

**Diplomado de actualización en nuevas tecnologías para el desarrollo de  
software**

**Taller Unidad 2 Backend**

**LEIDY TATIANA URBANO YELA**

**Correo: [leidyurbano@udenar.edu.co](mailto:leidyurbano@udenar.edu.co)**

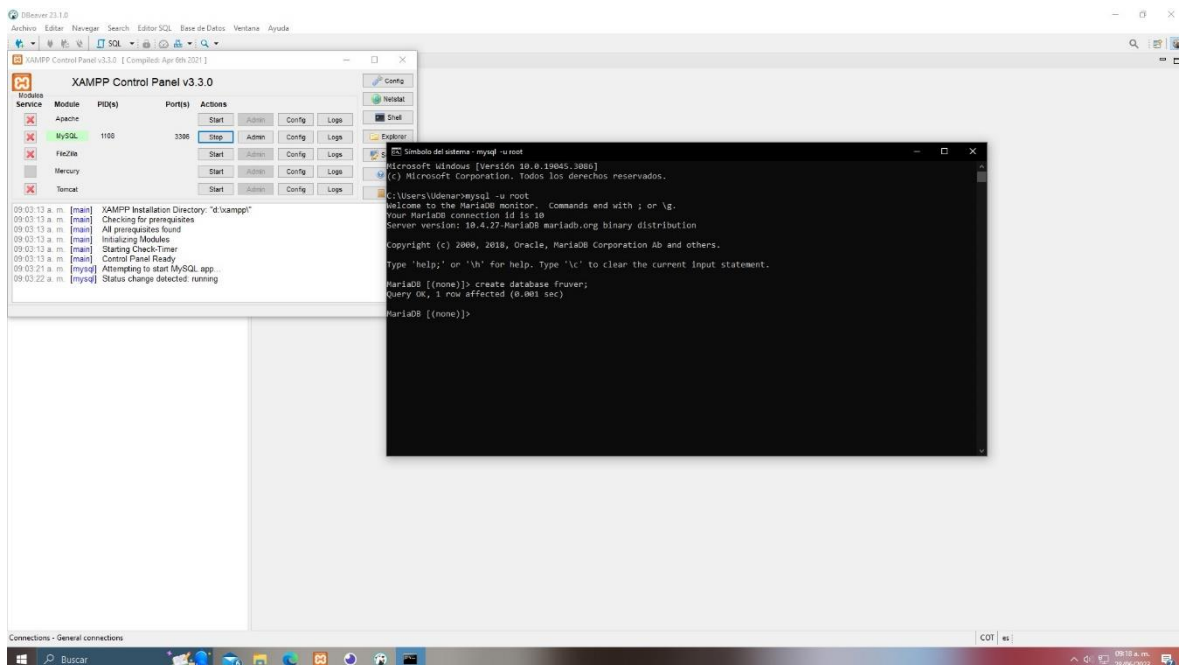
**UNIVERSIDAD DE NARIÑO**

**INGENIERIA DE SISTEMAS**

**2023**

## Desarrollo Taller 2 Backend

1. Para el desarrollo de esta actividad necesitaremos XAMPP y Dbeaver para la administración de la base de datos.
  - Se inicia el servicio de mysql en xampp, creando por medio de la consola la base de datos Fruver con el comando “create database fruver;”



- En Dbeaver se crea una conexión con la base de datos Fruver

Conectar a base de datos

**Connection Settings**  
MariaDB ajustes de conexión

General Driver properties SSH Proxy SSL

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:mariadb://localhost:3306/fruver

Server Host: localhost Port: 3306

Database: fruver

Authentication (Database Native)

Nombre de usuario: root

Contraseña:  ☒ Save password locally

Advanced

Server Time Zone: Auto-detect

Local Client: D:\xampp\mysql

[You can use variables in connection parameters.](#) Connection details (name, type, ...)

Driver name: MariaDB Driver Settings

Probar conexión ... < Back Next > Finish Cancel

Conectar a base de datos

**Seleccione su base de datos**  
Crear nueva conexión a base de datos. Encuentre su driver de la base de datos en la lista inferior.

