

Universidad de nariño

Ingenieria de sistemas

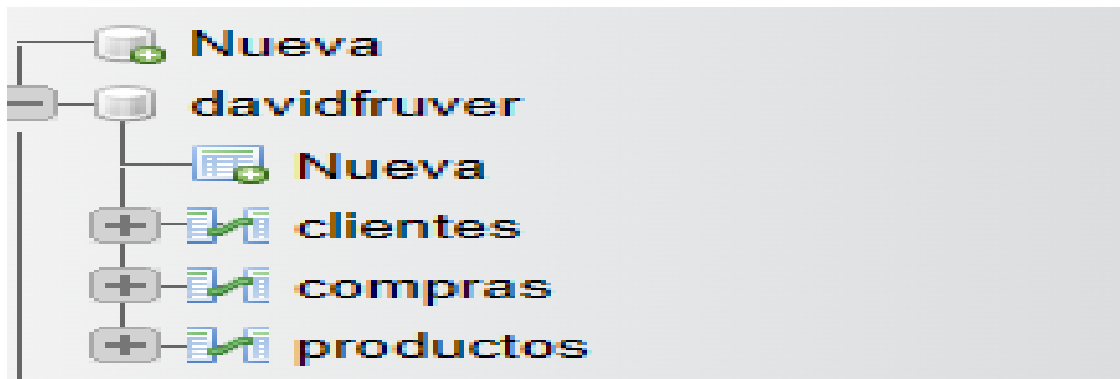
Diplomado de actualizacion en nuevas tecnologias para el desarrollo de
software

Desarrollo del taller 2 Backend

David Alberto Garcia Portocarrero

1. Creamos una base de datos en Mysql que permita llevar el registro de un FRUVER (FRUTAS Y VERDURAS), así como también el proceso de solicitud de compras

Creamos la base de datos llamada davidfruver con sus respetivas tablas productos, compras y clientes . para realizar los procesos que necesitamos para desarrollar nuestro taller




Esta es la tabla productos donde se le inserte unos registros

idproducto	nombrequiproducto	detalleproducto	precioproducto
3	platano	dominico	1500
4	mango	cimaron	3000
8			0

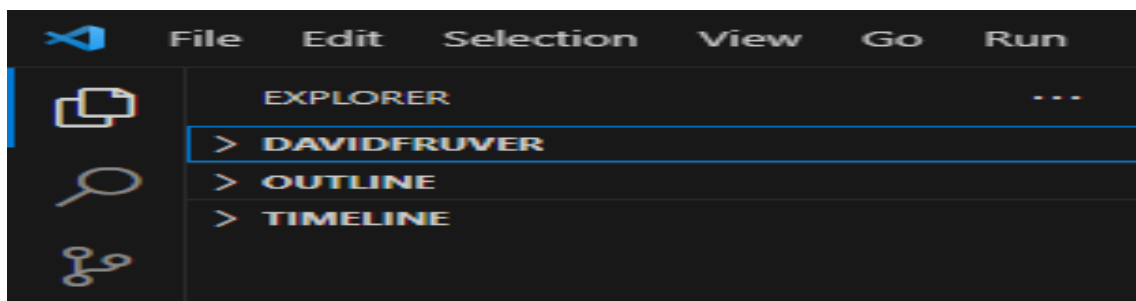
Tabla compras

<input type="checkbox"/>	1	numero compra 	int	No	Ninguna	AUTO_INCREMENT
<input type="checkbox"/>	2	descripcion	varchar(25) utf8mb4_0900_ai_ci	No	Ninguna	
<input type="checkbox"/>	3	cantidad	int	No	Ninguna	
<input type="checkbox"/>	4	precio	int	No	Ninguna	
<input type="checkbox"/>	5	total	int	No	Ninguna	

