



Instituto Politécnico Nacional

Omar Montoya Romero

7CM1

**WEB CLIENT AND BACKEND DEVELOPMENT
FRAMEWORKS**

Sistemas Computacionales

Ejercicio 03.- API de la Tabla de Categorías

Mtro. Efraín Arredondo Morales

24/09/2023

Primero inicializamos el proyecto con el comando `npm init -y`, después instalamos `npm install express`, una vez instaladas las librerías podemos seguir configurando el servidor, el puerto lo configuramos de la siguiente manera `process.env.port` el cual nos sirve para que al momento de subirlo a render o algún otro servicio, el servidor tome como endpoint la liga que da al subirlo, al mismo tiempo se pone que si no está ese endpoint se prenda en el puerto 3000.

```
const express = require("express");
const app = express();
const port = process.env.port || 3000;
app.use(express.json()); //Habilitacion para recibir datos por medio de solicitud
(insomnia)
```

Creamos el arreglo de las categorías donde cada uno tiene su id, nombre, y descripción. Esto para poder hacer las consultas con los métodos GET, POST, PUT, DELETE

```
// Arreglo de objetos de categorías
let categorias = [
  { id: 1, nombre: "Cocina", descripcion: "Elementos para cocinar" },
  { id: 2, nombre: "Limpieza", descripcion: "Elementos para limpiar" },
  { id: 3, nombre: "Electronica", descripcion: "Elementos de electronica" },
  { id: 4, nombre: "Ropa bebe", descripcion: "Elementos para bebe" },
  { id: 5, nombre: "Linea Blanca", descripcion: "Elementos de linea blanca" },
  { id: 6, nombre: "Jardinera", descripcion: "Elementos de jardineria" },
  { id: 7, nombre: "Salud", descripcion: "Elementos para la salud" },
  { id: 8, nombre: "Muebles", descripcion: "Elementos para la sala y demas" },
  { id: 9, nombre: "Lacteos", descripcion: "Elementos para beber" },
  { id: 10, nombre: "Licores", descripcion: "Elementos para fiestas" },
  { id: 528, nombre: "Licores", descripcion: "Elementos para fiestas" },
];
```

Empezamos a implementar los diferentes métodos:

- Obtener la lista de todos los productos (GET).
- Obtener un producto por su ID (GET).
- Agregar un nuevo producto (POST).
- Actualizar un producto por su ID (PUT).
- Eliminar un producto por su ID (DELETE).

Primero tenemos el método de consulta GET para obtener todas las categorías, el cual hace una comprobación para saber si el arreglo esta lleno o vacío, en el caso de que este lleno muestra la lista de las categorías, mientras que si esta vacío arroja el mensaje de “No existen categorías”

```
// Obtener la lista de todos los productos (GET).
app.get("/socios/v1/categorias", (req, res) => {
  //1.- Verificar si existen las categorías
  if (categorias.length > 0) {
    //2.- Mostrarlas con un estado y mensaje
    res.status(200).json({
      estado: 1,
      mensaje: "Existen categorías",
      categorias: categorias,
    });
  } else {
    //3.- No existe, mostrar estado y mensaje
    res.status(404).json({
      estado: 0,
      mensaje: "No existen categorías",
    });
  }
});
```

El segundo método es usando el GET pero por id, donde el id lo consigue de los params, pero al mismo tiempo hace la comprobación de que el id este en la lista de categorías, en caso de que si este muestra la categoría más un mensaje que dice “Categoría encontrada” mientras si el id ingresado no se encuentra lanza un mensaje “No existe categoría”.

```
// Obtener un producto por su ID (GET).
app.get("/socios/v1/categorias/:id", (req, res) => {
  //Solo una categoría
  const id = req.params.id;
  const categoria = categorias.find((categorias) => categorias.id == id);
  if (categoria) {
    res.status(200).json({
      estado: 1,
      mensaje: "Categoría encontrada",
      categoria: categoria,
    });
  } else {
    res.status(404).json({
      estado: 0,
      mensaje: "No existen categoría",
    });
  }
});
```

