaryKey = 'id';
10     protected $allowedFields = ['cliente_id', 'producto', 'cantidad', 'fecha'];
11
12     public function getPedidosByCliente($cliente_id)
13     {
14         return $this->where('cliente_id', $cliente_id)->findAll();
15     }
16 }
17
```

Configurar las Rutas

Se define las nuevas rutas para la API en `app/Config/Routes.php`:

```
app > Config > Routes.php
32
33 $routes->get('clientes', 'ApiController::getClientes');
34 $routes->get('cliente/(:num)', 'ApiController::getClientes/$1');
35 $routes->post('cliente', 'ApiController::createCliente'); // Insertar cliente
36 $routes->put('cliente/(:num)', 'ApiController::updateCliente/$1'); // Modificar cliente
37 $routes->delete('cliente/(:num)', 'ApiController::deleteCliente/$1'); // Eliminar cliente
38 $routes->get('pedidos/(:num)', 'ApiController::getPedidos/$1');
39
40
41
```

2. CONECTADO A BASE DE DATOS POSTGRESQL

The screenshot shows a database management tool interface. On the left, a tree view displays the 'tienda' database structure, including 'Catálogos', 'Contenedores de Datos F', 'Conversiones', 'Disparadores por evento', and 'Esquemas (1)'. Under 'Esquemas (1)', the 'public' schema is expanded, showing various objects like 'Aggregates', 'Analizadores FTS', 'Colaciones', 'Configuraciones FT', 'Diccionarios FTS', 'Dominios', 'Funciones', 'Funciones disparad', 'Operators', 'Plantillas FTS', 'Procedimientos', '1.3 Secuencias', and 'Tablas (2)'. The 'Tablas (2)' folder is selected, showing 'clientes' and 'pedidos' tables.

The main window displays a query editor with the following SQL query:

```
1 select * from pedidos
2
```

The query results are shown in a table with the following columns: id [PK] integer, cliente_id integer, descripcion text, total numeric (10,2), and fecha timestamp without time zone. The results show 7 rows of data:

	id [PK] integer	cliente_id integer	descripcion text	total numeric (10,2)	fecha timestamp without time zone
1	1	1	Compra de electrodomésticos	1200.50	2024-12-14 01:34:22.281789
2	2	1	Compra de muebles	850.00	2024-12-14 01:34:22.281789
3	3	2	Pedido de ropa	250.75	2024-12-14 01:34:22.281789
4	4	3	Pedido de libros	100.00	2024-12-14 01:34:22.281789
5	5	4	Compra de alimentos	300.25	2024-12-14 01:34:22.281789
6	6	5	Pedido de electrónicos	450.99	2024-12-14 01:34:22.281789
7	7	5	Compra de herramientas	620.50	2024-12-14 01:34:22.281789

Datos almacenados en las tablas

The screenshot shows the same database management tool interface. The query editor displays the following SQL query:

```
1 select * from clientes
2
```

The query results are shown in a table with the following columns: id [PK] integer, nombre character varying (100), email character varying (100), and telefono character varying (15). The results show 5 rows of data:

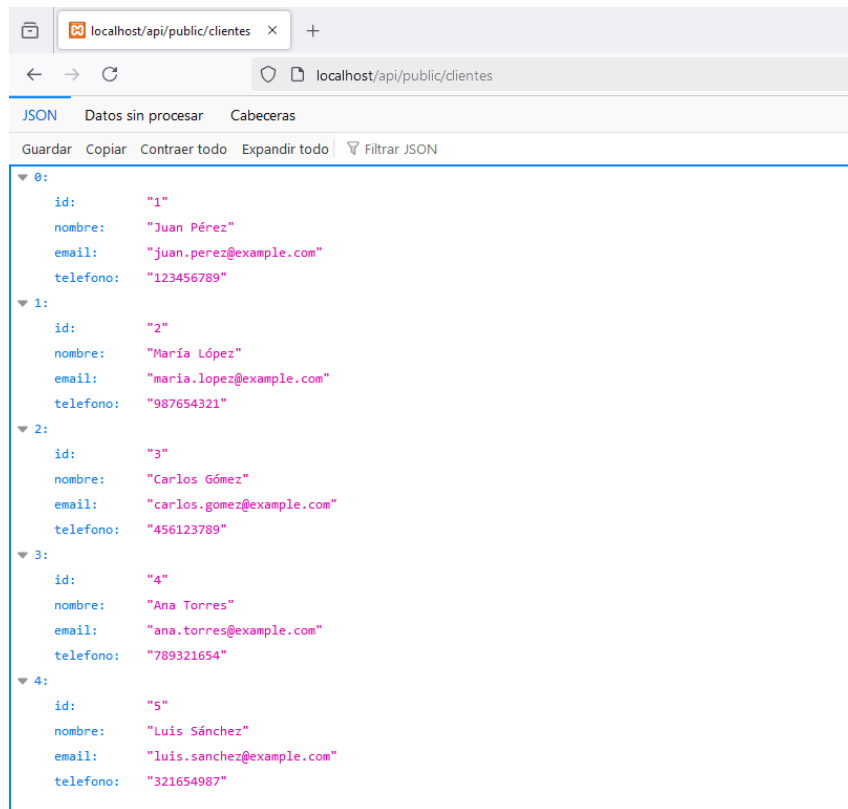
	id [PK] integer	nombre character varying (100)	email character varying (100)	telefono character varying (15)
1	1	Juan Pérez	juan.perez@example.com	123456789
2	2	María López	maria.lopez@example.com	987654321
3	3	Carlos Gómez	carlos.gomez@example.co...	456123789
4	4	Ana Torres	ana.torres@example.com	789321654