Tabla clientes

<input type="checkbox"/>	1	cedula 	int	No	Ninguna	
<input type="checkbox"/>	2	nombres	varchar(25) utf8mb4_0900_ai_ci	No	Ninguna	
<input type="checkbox"/>	3	apellidos	varchar(25) utf8mb4_0900_ai_ci	No	Ninguna	
<input type="checkbox"/>	4	telefono	int	No	Ninguna	
<input type="checkbox"/>	5	direccion	varchar(25) utf8mb4_0900_ai_ci	No	Ninguna	
<input type="checkbox"/>	6	Email	varchar(25) utf8mb4_0900_ai_ci	No	Ninguna	

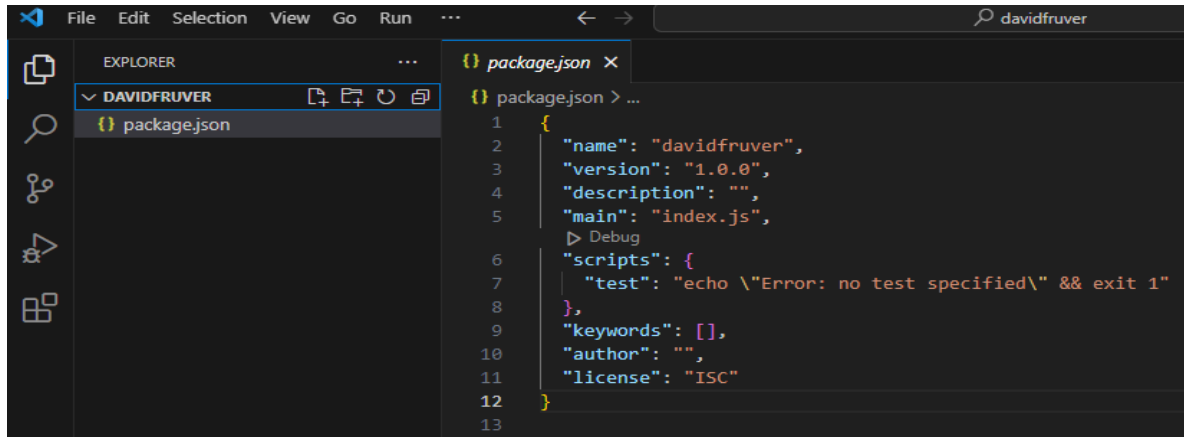
2. Creamos un directorio llamado davidfruver



Ejecutamos el comando `npm init -y`, para instalar el manejador de paquetes de nodejs

```
PS C:\Users\lidam\OneDrive\Documentos\David Diplomado\modulo 2\davidfruver> npm init -y
```

Una vez ejecutamos la instrucción anterior nos crea el archivo package.json así como lo muestra la siguiente imagen



Continuando con las instalaciones procedemos a instalar los paquetes ExpressJS y Mysql para ello vamos a la terminal de VisualStudio code y le damos el siguiente comando npm install express

```
PS C:\Users\lidam\OneDrive\Documentos\David Diplomado\modulo 2\davidfruver> npm install express
```

Una vez instalado express nos podemos dar cuenta que se crean los archivos node_modules con todos los módulos y package-lock.json el archivo de configuración, y como lo muestra el archivo de package.json se adiciona la dependencia express dentro del mismo

De igual forma procedemos a instalar Mysql y Mysql2 para que no tengamos conflictos con la versión

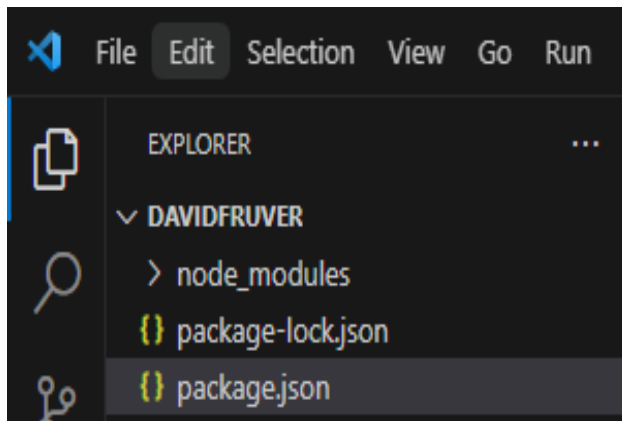
```
PS C:\Users\lidam\OneDrive\Documentos\David Diplomado\modulo 2\davidfruver> npm install mysql
```

```
PS C:\Users\lidam\OneDrive\Documentos\David Diplomado\modulo 2\davidfruver> npm install mysql2
```