Escribe parte del nombre de la base de datos/driver para filtrar Sort by: ☐ Title ☒ Score

All Popular SQL NoSQL Analytical Timeseries Embedded Hadoop / BigData Full-text search Graph databases

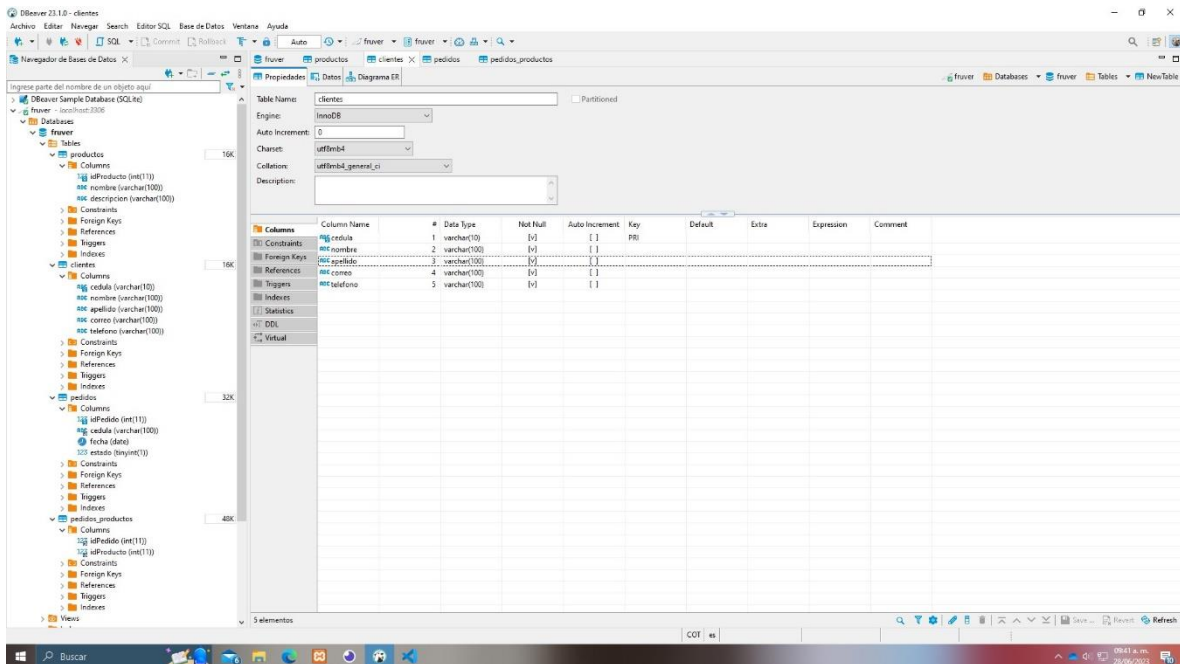
PostgreSQL SQLite IBM DB2 Do2 for LUW DuckDB MariaDB