5	5	Luis Sánchez	luis.sanchez@example.com	321654987

3. RESULTADOS DE LA MODIFICACIÓN DE LA API

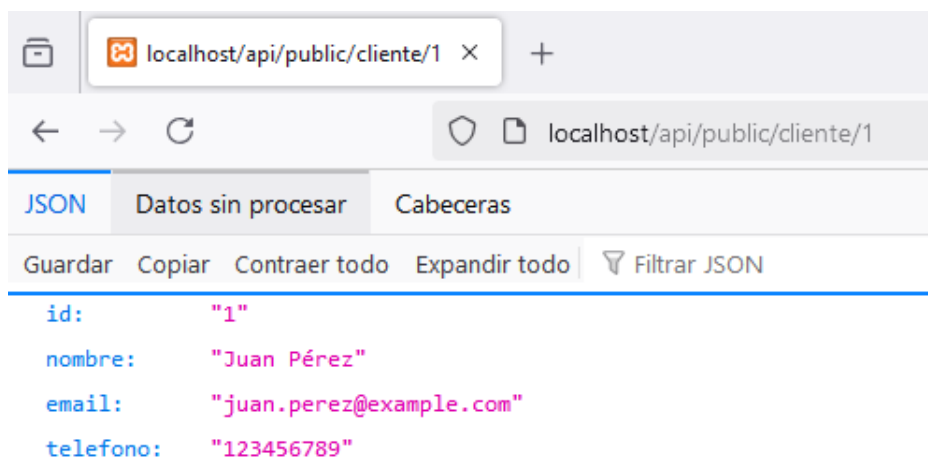
Probar la API

En el navegador se prueba las siguientes URLs:

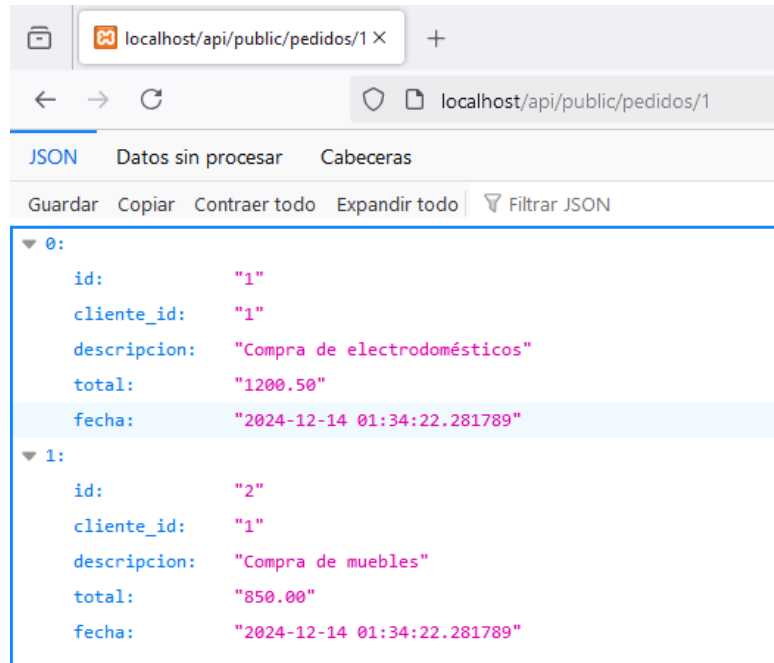
- <http://localhost/api/public/clientes> (obtendrá todos los clientes)



- <http://localhost/api/public/cliente/1> (obtendrá el cliente con ID 1)

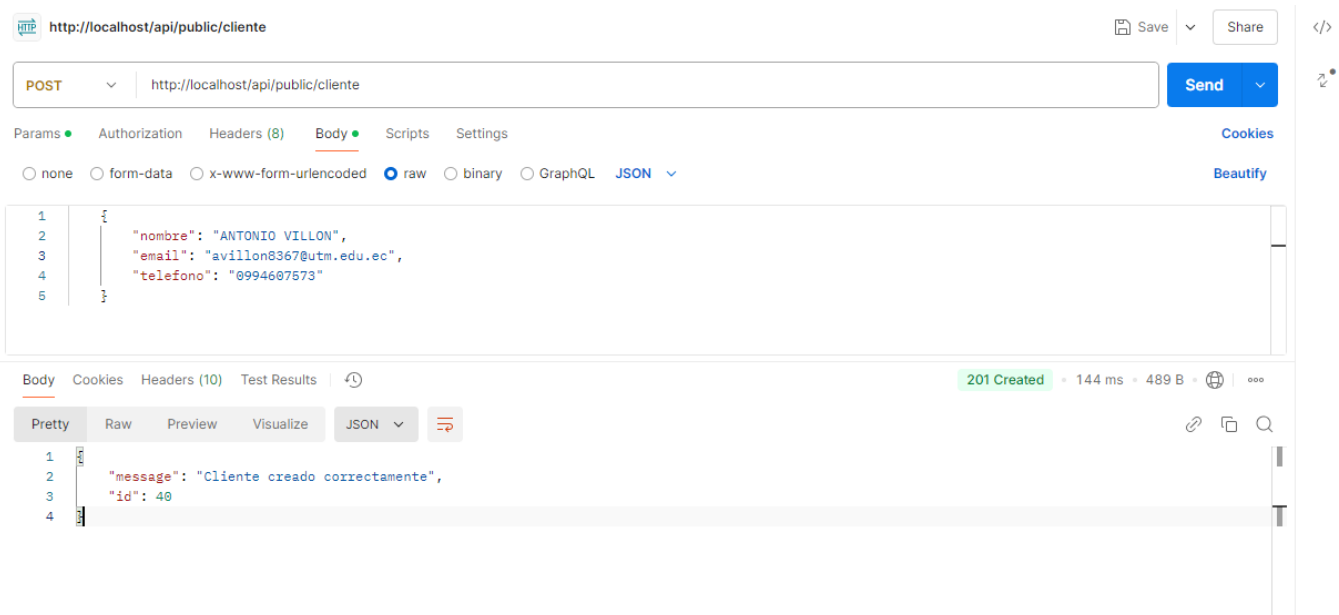


- <http://localhost/api/public/pedidos/1> (obtendrá los pedidos del cliente con ID 1)



Pruebas en Postman para los resultados esperados

INSERTAR