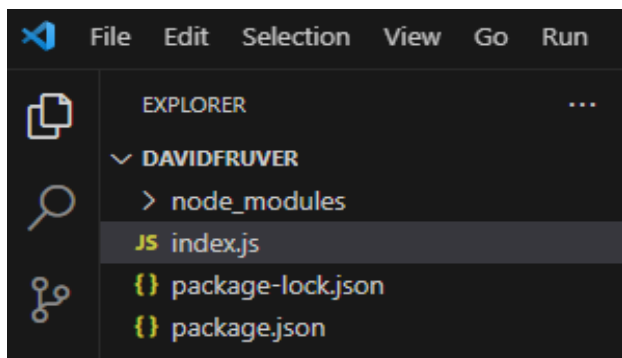
Se instala sequelize

```
PS C:\Users\lidam\OneDrive\Documentos\David Diplomado\modulo 2\davidfruver> npm install sequelize
```

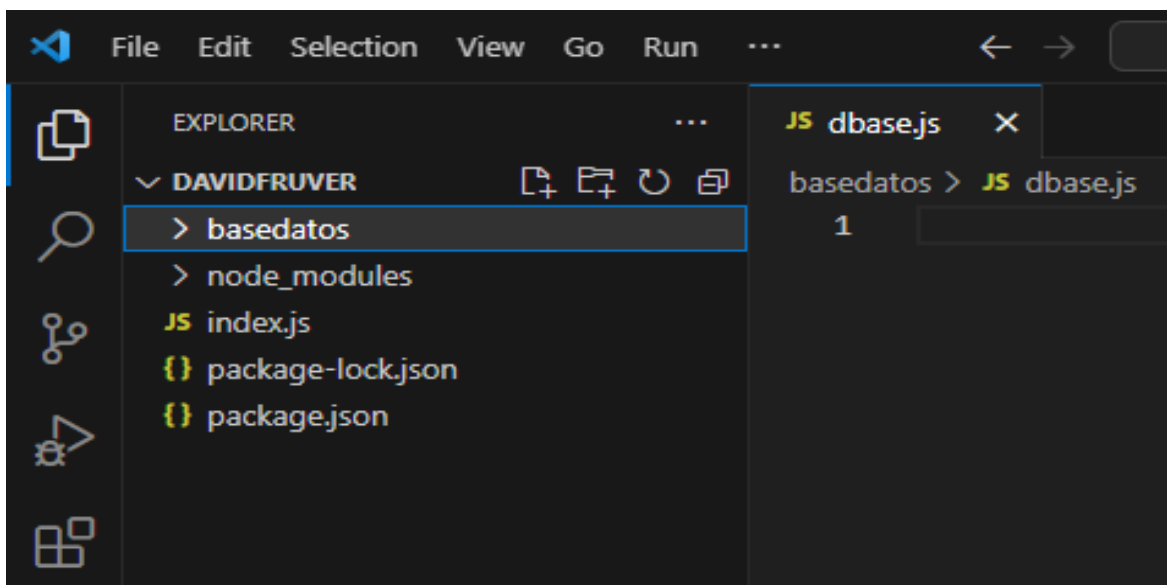
Esta instalación nos crea los archivos node_modules y package-lock.json el archivo index.js



Creamos el archivo index.js



Una vez creado el archivo index.js procedemos a realizar nuestro proyecto creamos la carpeta para la base de datos y dentro de la carpeta basedatos creamos el archivo dbase.js donde tendrá toda la configuración para la base de datos.