MySQL Oracle SQL Server TiDB Apache Calcite Avatica

Apache Drill Apache Hive

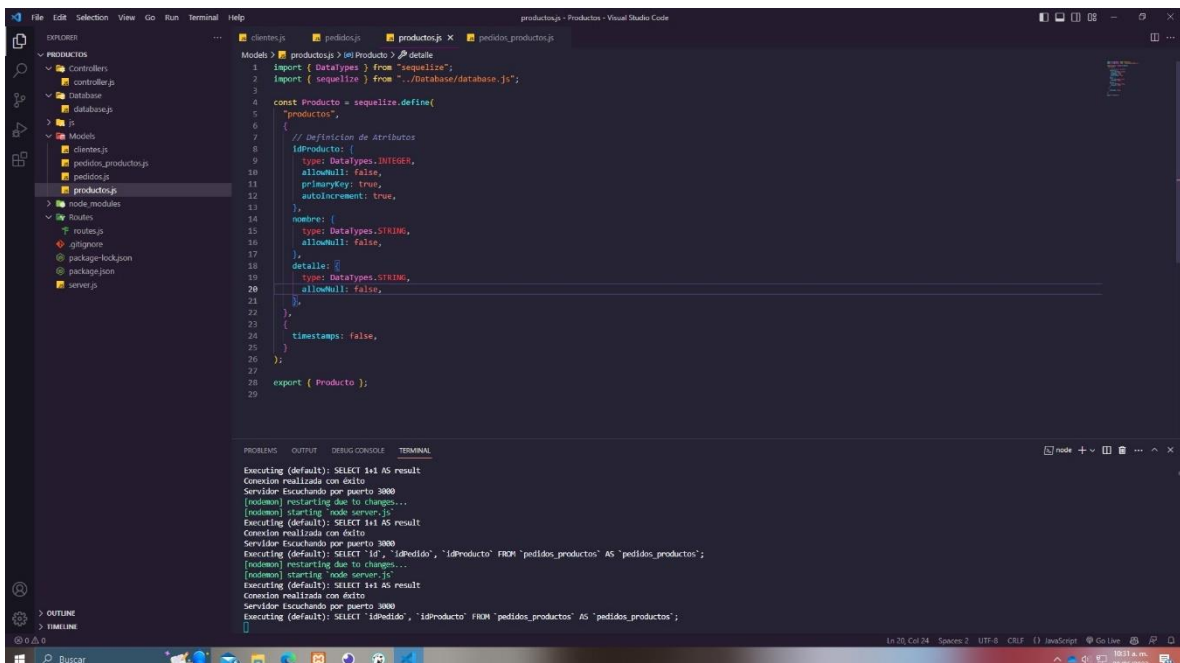
Probar conexión ... < Back Next > Finish Cancel

- Creamos las respectivas tablas en la base de datos



## 2. Desarrollar una aplicación backend implementada en node js, usando la estructura vista en clase.

- Se define cada tabla en un modelo usando sequelize, un archivo js para cada tabla.



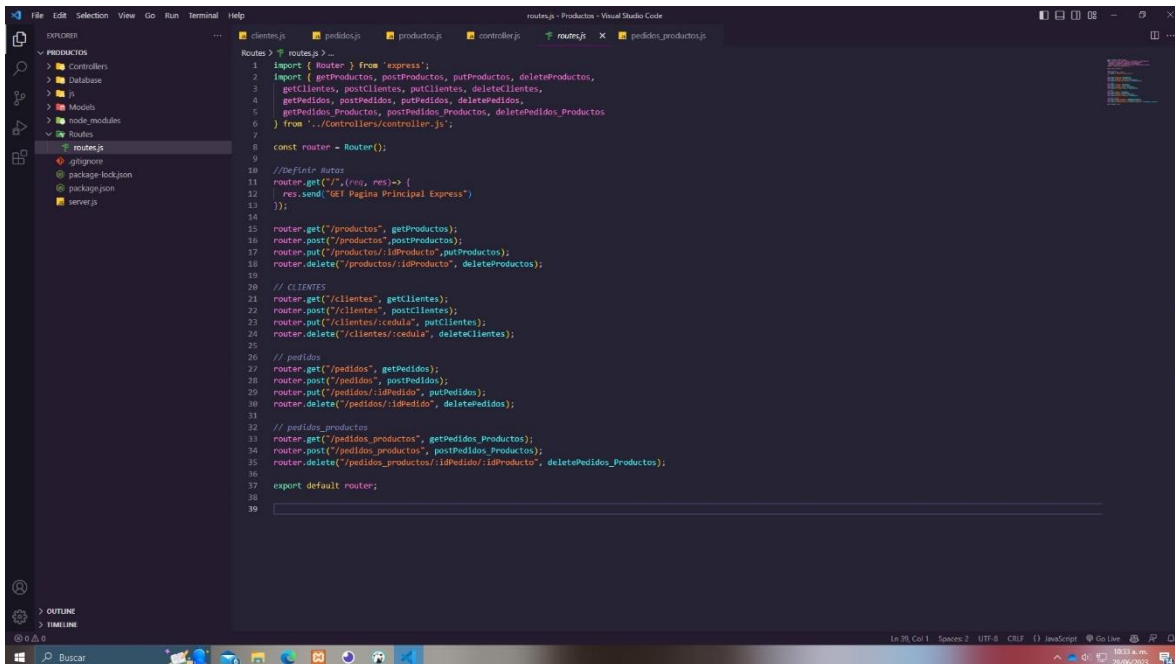
- En el archivo controller.js se definen las funciones que se van a utilizar para la gestión de los datos en la base de datos

```
1 import { Product } from "../Models/productos.js";
2 import { Pedido } from "../Models/pedidos.js";
3 import { Pedido_Producto } from "../Models/pedidos_productos.js";
4 import { Cliente } from "../Models/clientes.js";
5
6
7 const getProductos = async (req, res) => {
8   //res.send("GET Pagina Productos desde Controller");
9   try {
10     const productos = await Product.findAll();
11     res.status(200).json(productos);
12   } catch (error) {
13     res.status(400).json({ mensaje: error });
14   }
15 };
16
17 const postProductos = async (req, res) => {
18   //res.send("POST Pagina Productos desde Controller");
19   const { nombre, detalle } = req.body;
20   try {
21     const newProducto = await Product.create({
22       nombre,
23       detalle,
24     });
25     res.status(200).json(newProducto);
26   } catch (error) {
27     res.status(400).json({ mensaje: error });
28   }
29 };
30
31 const putProducto = async (req, res) => {
32   const { idProducto } = req.params;
33   const { nombre, detalle } = req.body;
34   try {
35     const oldProducto = await Product.findById(idProducto);
36     oldProducto.nombre = nombre;
37     oldProducto.detalle = detalle;
38     const modProducto = await oldProducto.save();
39     res.status(200).json(modProducto);
40   } catch (error) {
41     res.status(400).json({ mensaje: error });
42   }
43 };
44
45 const deleteProductos = async (req, res) => {
46   //res.send("DELETE Pagina Productos desde Controller");
47   const { idProducto } = req.params;
48   try {
49     const respuesta = await Product.destroy({
50       idProducto
51     });
52     res.status(200).json(respuesta);
53   } catch (error) {
54     res.status(400).json({ mensaje: error });
55   }
56 };
57
58 export { getProductos, postProductos, putProducto, deleteProductos };
```