```

26     {
27         "id": "5",
28         "nombre": "Luis Sánchez",
29         "email": "luis.sanchez@example.com",
30         "telefono": "321654987"
31     },
32     {
33         "id": "40",
34         "nombre": "ANTONIO VILLON",
35         "email": "avillon8367@utm.edu.ec",
36         "telefono": "0994607573"
37     }
38 ]

```

MODIFICAR

 **http://localhost/api/public/cliente/40**

PUT
▼

http://localhost/api/public/cliente/40

Params ●

Authorization

Headers (8)

Body ●

Scripts

Settings

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

```

1     {
2         "nombre": "ANTONIO VILLON",
3         "email": "avillon8367@utm.edu.ec",
4         "telefono": "0999999999"
5     }

```

Body

Cookies

Headers (10)

Test Results

↺

Pretty

Raw

Preview

Visualize

JSON ▼

↻

```

1     {
2         "message": "Cliente actualizado correctamente"
3     }

```

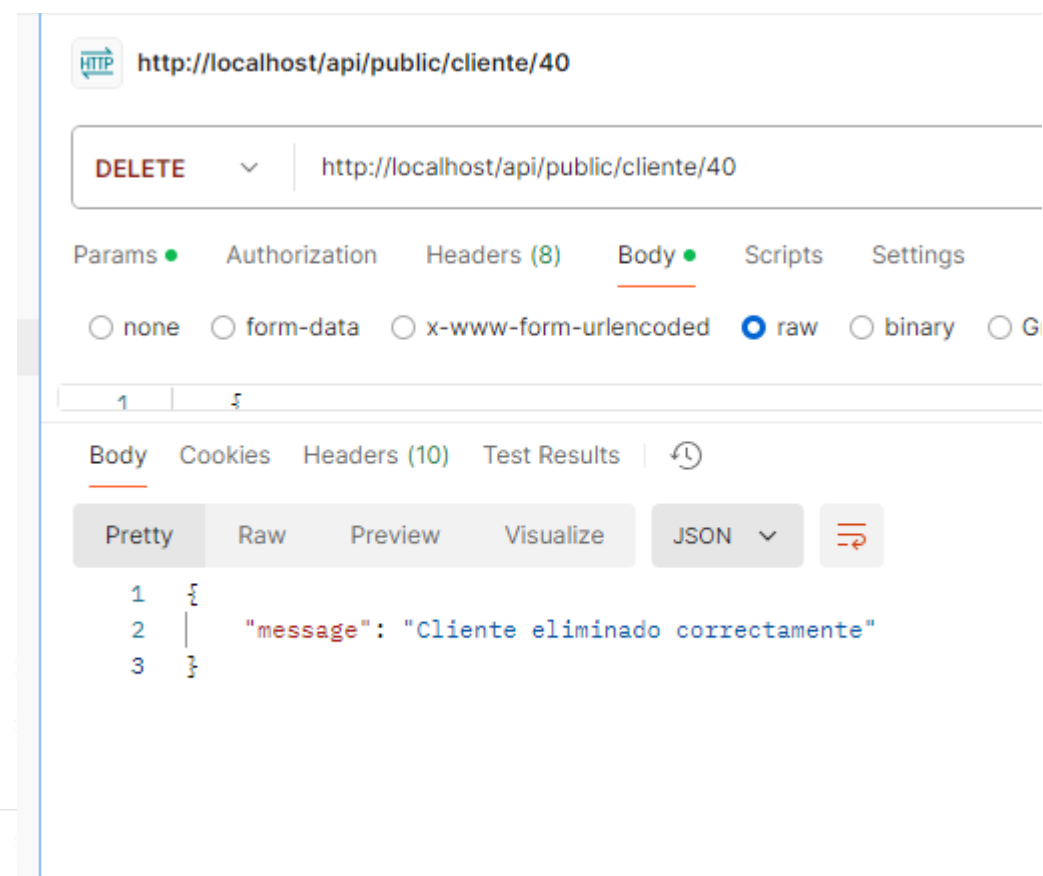