El archivo dbase.js nos quedara así

```

basedatos > JS dbase.js > [x] sequelize
1  import Sequelize from "sequelize";
2  const sequelize = new Sequelize ("davidfruver", "root", "", {
3    host: "localhost",
4    dialect: "mysql"
5  });
6  export { sequelize }

```

Luego creamos la carpeta controlador que contine el archivo controlador.js y dentro del archivo tendremos los métodos los cuales serán gestinados por la base de datos

En la siguiente imagen tenemos los métodos:

- consultarproducto
- consultarproductos

```

// este metodo permite consultar un producto en especifico
✓ const consultarproducto = async (req, res) => {
  const { idproducto } = req.params;
  ✓ try {
    const prod = await producto.findByPk(idproducto);
    res.status(200).json([prod]);
  } catch (error){
  ✓   res.status(400).json({mensaje: error});
  }
  };

//este metodo permite consultar todos los productos que tenemos disponibles
✓ const consultarproductos = async (req, res) => {
  ✓ try {
    const prod = await producto.findAll();
    res.status(200).json(prod);
  } catch (error){
  ✓   res.status(400).json({mensaje: error});
  }
  };

```

- almacenarProducto

```
// este metodo permite almacenar los productos
const almacenarproducto = async (req, res) => {
  const { idproducto, nombreproducto, detalleproducto, precioproducto } = req.body;
  try{
    const newproducto = await producto.create({
      idproducto,
      nombreproducto,
      detalleproducto,
      precioproducto
    });
    res.status(200).json(newproducto);
  }catch (error){
    res.status(400).json({mensaje: error});
  }
};
```

- actualizarproducto

```
// este metodo permite actualizar la lista los productos
const actualizarproducto = async (req, res) => {
  const { idproducto } = req.params;
  const { nombreproducto, detalleproducto, precioproducto } = req.body;
  try{
    const oldproducto = await producto.findByPk(idproducto);
    oldproducto.nombreproducto = nombreproducto;
    oldproducto.detalleproducto = detalleproducto;
    oldproducto.precioproducto = precioproducto;

    const modproducto = await oldproducto.save();
    res.status(200).json(modproducto);
  }catch (error){
    res.status(400).json({mensaje: error});
  }
};
```

- eliminar producto

```
// este metodo permite eliminar un registro de producto
const eliminarproducto = async (req, res) => {
  const { idproducto } = req.params;

  try{
    const respuesta = await producto.destroy({
      where: {
        idproducto: idproducto
      }
    });
    res.status(200).json({mensaje: "producto eliminado"});
  }catch (error){
    res.status(400).json({mensaje: "producto no eliminado" + error});
  }
};
```

Creamos la carpeta módulos donde tendrá el archivo módulo_producto.js para crear los productos

```
basedatos > módulos > JS módulo_producto.js > [?] producto > [?] precioproducto
1  import { DataTypes } from "sequelize";
2  import { sequelize } from "../dbase.js";
3
4  const producto = sequelize.define('productos', {
5    idproducto: {
6      type: DataTypes.INTEGER,
7      allowNull: true,
8      primaryKey: true,
9      autoIncrement: true
10   },
11   nombreproducto: {
12     type: DataTypes.STRING
13   },
14   detalleproducto: {
15     type: DataTypes.STRING
16   },
17   precioproducto: {
18     type: DataTypes.INTEGER
19   }
20 },
21 {
22   timestamps: false
23 });
24 export { producto }
```

Creamos la carpeta rutas y dentro de ella el archivo rutas.js donde tendrá los métodos que necesitamos para hacer las consultas ,almacenar, actualizar, y eliminar los productos desde el la web.

```
basedatos > rutas > JS rutas.js > ...
1  import {Router} from "express";
2
3  import { consultarproducto,
4    consultarproductos,
5    almacenarproducto,
6    actualizarproducto,
7    eliminarproducto
8  } from "../controlador/controlador.js";
9
10 const router = Router();
11
12 router.get("/productos/:idproducto", consultarproducto);
13 router.get("/productos", consultarproductos);
14 router.post("/productos", almacenarproducto);
15 router.put("/productos/:idproducto", actualizarproducto);
16 router.delete("/productos/:idproducto", eliminarproducto);
17
18 export default router;
```

