

UNIVERSIDAD MAYOR DE SAN ANDRES
FACULTAD DE CIENCIAS PURAS Y NATURALES

INFORMÁTICA



PRACTICA #2

PROGRAMACION WEB 3

Nombre Completo : ADUVIRI ZAPANA ADRIAN JORGE

C.I. : 14878180

Carrera : INFORMATICA

Materia : PROGRAMACION WEB 3

Fecha : 10/11/2025

1. Crea un endpoint POST /categorias que permita registrar una nueva categoría enviando nombre y descripcion en el body de la petición.

CODIGO

```
5 // 1. Crear categoría
6 router.post('/', (req, res) => {
7   const { nombre, descripcion } = req.body;
8   if (!nombre) return res.status(400).json({ error: 'nombre es requerido' });
9
10  db.query(
11    'INSERT INTO categorias (nombre, descripcion) VALUES (?, ?)',
12    [nombre, descripcion || null],
13    (err, result) => {
14      if (err) return res.status(500).json({ error: err.message });
15      res.status(201).json({ message: 'Categoría creada', id: result.insertId });
16    }
17  );
18 });
19
```

PROCESO

POST	http://localhost:3000/api/categorias	Send	Status: 201 Created	Size: 38 Bytes	Time: 23 ms
Query	Headers ²	Auth	Body ¹	Tests	Pre Run
JSON	XML	Text	Form	Form-encode	GraphQL
JSON Content			Format		
<pre>1 { 2 "nombre": "Dulces", 3 "descripcion": "Productos derivados de la Miel" 4 }</pre>					
			Response	Headers ⁷	Cookies
			<pre>1 { 2 "message": "Categoría creada", 3 "id": 6 4 }</pre>	Results	Docs

RESPUESTA ESPERADA

```
array [3]
  0 {3}
    id : 1
    nombre : "Lácteos"
    descripcion : "Productos derivados de la leche"
  1 {3}
    id : 2
    nombre : "Bebidas"
    descripcion : "Productos derivados de Frutras"
  2 {3}
    id : 3
    nombre : "Dulces"
    descripcion : "Productos derivados de la Miel"
```

2. Crea un endpoint GET /categorias que devuelva todas las categorías registradas en la base de datos.

CODIGO

```
router.get('/', (req, res) => {  
  db.query('SELECT * FROM categorias', (err, results) => {  
    if (err) return res.status(500).json({ error: err });  
    res.json(results);  
  });  
});
```

PROCESO

The screenshot shows a REST client interface. On the left, the 'Query' tab is active, displaying a GET request to 'http://localhost:3000/api/categorias'. The 'Body' tab is also visible. On the right, the 'Response' tab shows the status '200 OK', 'Size: 227 Bytes', and 'Time: 37 ms'. The response body is a JSON array with three objects, each representing a category with 'id', 'nombre', and 'descripcion' fields.

```
1 [
2   {
3     "id": 1,
4     "nombre": "Lácteos",
5     "descripcion": "Productos derivados de la leche"
6   },
7   {
8     "id": 2,
9     "nombre": "Bebidas",
10    "descripcion": "Productos derivados de Frutras"
11  },
12  {
13    "id": 3,
14    "nombre": "Dulces",
15    "descripcion": "Productos derivados de la Miel"
16  }
17 ]
```

RESPUESTA ESPERADA

```
✓ array [3]
  ✓ 0 {3}
    id : 1
    nombre : "Lácteos"
    descripcion : "Productos derivados de la leche"
  ✓ 1 {3}
    id : 2
    nombre : "Bebidas"
    descripcion : "Productos derivados de Frutras"
  ✓ 2 {3}
    id : 3
    nombre : "Dulces"
    descripcion : "Productos derivados de la Miel"
```

3. Crea un endpoint GET /categorias/:id que devuelva la categoría con el ID especificado y además, incluya todos los productos que pertenecen a esa categoría.