El tercer método a usar es el POST para agregar una nueva categoría, para crear una nueva categoría el id se le asigna aleatorio de 0 a 1000, esto para evitar que se repitan, el nombre y descripción son recibidos por el body, el cual se ingresa en insomnia para poder crear la categoría, primero se comprueba que nombre o categoría no sean undefined, en caso de que sea undefined manda una respuesta de BAD REQUEST, en caso contrario para saber si se agregó correctamente se compara la longitud inicial del arreglo con la nueva al agregar la categoría, si el arreglo es mayor al estado inicial arroja el mensaje de "Categoría creada correctamente", mientras si no es mayor al original arroja de que "No se creó correctamente".

```
// Agregar un nuevo producto (POST).
app.post("/socios/v1/categorias", (req, res) => {
  //Crear un recurso - Crear una categoria
  // Requerimos
  // id = Generar un numero aleatorio
  // nombre y descripcion = body

  const { nombre, descripcion } = req.body;
  const id = Math.round(Math.random() * 1000);
  // Comprobar que el cliente = usuario = programador
  if (nombre == undefined || descripcion == undefined) {
    res.status(400).json({
      estado: 0,
      mensaje: "BAD REQUEST Faltan parametros en la solicitud",
    });
  } else {
    //En javascript como agregar un nuevo elemento al arreglo
    const categoria = { id: id, nombre: nombre, descripcion: descripcion };
    const longitudInicial = categorias.length;
    categorias.push(categoria);
    if (categorias.length > longitudInicial) {
      res.status(201).json({
        estado: 1,
        mensaje: "Categoría creada correctamente",
        categoria: categoria,
      });
    } else {
      res.status(500).json({
        estado: 0,
        mensaje: "No se agrego correctamente",
      });
    }
  }
});
```

Pasamos al tercer método PUT para actualizar una categoría por id, donde el id se obtiene de los params, el nombre y descripción del body.

```
// Actualizar un producto por su ID (PUT).
app.put("/socios/v1/categorias/:id", (req, res) => {
  //id viene ? = params
  // nombre y descripción ? = body
  const { id } = req.params;
  const { nombre, descripcion } = req.body;
  if (nombre == undefined && descripcion == undefined) {
    res.status(400).json({
      estado: 0,
      mensaje: "BAD REQUEST Faltan parametros en la solicitud",
    });
  } else {
    const posActualizar = categorias.findIndex(
      (categoria) => categoria.id == id
    );
    if (posActualizar != -1) {
      //Si encuentro la categoria con el id buscado
      //Actualizar la categoria
      categorias[posActualizar].nombre = nombre;
      categorias[posActualizar].descripcion = descripcion;
      res.status(200).json({
        estado: 1,
        mensaje: "Categoria actualizada correctamente",
        categoria: categorias[posActualizar],
      });
    } else {
      //No se encontro la categoria con el id buscado
      res.status(404).json({
        estado: 0,
        mensaje: "No se actualizo",
      });
    }
  }
});
```

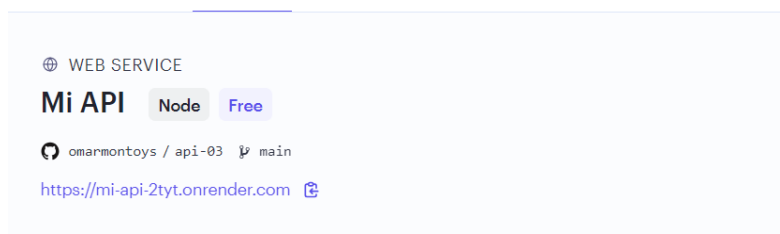
Pasamos al tercer método PUT para actualizar una categoría por id, donde el id se obtiene de los params.

```
// Eliminar un producto por su ID (DELETE).
app.delete("/socios/v1/categorias/:id", (req, res) => {
  //Solo una categoria
  const { id } = req.params;
  const indiceEliminar = categorias.findIndex(
    (categoria) => categoria.id == id
  );
  if (indiceEliminar !== -1) {
    //Borrar la categoria
    categorias.splice(indiceEliminar, 1);
    res.status(201).json({
      estado: 1,
      mensaje: "Categoria eliminada correctamente",
    });
  } else {
    res.status(404).json({
      estado: 0,
      mensaje: "No se elimino",
    });
  }
});
```