Finalmente en el index.js importamos las librerías de cada una de las carpetas que creamos para nuestro proyecto quedando de esta manera

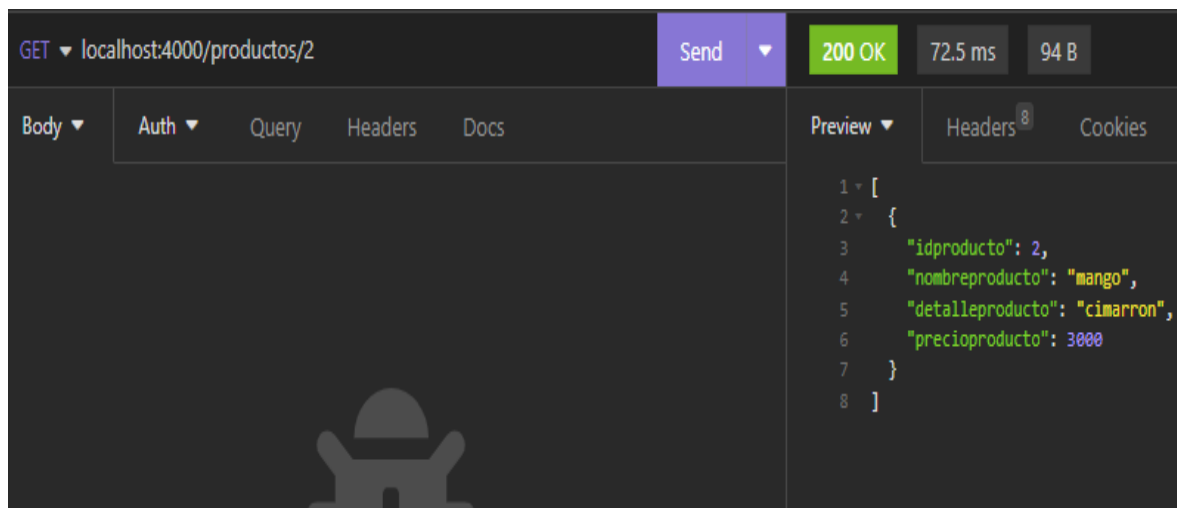
```
import express from "express";
import router from "../basedatos/rutas/rutas.js";
import { sequelize } from "../basedatos/dbase.js";
import { producto } from "../basedatos/modulos/modulo_producto.js";
import cors from 'cors';

// creamos la instancia
const app = express();
app.use(cors());
app.use(express.json());
app.use(router);

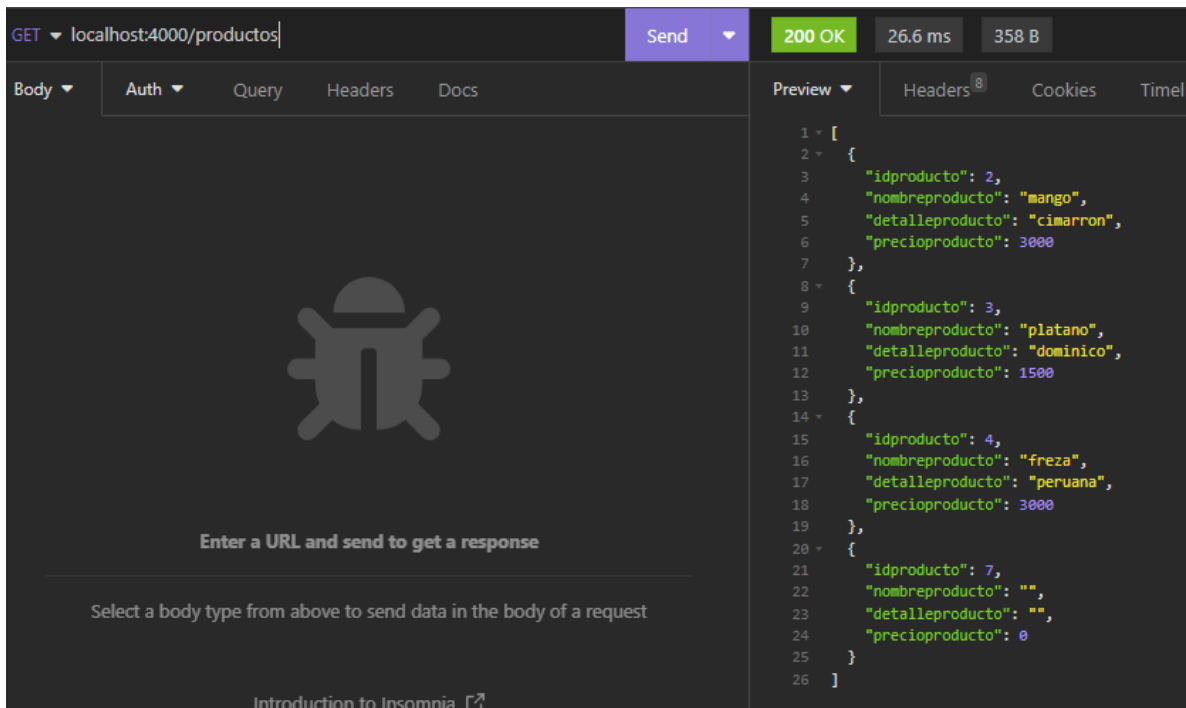
app.set("puerto", 4000);
app.listen(app.get("puerto"), () => {
  console.log(`puerto escuchando ${app.get("puerto")} `);
});
app.use(express.json());
app.use(router);
```