CODIGO

```
28 // 3. Obtener una categoría por ID (con sus productos)
29 router.get('/:id', (req, res) => {
30   const categoriaId = req.params.id;
31
32   db.query('SELECT * FROM categorias WHERE id = ?', [categoriaId], (err, catRows) => {
33     if (err) return res.status(500).json({ error: err.message });
34     if (catRows.length === 0) return res.status(404).json({ error: 'Categoría no encontrada' });
35
36     const categoria = catRows[0];
37
38     db.query('SELECT * FROM productos WHERE categoria_id = ?', [categoriaId], (err, prodRows) => {
39       if (err) return res.status(500).json({ error: err.message });
40       res.json({ categoria, productos: prodRows });
41     });
42   });
43 });
```

PROCESO

GET

Status: 200 OK Size: 345 Bytes Time: 88 ms

Query Headers 2 Auth Body Tests Pre Run

Response Headers 7 Cookies Results Docs

Query Parameters

☐ parameter value

```
7 "productos": [
8   {
9     "id": 1,
10    "nombre": "Yogurt natural",
11    "descripcion": "Yogurt de 1L sin azúcar",
12    "precio": "8.50",
13    "stock": 20,
14    "categoria_id": 1
15  },
16  {
17    "id": 2,
18    "nombre": "Mantequilla Natural",
19    "descripcion": "Mantequilla de 800gm",
20    "precio": "9.50",
21    "stock": 10,
22    "categoria_id": 1
23  }
24 ]
```

RESPUESTA ESPERADA

```
object {2}
  categoria {3}
    id : 1
    nombre : "Lácteos"
    descripcion : "Productos derivados de la leche"
  productos [2]
    0 {6}
      id : 1
      nombre : "Yogurt natural"
      descripcion : "Yogurt de 1L sin azúcar"
      precio : "8.50"
      stock : 20
      categoria_id : 1
    1 {6}
      id : 2
      nombre : "Mantequilla Natural"
      descripcion : "Mantequilla de 800gm"
      precio : "9.50"
      stock : 10
      categoria_id : 1
```

4. Crea un endpoint PUT /categorias/:id que permita actualizar todos los datos de la categoría con el ID especificado.

CODIGO

```
45 // 4. Actualizar categoría
46 router.put('/:id', (req, res) => {
47   const id = req.params.id;
48   const { nombre, descripcion } = req.body;
49   if (!nombre) return res.status(400).json({ error: 'nombre es requerido' });
50
51   db.query(
52     'UPDATE categorias SET nombre = ?, descripcion = ? WHERE id = ?',
53     [nombre, descripcion || null, id],
54     (err, result) => {
55       if (err) return res.status(500).json({ error: err.message });
56       if (result.affectedRows === 0) return res.status(404).json({ error: 'Categoría no encontrada' });
57       res.json({ message: 'Categoría actualizada' });
58     }
59   );
60 });
```

PROCESO

The screenshot shows a REST client interface. The method is PUT and the URL is http://localhost:3000/api/categorias/1. The status is 200 OK, size is 36 Bytes, and time is 322 ms. The request body is a JSON object: {"nombre": "Lácteos Actualizados", "descripcion": "Productos derivados de la leche modificados"}. The response body is a JSON object: {"message": "Categoría actualizada"}.

Method	URL	Status	Size	Time
PUT	http://localhost:3000/api/categorias/1	200 OK	36 Bytes	322 ms

Query Headers Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

```
1 {
2   "nombre": "Lácteos Actualizados",
3   "descripcion": "Productos derivados de la leche modificados"
4 }
```

Format

Response Headers Cookies Results Docs

```
1 {
2   "message": "Categoría actualizada"
3 }
```

RESPUESTA ESPERADA

				id	nombre	descripcion
<input type="checkbox"/>		Editar		Copiar		Borrar
1	Lácteos Actualizados		Productos derivados de la leche modificados			
2	Bebidas		Productos derivados de Frutras			
3	Dulces		Productos derivados de la Miel			

5. Crea un endpoint DELETE /categorias/:id que elimine la categoría indicada y, al mismo tiempo, elimine automáticamente todos los productos que pertenecen a esa categoría.