Cuando encendemos el servidor desde la terminal el listen lo usamos para saber en que puerto se encendio y ver si esta jalando correctamente.

```
app.listen(port, () => {
  console.log("Ejecutandose en el servidor: ", port);
});
```

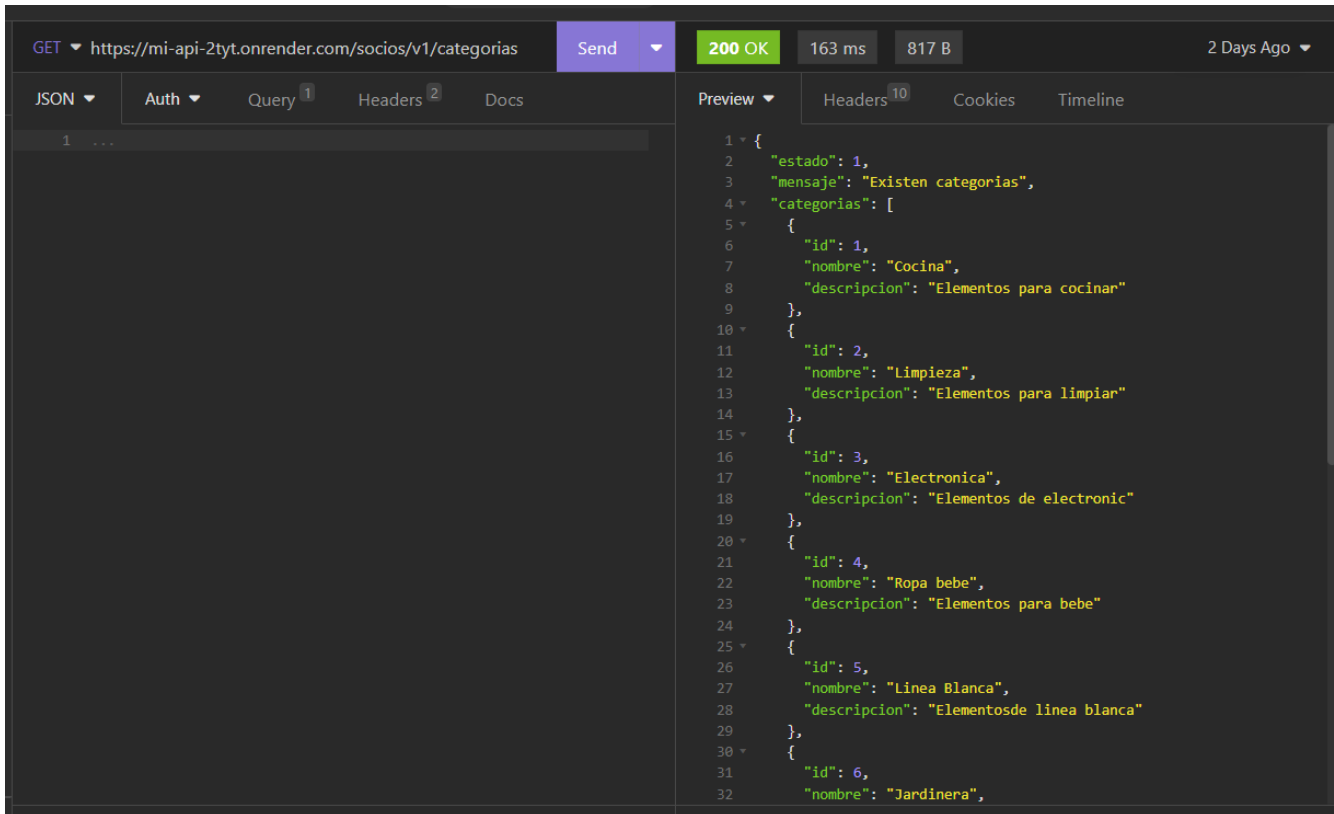
Despues de hacerlo jalar en la terminal pasamos a implementarlo en Render



