

**Facultad:**

Ciencias de la vida y Tecnología

**Carrera:**

Ingeniería en Tecnologías de la  
información

**Tema:**

**Implementación Individual de Servicio Web – OData**

**Estudiante:**

Carlos Johan Macias Sosa

## Método Utilizado Para Mostrar Todo el listado de Productos

<http://localhost:3000/odata/productos> La URL <http://localhost:3000/odata/productos> hace una solicitud GET al servidor Express, lo que indica que el cliente desea obtener datos, en este caso, todos los productos almacenados. Esta solicitud es manejada por el enrutador en `productos.js`, donde la ruta `/` está asociada con la función `controller.listar`. Dentro de esta función, se realiza una consulta a la base de datos para obtener todos los productos, lo cual generalmente se logra a través de un ORM o una consulta SQL como `SELECT * FROM productos`. Los productos obtenidos se envían como respuesta al cliente en formato JSON, mostrando detalles como `id`, `nombre` y `precio`. De esta manera, cuando se accede a la URL, se visualizan todos los productos registrados en la base de datos, ya que la función `listar` simplemente recupera y devuelve todos los datos sin ningún filtro adicional.

The screenshot shows a web browser window with the address bar displaying `http://localhost:3000/odata/productos`. The browser's developer tools are open, showing the network tab with a single request. The request is a GET method to the same URL. The response is a 200 OK status with a response time of 19 ms and a size of 498 B. The response body is displayed in JSON format, showing a list of five products:

```
[{"id": 1, "nombre": "Portátil ", "precio": 5000}, {"id": 2, "nombre": "Tableta", "precio": 100}, {"id": 3, "nombre": "Ratón", "precio": 25}, {"id": 4, "nombre": "Monitor", "precio": 15}, {"id": 5, "nombre": "Teclado", "precio": 45}]
```

The browser's taskbar at the bottom shows various application icons, including Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and a system tray with the date 06/06/2025 and time 20:09.

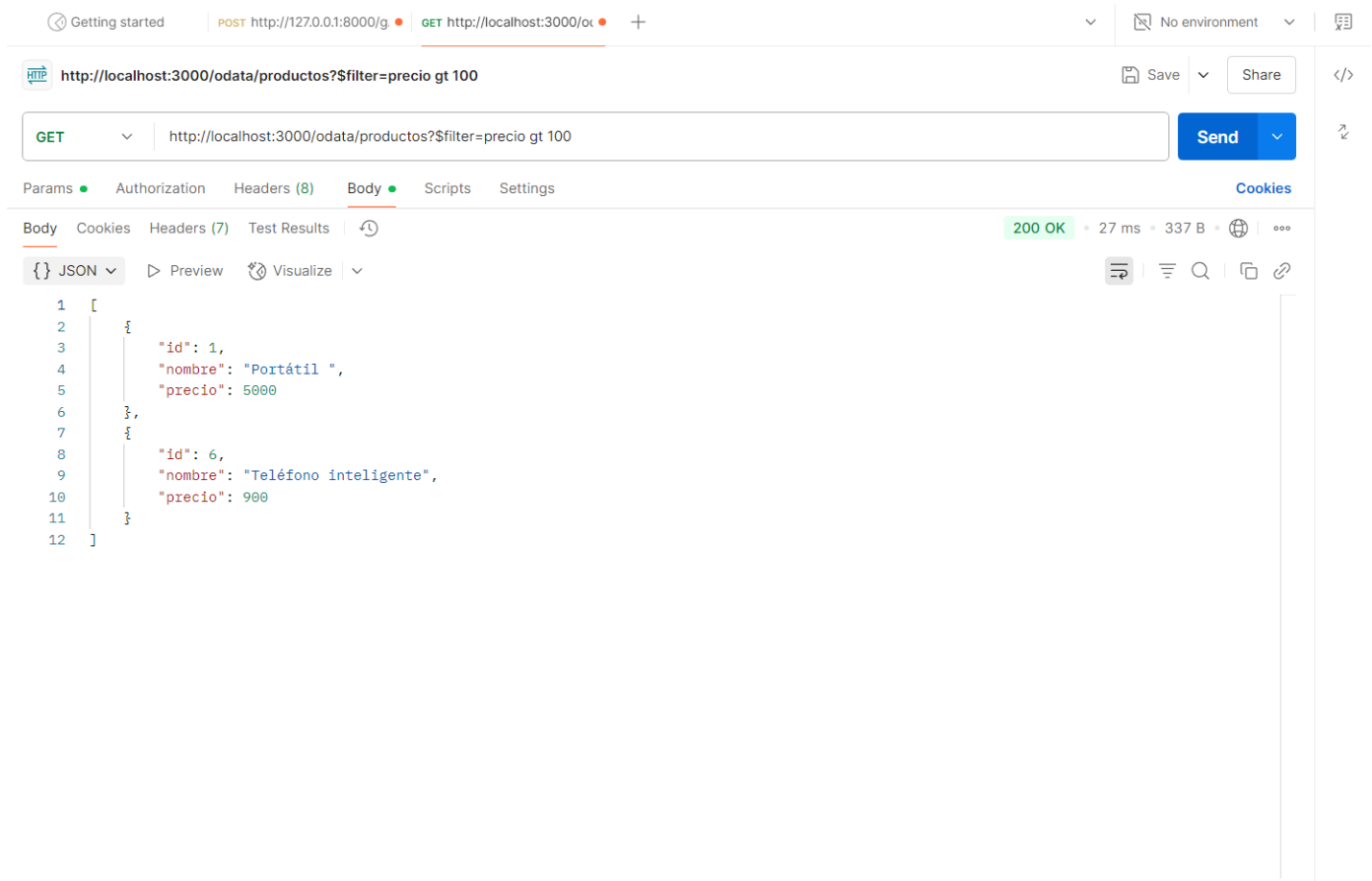
## Metodo para ver productos mayores 100