CODIGO

```
63 // 5. Eliminar categoría
64 router.delete('/:id', (req, res) => {
65   const id = req.params.id;
66   db.query('DELETE FROM categorias WHERE id = ?', [id], (err, result) => {
67     if (err) return res.status(500).json({ error: err.message });
68     if (result.affectedRows === 0) return res.status(404).json({ error: 'Categoría no encontrada' });
69     res.json({ message: 'Categoría eliminada (y productos asociados también)' });
70   });
71 });
72
73 module.exports = router;
74
```

PROCESO

DELETE ▼ http://localhost:3000/api/categorias/1 Send

Query Headers² Auth Body Tests Pre Run

Query Parameters


☐ parameter value

Status: 200 OK Size: 67 Bytes Time: 32 ms

Response Headers⁷ Cookies Results Docs {}

```
1 {
2   "message": "Categoría eliminada (y productos asociados también)"
3 }
```

RESPUESTA ESPERADA

				id	nombre	descripcion
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2	Bebidas	Productos derivados de Frutras
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	3	Dulces	Productos derivados de la Miel

```
▼ object {1}
  error : "Categoría no encontrada"
```

(Categoría con id 1 ELIMINADA)

6. Crea un endpoint POST /productos que permita registrar un nuevo producto enviando nombre, precio, stock y categoria_id para asociarlo a una categoría existente.

CODIGO

```
5 // 6. Crear producto
6 router.post('/', (req, res) => {
7   const { nombre, descripcion, precio, stock, categoria_id } = req.body;
8
9   if (!nombre || !precio || !categoria_id) {
10    return res.status(400).json({ error: 'nombre, precio y categoria_id son requeridos' });
11  }
12
13  const sql = 'INSERT INTO productos (nombre, descripcion, precio, stock, categoria_id) VALUES (?, ?, ?, ?, ?)';
14  db.query(sql, [nombre, descripcion || null, precio, stock || 0, categoria_id], (err, result) => {
15    if (err) return res.status(500).json({ error: err.message });
16    res.status(201).json({ message: 'Producto creado', id: result.insertId });
17  });
18 });
19
20 module.exports = router;
21
```

PROCESO

POST ▼ http://localhost:3000/api/productos Send

Query Headers 2 Auth **Body 1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format










```
1 {
2   "nombre": "Gaseosas",
3   "descripcion": "Coca Cola 3L",
4   "precio": 16,
5   "stock": 50,
6   "categoria_id": 1
7 }
8
```

Status: 201 Created Size: 36 Bytes Time: 19 ms

Response Headers 7 Cookies Results Docs

```
1 {
2   "message": "Producto creado",
3   "id": 4
4 }
```

RESPUESTA ESPERADA

<div>←T→</div>			<div>▼</div>	id	nombre	descripcion	precio	stock	categoria_id
<input type="checkbox"/>	<div><div></div> Editar</div>	<div><div></div> Copiar</div>	<div><div></div> Borrar</div>	1	Yogurt natural	Yogurt de 1L sin azúcar	8.50	20	1
<input type="checkbox"/>	<div><div></div> Editar</div>	<div><div></div> Copiar</div>	<div><div></div> Borrar</div>	2	Mantequilla Natural	Mantequilla de 800gm	9.50	10	1
<input type="checkbox"/>	<div><div></div> Editar</div>	<div><div></div> Copiar</div>	<div><div></div> Borrar</div>	4	Gaseosas	Coca Cola 3L	16.00	50	1

7. Crea un endpoint GET /productos que devuelva todos los productos y muestre el nombre de la categoría a la que pertenece cada uno.

CODIGO

```
20 // 7. Obtener todos los productos
21 router.get('/', (req, res) => {
22   const sql = `
23     SELECT p.id, p.nombre, p.descripcion, p.precio, p.stock, c.nombre AS categoria
24     FROM productos p
25     JOIN categorias c ON p.categoria_id = c.id
26   `;
27
28   db.query(sql, (err, results) => {
29     if (err) return res.status(500).json({ error: err.message });
30     res.json(results);
31   });
32 });
33
34
35 module.exports = router;
```