- En el archivo database.js se configura la conexión a la base de datos

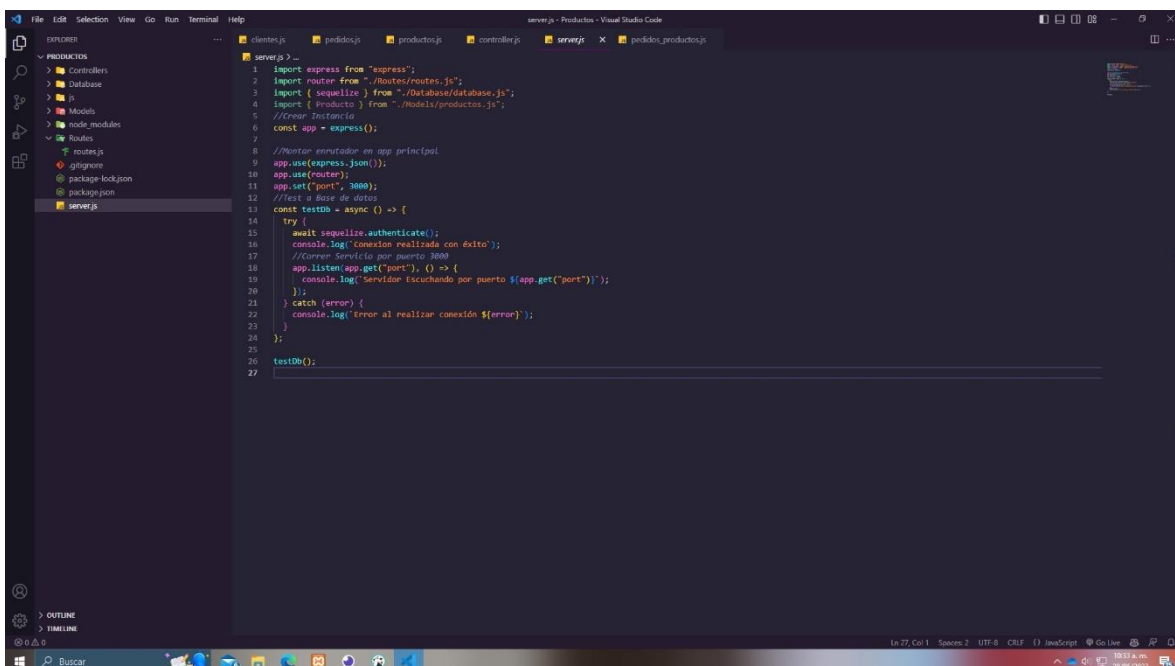
```
1 import Sequelize from "sequelize";
2
3 const sequelize = new Sequelize("fruver", "root", "", {
4   host: "localhost",
5   dialect: "mysql",
6 });
7
8
9 export {
10   sequelize
11 };
```