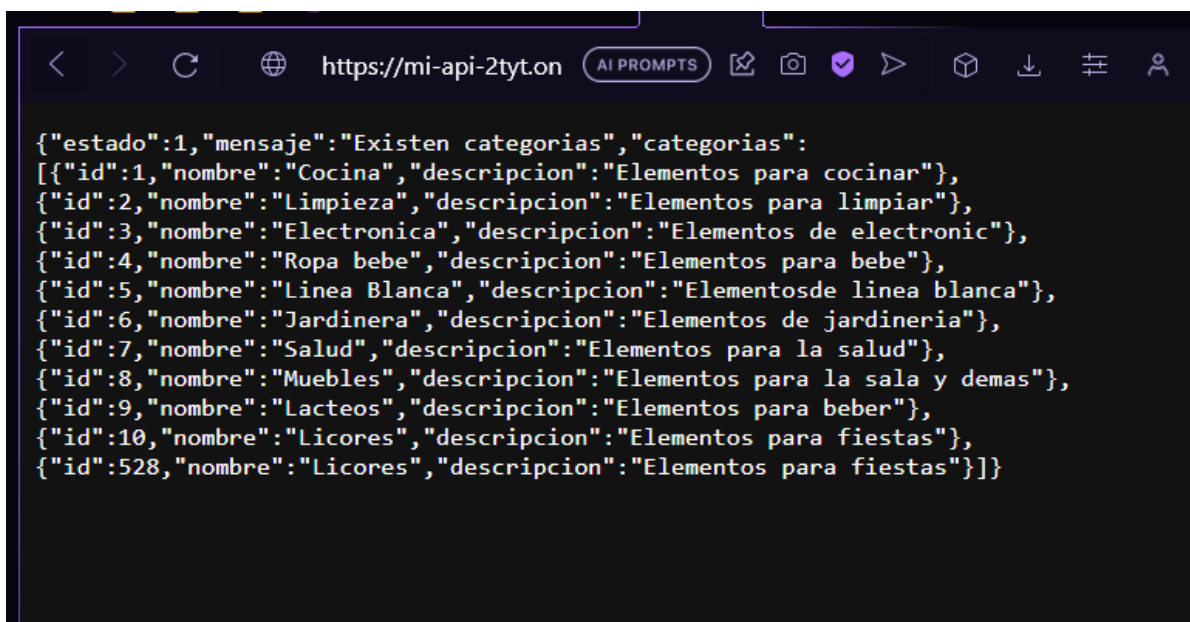
```
Sep 22 11:30:29 AM ==> Using Node version 14.17.0 (default)
Sep 22 11:30:29 AM ==> Docs on specifying a Node version: https://render.com/docs/node-version
Sep 22 11:30:29 AM ==> Running 'node app.js'
Sep 22 11:30:30 AM Ejecutandose en el servidor: 3000
Sep 22 11:30:37 AM ==> Detected service running on port 3000
Sep 22 11:30:37 AM ==> Docs on specifying a port: https://render.com/docs/web-services#port-detection
Sep 22 11:31:11 AM ==> Using Node version 14.17.0 (default)
Sep 22 11:31:11 AM ==> Docs on specifying a Node version: https://render.com/docs/node-version
Sep 22 11:31:11 AM ==> Running 'node app.js'
Sep 22 11:31:11 AM Ejecutandose en el servidor: 3000
```

Al tener el servidor en línea gracias a render procedemos a usarlo en insomnia y al mismo tiempo se puede visualizar en el navegador

Utilizando el método GET



Utilizando el método GET



Utilizando el método GET por ID

GET <https://mi-api-2tyt.onrender.com/socios/v1/categorias/2> Send 200 OK 665 ms 125 B Just Now

JSON Auth Query Headers Docs

```
1 {
  "estado": 1,
  "mensaje": "Categoria encontrada",
  "categoria": {
    "id": 2,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  }
}
```

Utilizando el método POST

POST <https://mi-api-2tyt.onrender.com/socios/v1/categorias> Send 201 Created 196 ms 146 B Just Now

JSON Auth Query Headers Docs

```
1 {
  "nombre": "Deportes",
  "descripcion": "Area donde se encuentran balones"
}
```