`http://localhost:3000/productos?$filter=precio gt 100` La URL

`http://localhost:3000/productos?$filter=precio gt 100` realiza una solicitud GET al servidor Express con un filtro adicional en la cadena de consulta (`$filter=precio gt 100`), lo que indica que se deben recuperar solo los productos cuyo precio sea mayor a 100.

Esta solicitud es procesada por el enrutador en `productos.js`, donde la ruta `/` también está asociada con la función `controller.listar`, pero en este caso, se pasa un parámetro de filtro.

La función `listar` debe interpretar este filtro y aplicarlo a la consulta a la base de datos, generalmente transformando la expresión `$filter=precio gt 100` en una condición SQL o similar, como `WHERE precio > 100`. Al realizar la consulta, el servidor obtiene solo los productos que cumplen con la condición de tener un precio mayor a 100 y los devuelve en formato JSON al cliente, mostrando únicamente aquellos productos que cumplen con el filtro especificado.



## Método para ver productos menor que 50

`http://localhost:3000/productos?$filter=precio lt 50`

La URL `http://localhost:3000/odata/productos?$filter=precio lt 50` realiza una solicitud GET al servidor Express con un filtro adicional en la cadena de consulta (`$filter=precio lt 50`), lo que indica que la API debe devolver solo los productos cuyo precio sea menor a 50. Al igual que en el caso anterior, esta solicitud es gestionada por el enrutador en `productos.js`, donde la ruta `/` se asocia con la función `controller.listar`. Esta función recibe el parámetro de filtro y lo interpreta, aplicándolo a la consulta de la base de datos para seleccionar solo aquellos productos con un precio inferior a 50. En términos de base de datos, esto se traduce generalmente en una condición SQL como `WHERE precio < 50`. El servidor procesa la consulta, filtra los productos que cumplen con esta condición y envía la respuesta en formato JSON, conteniendo solo los productos cuyo precio sea menor a 50, asegurando que la respuesta sea precisa según el filtro solicitado.

The screenshot shows a REST client interface with a GET request to `http://localhost:3000/odata/productos?$filter=precio lt 50`. The response is a JSON array of three products:

```
1 {
2   {
3     "id": 3,
4     "nombre": "Ratón",
5     "precio": 25
6   },
7   {
8     "id": 4,
9     "nombre": "Monitor",
10    "precio": 15
11  },
12  {
13    "id": 5,
14    "nombre": "Teclado",
15    "precio": 45
16  }
17 }
```

The interface also shows the status `200 OK`, a response time of `5 ms`, and a response size of `356 B`. The bottom of the image shows a Windows taskbar with various icons and the system clock displaying `20:21` on `06/06/2025`.

## Mi código Funcional:

The screenshot displays the Visual Studio Code editor interface. The left sidebar shows the Explorer view with the project structure for 'PROYECTO-ODATA-MAIN'. The main editor area shows the file 'producto.js' with the following JavaScript code:

```
1 // Simulación de una base de datos con un arreglo de productos tecnológicos
2
3 // Arreglo de productos simulando una base de datos con 10 elementos
4 const productos = [
5   { id: 1, nombre: "Portátil ", precio: 5000 },
6   { id: 2, nombre: "Tableta", precio: 100 },
7   { id: 3, nombre: "Ratón", precio: 25 },
8   { id: 4, nombre: "Monitor", precio: 15 },
9   { id: 5, nombre: "Teclado", precio: 45 },
10  { id: 6, nombre: "Teléfono inteligente", precio: 900 },
11 ];
12
13 // Exportamos el arreglo para usarlo en otros archivos como server.js
14 module.exports = productos;
15
16
```

The bottom panel shows the TERMINAL view with the following output:

```
> odata-demo@1.0.0 start
> node server.js

API corriendo en http://localhost:3000/odata/productos
PS C:\Users\johan\Desktop\Proyecto-ODATA-main> npm start

> odata-demo@1.0.0 start
> node server.js

API corriendo en http://localhost:3000/odata/productos
PS C:\Users\johan\Desktop\Proyecto-ODATA-main> npm start

> odata-demo@1.0.0 start
> node server.js

API corriendo en http://localhost:3000/odata/productos
```

The status bar at the bottom indicates the current file is 'producto.js' at line 8, column 41, with 2 spaces, UTF-8 encoding, and LF line endings. The language is set to JavaScript. The system tray shows the date and time as 20:23 on 06/06/2025.