- En el archivo routes.js se define las rutas y con la respectiva función que se importa del controlador.



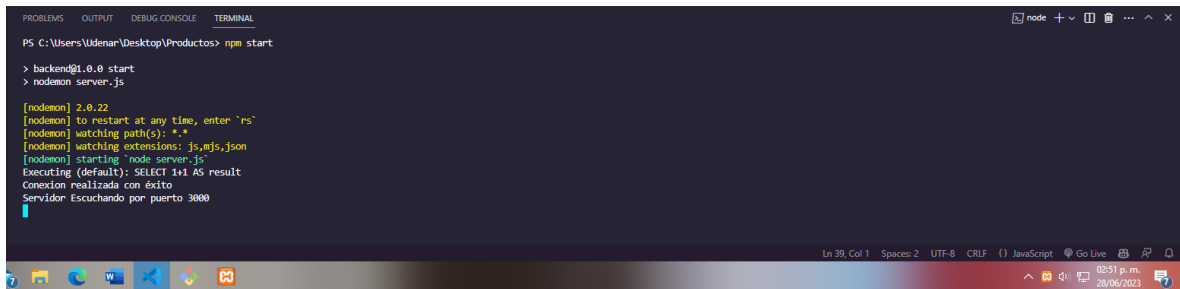
```
1 import { Router } from 'express';
2 import { getProductos, postProductos, putProductos, deleteProductos,
3   getClientes, postClientes, putClientes, deleteClientes,
4   getPedidos, postPedidos, putPedidos, deletePedidos,
5   getPedidos_Productos, postPedidos_Productos, deletePedidos_Productos
6 } from '../Controllers/controller.js';
7
8 const router = Router();
9
10 //Definir rutas
11 router.get('/', (req, res) => {
12   res.send("Get Pagina Principal Express")
13 });
14
15 router.get("/productos", getProductos);
16 router.post("/productos", postProductos);
17 router.put("/productos/:idProducto", putProductos);
18 router.delete("/productos/:idProducto", deleteProductos);
19
20 // CLIENTES
21 router.get("/clientes", getClientes);
22 router.post("/clientes", postClientes);
23 router.put("/clientes/:cedula", putClientes);
24 router.delete("/clientes/:cedula", deleteClientes);
25
26 // pedidos
27 router.get("/pedidos", getPedidos);
28 router.post("/pedidos", postPedidos);
29 router.put("/pedidos/:idPedido", putPedidos);
30 router.delete("/pedidos/:idPedido", deletePedidos);
31
32 // pedidos_productos
33 router.get("/pedidos_productos", getPedidos_Productos);
34 router.post("/pedidos_productos", postPedidos_Productos);
35 router.delete("/pedidos_productos/:idPedido/:idProducto", deletePedidos_Productos);
36
37 export default router;
38
```