3. vamos a verificar las operaciones a través de un cliente gráfico Insomnia, y mostraremos el resultado de las solicitudes realizadas.

Como podemos ver hacemos la consulta con el método get vamos a visualizar un producto en específico enviándole el idproducto el cual es 2 y nos muestra dicho producto.

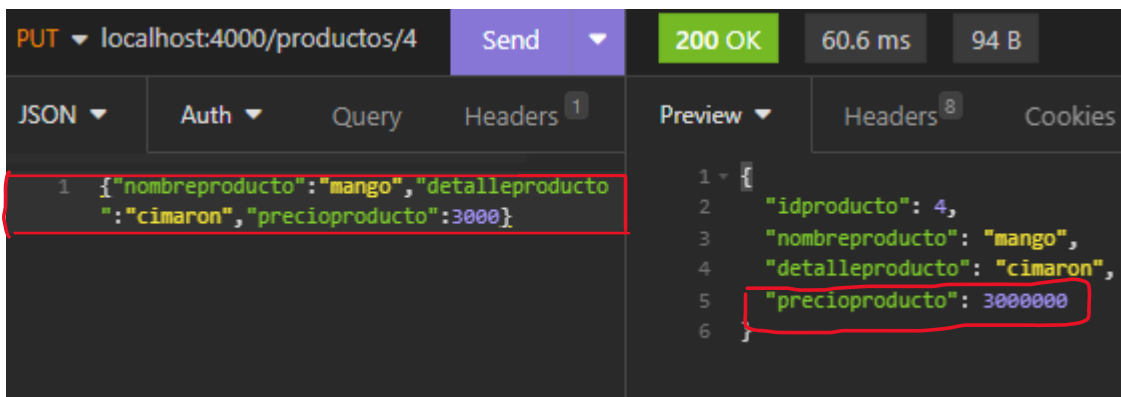


También tenemos otro método get que nos lista todos los productos que tenemos en la base de datos dándonos una lista de todos los productos disponibles



Otro método que tenemos es Put que nos actualiza o modifica un producto ya sea que nos hemos equivocado a la hora de introducirlo a nuestra base de datos

Como podemos observar en la siguiente imagen nos damos cuenta que en el registro de la parte derecha el precio del producto esta muy costoso "3,000,000", lo vamos a modificar haciendo un Put y le enviamos el id que en este caso seria 4 el cambio se hace como lo muestra la figura de la izquierda dándole un precio de 3000