PROCESO

The screenshot shows a web browser with the address bar set to `http://localhost:3000/api/productos`. The status bar indicates a successful response: **Status: 200 OK**, **Size: 467 Bytes**, and **Time: 10 ms**. The **Response** tab is active, displaying a JSON array of two objects. The first object represents a product with ID 1, named 'Yogurt natural', priced at 8.50, with a stock of 20, belonging to the 'Lacteos' category. The second object represents a product with ID 2, named 'Mantequilla Natural', priced at 9.50, with a stock of 10, also belonging to the 'Lacteos' category.

```
1 [
2   {
3     "id": 1,
4     "nombre": "Yogurt natural",
5     "descripcion": "Yogurt de 1L sin azúcar",
6     "precio": "8.50",
7     "stock": 20,
8     "categoria": "Lacteos"
9   },
10  {
11    "id": 2,
12    "nombre": "Mantequilla Natural",
13    "descripcion": "Mantequilla de 800gm",
14    "precio": "9.50",
15    "stock": 10,
16    "categoria": "Lacteos"
17  },
18 ]
```

RESPUESTA ESPERADA

```
✓ array [4]
  ✓ 0 {6}
    id : 1
    nombre : "Yogurt natural"
    descripcion : "Yogurt de 1L sin azúcar"
    precio : "8.50"
    stock : 20
    categoria : "Lacteos"
  ✓ 1 {6}
    id : 2
    nombre : "Mantequilla Natural"
    descripcion : "Mantequilla de 800gm"
    precio : "9.50"
    stock : 10
    categoria : "Lacteos"
  ✓ 2 {6}
    id : 3
    nombre : "Gaseosas"
    descripcion : "Coca Cola 3L"
    precio : "16.00"
    stock : 50
    categoria : "Bebidas"
  ✓ 3 {6}
    id : 4
    nombre : "Alikal"
    descripcion : "Dulce de menta"
    precio : "0.50"
    stock : 200
    categoria : "Dulces"
```


8. Crea un endpoint GET /productos/:id que devuelva la información de un producto por su ID, incluyendo el nombre de la categoría asociada.

CODIGO

```
34 // 8. Obtener un producto por su ID
35 router.get('/:id', (req, res) => {
36   const id = req.params.id;
37
38   const sql = `
39     SELECT p.id, p.nombre, p.descripcion, p.precio, p.stock, c.nombre AS categoria
40     FROM productos p
41     LEFT JOIN categorias c ON p.categoria_id = c.id
42     WHERE p.id = ?
43   `;
44
45   db.query(sql, [id], (err, results) => {
46     if (err) return res.status(500).json({ error: err.message });
47     if (results.length === 0) return res.status(404).json({ error: 'Producto no encontrado' });
48     res.json(results[0]);
49   });
50 });
```

PROCESO

GET	http://localhost:3000/api/productos/4	Send	Status: 200 OK	Size: 106 Bytes	Time: 8 ms
Query	Headers 2	Auth	Body	Tests	Pre Run
Query Parameters			Response		
<input type="checkbox"/> parameter			1 {		
			2 "id": 4,		
			3 "nombre": "Alikal",		
			4 "descripcion": "Dulce de menta",		
			5 "precio": "0.50",		
			6 "stock": 200,		
			7 "categoria": "Dulces"		
			8 }		

RESPUESTA ESPERADA

```
localhost:3000/api/productos/4
object {6}
  id : 4
  nombre : "Alikal"
  descripcion : "Dulce de menta"
  precio : "0.50"
  stock : 200
  categoria : "Dulces"
```

```
localhost:3000/api/productos/5
object {1}
  error : "Producto no encontrado"
```

9. Crea un endpoint PUT /productos/:id que permita actualizar todos los datos de un producto, incluyendo la opción de cambiar la categoría a la que pertenece mediante categoria_id.