- En el archivo app.js se configura el servidor usando el entorno de trabajo Express.



```
1 import express from 'express';
2 import router from './Routes/routes.js';
3 import { sequelize } from './Database/database.js';
4 import { producto } from './Models/productos.js';
5 //Crear Instancia
6 const app = express();
7
8 //Montar servidor en app principal
9 app.use(express.json());
10 app.use(router);
11 app.set('port', 3000);
12 //Test a Base de datos
13 const testDb = async () => {
14   try {
15     await sequelize.authenticate();
16     console.log('Conexion realizada con éxito');
17     //Correr Servicio por puerto 3000
18     app.listen(app.get('port'), () => {
19       console.log('Servidor escuchando por puerto ${app.get('port')}');
20     });
21   } catch (error) {
22     console.log('Error al realizar conexión ${error}');
23   }
24 };
25
26 testDb();
27
```

- Se lanza el servicio backend con npm start



```
PS C:\Users\Wdenar\Desktop\Productos> npm start

> backend@1.0.0 start
> nodemon server.js

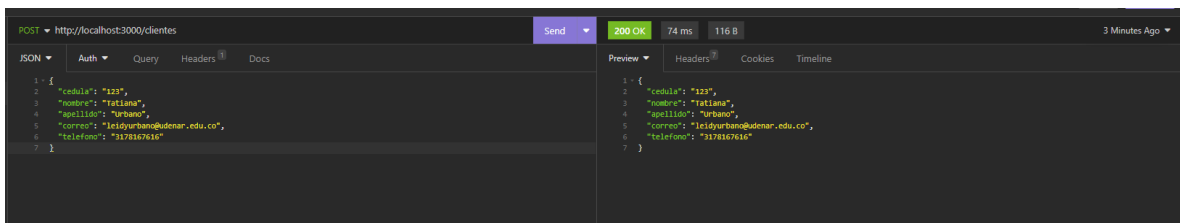
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Executing (default): SELECT 1+1 AS result
Conexion realizada con éxito
Servidor Escuchando por puerto 3000
```

### 3. Pruebas

Usando la plataforma API Insomnia REST se realizarán las solicitudes http para probar las rutas que ya definimos previamente para la gestión de los datos

## Cientes

### 1. crear nuevo cliente

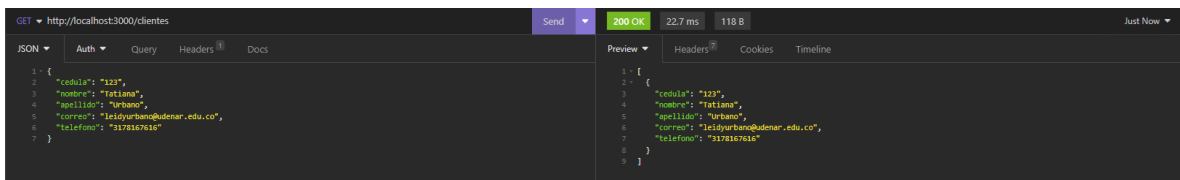


```
POST http://localhost:3000/clientes 200 OK 74 ms 116 B 3 Minutes Ago

JSON Auth Query Headers Docs

1 {
2   "cedula": "123",
3   "nombre": "Tatiana",
4   "apellido": "Arbano",
5   "correo": "leidyarbano@udenar.edu.co",
6   "telefono": "3178167616"
7 }
```

### 2. Obtener clientes

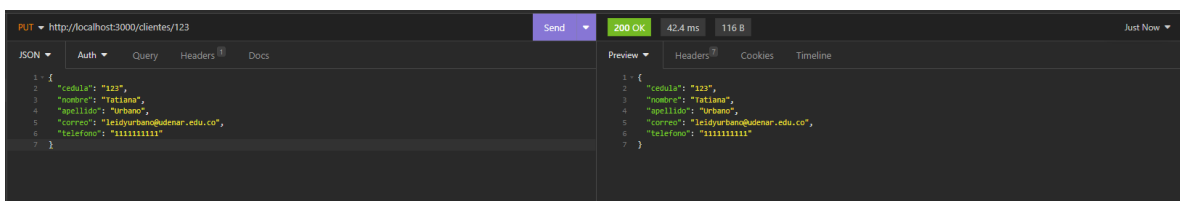


```
GET http://localhost:3000/clientes 200 OK 22.7 ms 118 B Just Now

JSON Auth Query Headers Docs

1 {
2   "cedula": "123",
3   "nombre": "Tatiana",
4   "apellido": "Arbano",
5   "correo": "leidyarbano@udenar.edu.co",
6   "telefono": "3178167616"
7 }
```