Quedando así ya modificado

A screenshot of a REST client interface. At the top, it shows the status '200 OK' in a green box, the response time '60.9 ms', and the response size '91 B'. On the right, it says '1 Minute Ago' with a dropdown arrow. Below this is a tab bar with 'Preview' selected, and 'Headers' (with a badge '8'), 'Cookies', and 'Timeline'. The main area displays a JSON object with 6 lines of code:

```
1 {  
2   "idproducto": 4,  
3   "nombreproducto": "mango",  
4   "detalleproducto": "cimaron",  
5   "precioproducto": 3000  
6 }
```

Seguimos con el metodo post este método lo que hace es insertar un registro a nuestra base de datos, desde json le enviamos los parámetros y los podemos visualizar en la lista de registros

A screenshot of a REST client interface. The top bar shows the method 'POST' with a dropdown, the URL 'localhost:4000/productos', a 'Send' button with a dropdown, and the response status '200 OK' in a green box, response time '25.3 ms', and response size '89 B'. Below this is a tab bar with 'JSON' selected, and 'Auth' (with a dropdown), 'Query', 'Headers' (with a badge '1'), and 'Docs'. The main area displays a JSON object with 1 line of code:

```
1 {"nombreproducto":"pera","detalleproducto":"verde",  
  "precioproducto":4000}
```

On the right side, there is a 'Preview' tab selected, with 'Headers' (with a badge '8') and 'Cookies' tabs. The main area displays a JSON object with 6 lines of code:

```
1 {  
2   "idproducto": 10,  
3   "nombreproducto": "pera",  
4   "detalleproducto": "verde",  
5   "precioproducto": 4000  
6 }
```

```
Preview ▾ Headers 8 Cookies Timeline
15     "idproducto": 3,
16     "nombreproducto": "platano",
17     "detalleproducto": "dominico",
18     "precioproducto": 1500
19   },
20   {
21     "idproducto": 4,
22     "nombreproducto": "mango",
23     "detalleproducto": "cimaron",
24     "precioproducto": 3000
25   },
26   {
27     "idproducto": 8,
28     "nombreproducto": "",
29     "detalleproducto": "",
30     "precioproducto": 0
31   },
32   {
33     "idproducto": 7,
34     "nombreproducto": "",
35     "detalleproducto": "",
36     "precioproducto": 0
37   },
38   {
39     "idproducto": 10,
40     "nombreproducto": "pera",
41     "detalleproducto": "verde",
42     "precioproducto": 4000
43   }
44 ]
```

Y por ultimo tenemos el método delete que básicamente lo que hace es borrar, eliminar y destruir un registro enviándole el idproducto a eliminar en este caso vamos a eliminar el mismo registro que anteriormente agregamos el registro pera le damos el idproducto como lo mencionamos anteriormente y nos vota un mensaje que el producto fue eliminado

```
DELETE ▾ localhost:4000/productos/10 Send ▾ 200 OK 38.4 ms 32 B
Body ▾ Auth ▾ Query Headers Docs Preview ▾ Headers 8 Cookies
1 {
2   "mensaje": "producto eliminado"
3 }
```

Lo comprobamos listando otra vez todos los registros y efectivamente ya esta eliminado el registro pera

GET

localhost:4000/productos

Send

200 OK

21.3 ms

265 B

Body

Auth

Query

Headers


Docs

Preview

Headers8

Cookies

Timeli



Enter a URL and send to get a response

```
1 [
2   {
3     "idproducto": 3,
4     "nombreproducto": "platano",
5     "detalleproducto": "dominico",
6     "precioproducto": 1500
7   },
8   {
9     "idproducto": 4,
10    "nombreproducto": "mango",
11    "detalleproducto": "cimaron",
12    "precioproducto": 3000
13  },
14  {
15    "idproducto": 8,
16    "nombreproducto": "",
17    "detalleproducto": "",
18    "precioproducto": 0
19  }
20 ]
```