```

25     },
26     {
27         "id": "5",
28         "nombre": "Luis Sánchez",
29         "email": "luis.sanchez@example.com",
30         "telefono": "321654987"
31     },
32     {
33         "id": "40",
34         "nombre": "ANTONIO VILLON",
35         "email": "avillon8367@utm.edu.ec",
36         "telefono": "0999999999"
37     }
38 ]

```

ELIMINAR



HTTP **http://localhost/api/public/cliente/40**

DELETE **http://localhost/api/public/cliente/40**

Params ● Authorization Headers (8) Body ● Scripts Settings

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

1 {

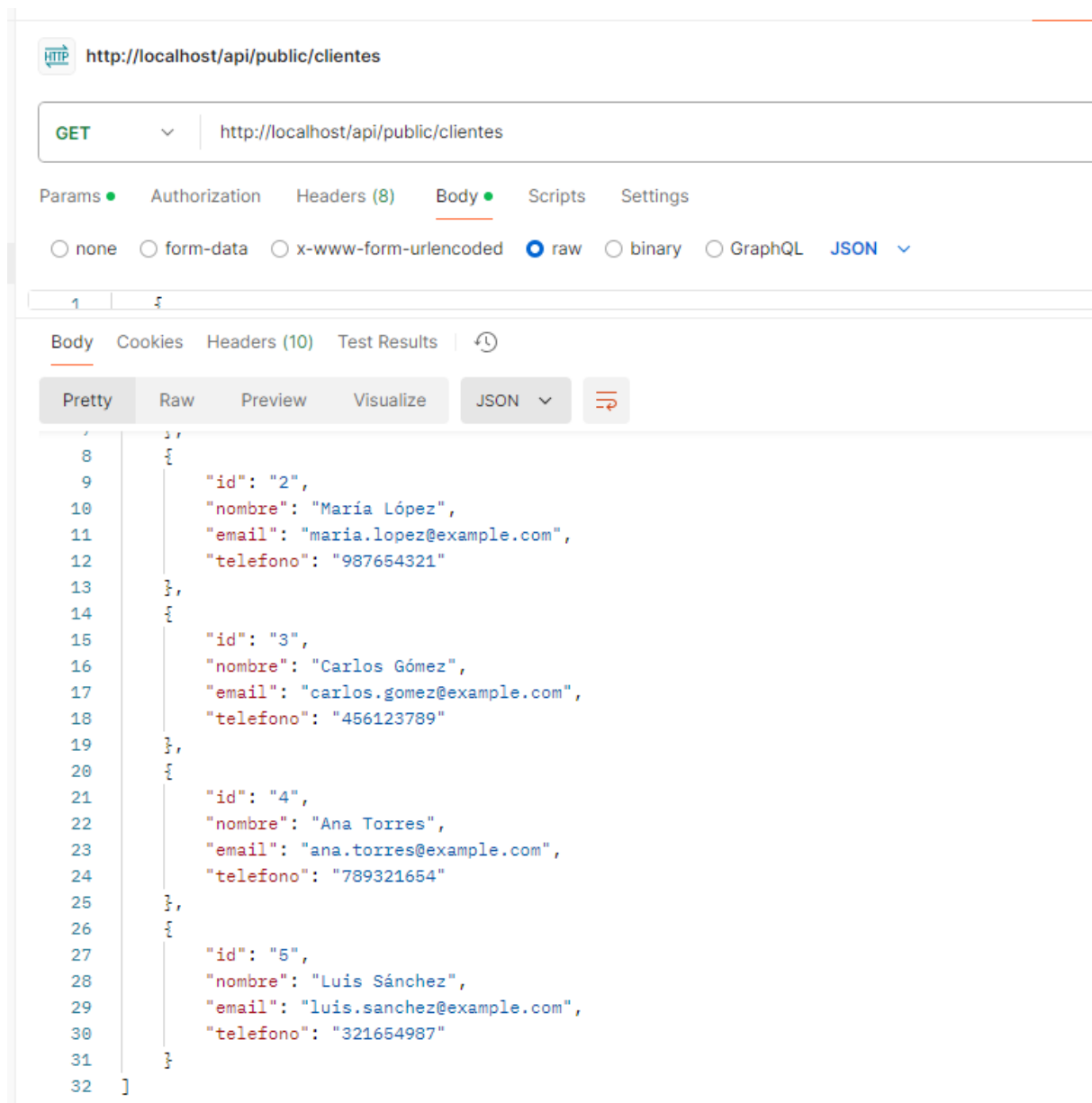
Body Cookies Headers (10) Test Results ↻

Pretty Raw Preview Visualize JSON **JSON** **JSON**

```

1 {
2   "message": "Cliente eliminado correctamente"
3 }

```



4. TAREA SUBIDA A UN GIT Y ENLACE DEL REPOSITORIO

<https://github.com/AntonioVillon/Api-Rest-2.git>