### 3. Actualizar cliente

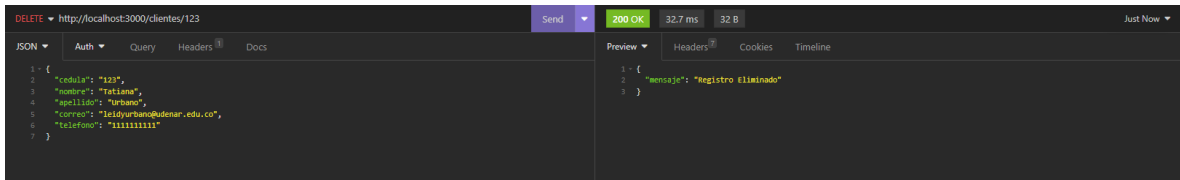


```
PUT http://localhost:3000/clientes/123 200 OK 42.4 ms 116 B Just Now

JSON Auth Query Headers Docs

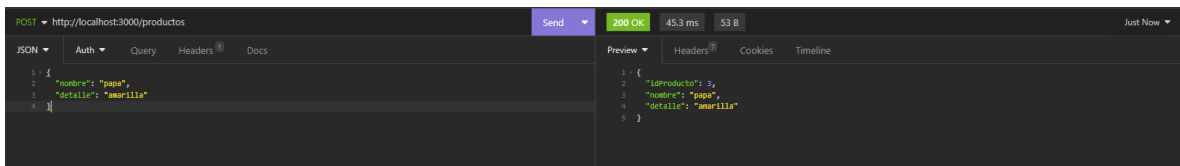
1 {
2   "cedula": "123",
3   "nombre": "Tatiana",
4   "apellido": "Arbano",
5   "correo": "leidyarbano@udenar.edu.co",
6   "telefono": "1111111111"
7 }
```