CODIGO

```
52 // 9. Actualizar un producto
53 router.put('/:id', (req, res) => {
54   const id = req.params.id;
55   const { nombre, descripcion, precio, stock, categoria_id } = req.body;
56
57   if (!nombre || !precio || !categoria_id) {
58     return res.status(400).json({ error: 'nombre, precio y categoria_id son requeridos' });
59   }
60
61   const sql = `
62     UPDATE productos
63     SET nombre = ?, descripcion = ?, precio = ?, stock = ?, categoria_id = ?
64     WHERE id = ?
65   `;
66
67   db.query(sql, [nombre, descripcion || null, precio, stock || 0, categoria_id, id], (err, result) => {
68     if (err) return res.status(500).json({ error: err.message });
69     if (result.affectedRows === 0) return res.status(404).json({ error: 'Producto no encontrado' });
70     res.json({ message: 'Producto actualizado correctamente' });
71   });
72 });
73
```

PROCESO

PUT <http://localhost:3000/api/productos/1> Send

Status: 200 OK Size: 48 Bytes Time: 44 ms

Query Headers² Auth **Body¹** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nombre": "Yogurt frutado",
3   "descripcion": "Yogurt sabor frutilla 1L",
4   "precio": 10,
5   "stock": 25,
6   "categoria_id": 1
7 }
```

Response Headers⁷ Cookies Results Docs

```
1 {
2   "message": "Producto actualizado correctamente"
3 }
```

RESPUESTA ESPERADA

<div><div>←T→</div><div></div></div>				id	nombre	descripcion	precio	stock	categoria_id
<input type="checkbox"/>				1	Yogurt frutado	Yogurt sabor frutilla 1L	10.00	25	1
<input type="checkbox"/>				2	Mantequilla Natural	Mantequilla de 800gm	9.50	10	1
<input type="checkbox"/>				3	Gaseosas	Coca Cola 3L	16.00	50	2
<input type="checkbox"/>				4	Alikal	Dulce de menta	0.50	200	3

[←](#) [↺](#) [ⓘ](#) localhost:3000/api/productos/1

✓ object {6}

```
id : 1
nombre : "Yogurt frutado"
descripcion : "Yogurt sabor frutilla 1L"
precio : "10.00"
stock : 25
categoria : "Lacteos"
```

10. Crea un endpoint PATCH /productos/:id/stock que permita incrementar o decrementar el stock de un producto enviando al body la cantidad que se desea sumar o restar.

CODIGO

```
74 // 10. Actualizar el stock de un producto
75 router.patch('/:id/stock', (req, res) => {
76   const id = req.params.id;
77   const { cantidad } = req.body;
78
79   if (typeof cantidad !== 'number') {
80     return res.status(400).json({ error: 'Debes enviar un número en "cantidad"' });
81   }
82
83   // Primero obtenemos el stock actual
84   db.query('SELECT stock FROM productos WHERE id = ?', [id], (err, results) => {
85     if (err) return res.status(500).json({ error: err.message });
86     if (results.length === 0) return res.status(404).json({ error: 'Producto no encontrado' });
87
88     const stockActual = results[0].stock;
89     const nuevoStock = stockActual + cantidad;
90
91     if (nuevoStock < 0) {
92       return res.status(400).json({ error: 'El stock no puede ser negativo' });
93     }
94
95     // Actualizamos el stock
96     db.query('UPDATE productos SET stock = ? WHERE id = ?', [nuevoStock, id], (err2) => {
97       if (err2) return res.status(500).json({ error: err2.message });
98       res.json({ message: 'Stock actualizado correctamente', stockAnterior: stockActual, stockNuevo: nuevoStock });
99     });
100   });
101 });
```

PROCESO

PATCH ⌵ http://localhost:3000/api/productos/3/stock Send

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format













```
1 {
2   "cantidad": 5
3 }
4
```

Status: 200 OK Size: 80 Bytes Time: 61 ms

Response Headers ⁷ Cookies Results Docs

```
1 {
2   "message": "Stock actualizado correctamente",
3   "stockAnterior": 50,
4   "stockNuevo": 55
5 }
```

RESPUESTA ESPERADA

<div>← T →</div>				id	nombre	descripcion	precio	stock	categoria_id
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	Yogurt frutado	Yogurt sabor frutilla 1L	10.00	25	1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2	Mantequilla Natural	Mantequilla de 800gm	9.50	10	1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	3	Gaseosas	Coca Cola 3L	16.00	55	2
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	4	Alikal	Dulce de menta	0.50	200	3

```
{
  "message": "Stock actualizado correctamente",
  "stockAnterior": 50,
  "stockNuevo": 55
}
```