Preview Headers Cookies Timeline

```
1 {
  "estado": 1,
  "mensaje": "Categoria creada correctamente",
  "categoria": {
    "id": 75,
    "nombre": "Deportes",
    "descripcion": "Area donde se encuentran balones"
  }
}
```

Consultamos todas las categorías y efectivamente se creo una nueva categoría

GET <https://mi-api-2tyt.onrender.com/socios/v1/categorias> Send 200 OK 709 ms 887 B Just Now

JSON Auth Query Headers Docs

```
1 [
  {
    "id": 1,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 2,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 3,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 4,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 5,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 6,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 7,
    "nombre": "Limpieza",
    "descripcion": "Elementos para limpiar"
  },
  {
    "id": 8,
    "nombre": "Muebles",
    "descripcion": "Elementos para la sala y demas"
  },
  {
    "id": 9,
    "nombre": "Lacteos",
    "descripcion": "Elementos para beber"
  },
  {
    "id": 10,
    "nombre": "Licores",
    "descripcion": "Elementos para fiestas"
  },
  {
    "id": 528,
    "nombre": "Licores",
    "descripcion": "Elementos para fiestas"
  },
  {
    "id": 75,
    "nombre": "Deportes",
    "descripcion": "Area donde se encuentran balones"
  }
]
```

Utilizando el método PUT

PUT <https://mi-api-2tyt.onrender.com/socios/v1/categorias/1> Send 200 OK 192 ms 129 B Just Now

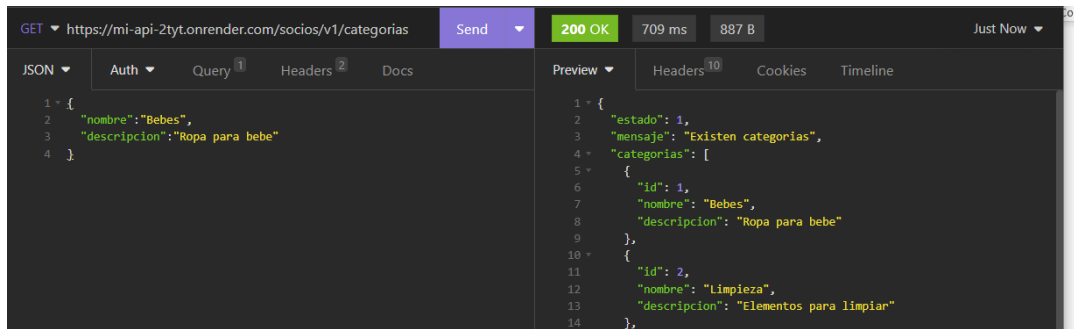
JSON Auth Query Headers Docs

```
1 {
  "nombre": "Bebes",
  "descripcion": "Ropa para bebe"
}
```

Preview Headers Cookies Timeline

```
1 {
  "estado": 1,
  "mensaje": "Categoria actualizada correctamente",
  "categoria": {
    "id": 1,
    "nombre": "Bebes",
    "descripcion": "Ropa para bebe"
  }
}
```


Consultamos todas las categorías y efectivamente se actualizo la Categoría



```
GET https://mi-api-2tyt.onrender.com/socios/v1/categorias 200 OK 709 ms 887 B Just Now
```

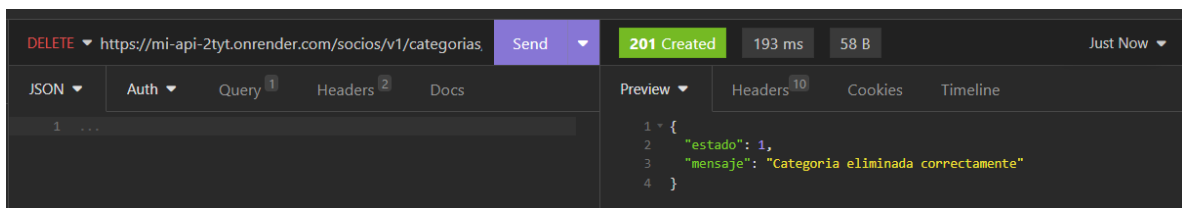
```
JSON Auth Query Headers Docs
```

```
1 {
2   "nombre": "Bebes",
3   "descripcion": "Ropa para bebe"
4 }
```

```
Preview Headers Cookies Timeline
```

```
1 {
2   "estado": 1,
3   "mensaje": "Existen categorias",
4   "categorias": [
5     {
6       "id": 1,
7       "nombre": "Bebes",
8       "descripcion": "Ropa para bebe"
9     },
10    {
11      "id": 2,
12      "nombre": "Limpieza",
13      "descripcion": "Elementos para limpiar"
14    },
15  ]
16 }
```

Utilizando el método DELETE, Eliminamos la categoría 528



```
DELETE https://mi-api-2tyt.onrender.com/socios/v1/categorias 201 Created 193 ms 58 B Just Now
```

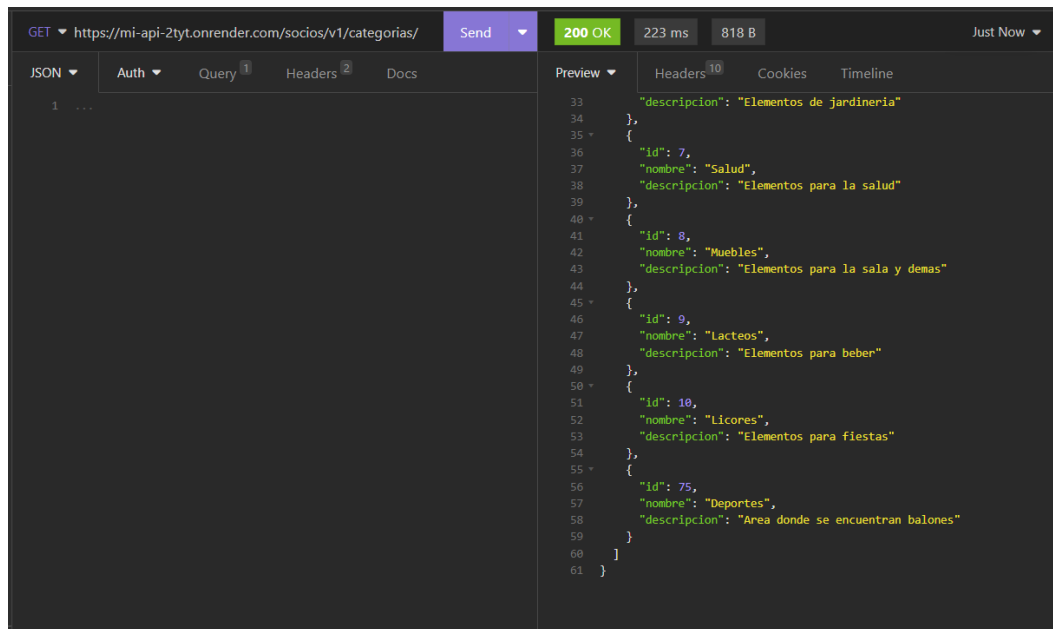
```
JSON Auth Query Headers Docs
```

```
1 ...
```

```
Preview Headers Cookies Timeline
```

```
1 {
2   "estado": 1,
3   "mensaje": "Categoría eliminada correctamente"
4 }
```

Consultamos todas las categorías y efectivamente se Elimino la categoría



```
GET https://mi-api-2tyt.onrender.com/socios/v1/categorias/ 200 OK 223 ms 818 B Just Now
```

```
JSON Auth Query Headers Docs
```

```
1 ...
```

```
Preview Headers Cookies Timeline
```

```
33   "descripcion": "Elementos de jardineria"
34 },
35 {
36   "id": 7,
37   "nombre": "Salud",
38   "descripcion": "Elementos para la salud"
39 },
40 {
41   "id": 8,
42   "nombre": "Muebles",
43   "descripcion": "Elementos para la sala y demas"
44 },
45 {
46   "id": 9,
47   "nombre": "Lacteos",
48   "descripcion": "Elementos para beber"
49 },
50 {
51   "id": 10,
52   "nombre": "Licores",
53   "descripcion": "Elementos para fiestas"
54 },
55 {
56   "id": 75,
57   "nombre": "Deportes",
58   "descripcion": "Area donde se encuentran balones"
59 }
60 ]
61 }
```