## 4. Borrar cliente

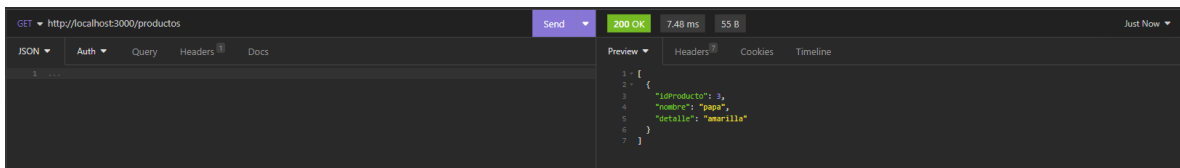


## Productos

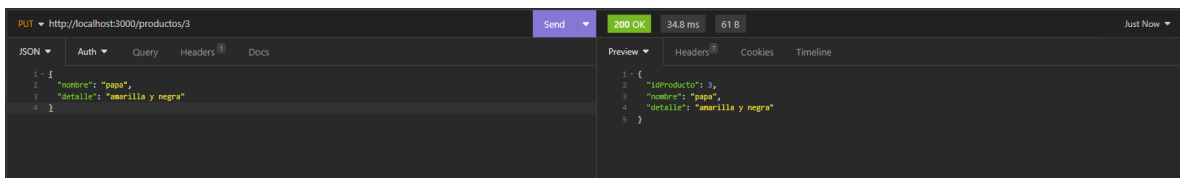
### 1. Crear nuevo producto



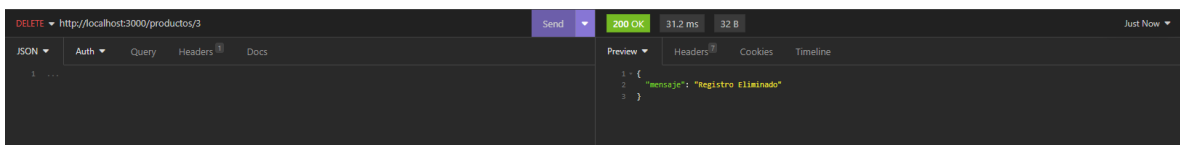
### 2. Obtener productos



### 3. Actualizar producto



### 4. Borrar producto





## Pedidos

Para las pruebas de gestión de la tabla pedidos se necesita tener una cedula ya registrada por la referencia que se hace hacia la tabla clientes, de lo contrario ocurrirá un error.

```
POST http://localhost:3000/pedidos
{
  "cedula": 456,
  "fecha": "28/06/2023"
}
```

```
{
  "mensaje": {
    "name": "SqliteForeignkeyConstraintError",
    "parent": {
      "code": "ER_NO_REFERENCED_ROW_2",
      "errno": 1452,
      "sqlstate": "23000",
      "sqlmessage": "Cannot add or update a child row: a foreign key constraint fails ('fruter'.pedidos, CONSTRAINT `pedidos_fk` FOREIGN KEY (`cedula`) REFERENCES `clientes` (`cedula`))",
      "sql": "INSERT INTO `pedidos` (`idpedido`, `cedula`, `fecha`) VALUES (DEFAULT,?,?)",
      "parameters": [
        456,
        "Invalid date"
      ]
    },
    "original": {
      "code": "ER_NO_REFERENCED_ROW_2",
      "errno": 1452,
      "sqlstate": "23000",
      "sqlmessage": "Cannot add or update a child row: a foreign key constraint fails ('fruter'.pedidos, CONSTRAINT `pedidos_fk` FOREIGN KEY (`cedula`) REFERENCES `clientes` (`cedula`))",
      "sql": "INSERT INTO `pedidos` (`idpedido`, `cedula`, `fecha`) VALUES (DEFAULT,?,?)",
      "parameters": [
        456,
        "Invalid date"
      ]
    },
    "sql": "INSERT INTO `pedidos` (`idpedido`, `cedula`, `fecha`) VALUES (DEFAULT,?,?)",
    "parameters": [
      456,
      "Invalid date"
    ],
    "table": "clientes",
    "fields": [
      "cedula"
    ],
    "value": 456,
    "index": "pedidos_fk",
    "reltype": "child"
  }
}
```

### 1. Crear pedido

```
POST http://localhost:3000/pedidos
{
  "cedula": 123,
  "fecha": "2023/06/28"
}
```

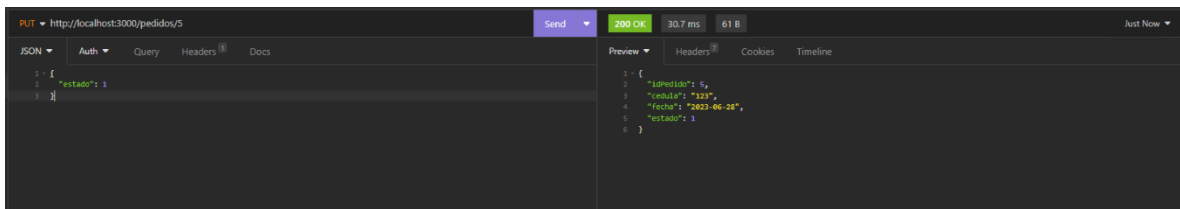
```
{
  "idpedido": 5,
  "cedula": 123,
  "fecha": "2023-06-28T05:00:00.000Z"
}
```

### 2. Obtener pedidos

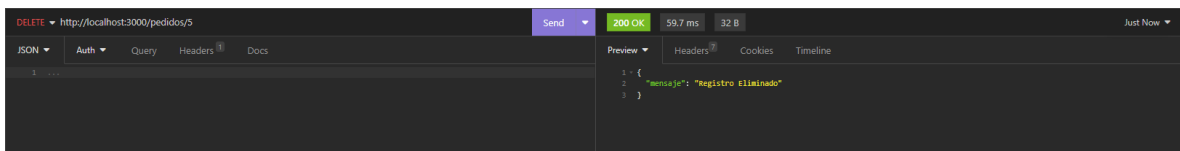
```
GET http://localhost:3000/pedidos
```

```
[
  {
    "idpedido": 5,
    "cedula": 123,
    "fecha": "2023-06-28",
    "created": 0
  }
]
```

### 3. Actualizar pedido

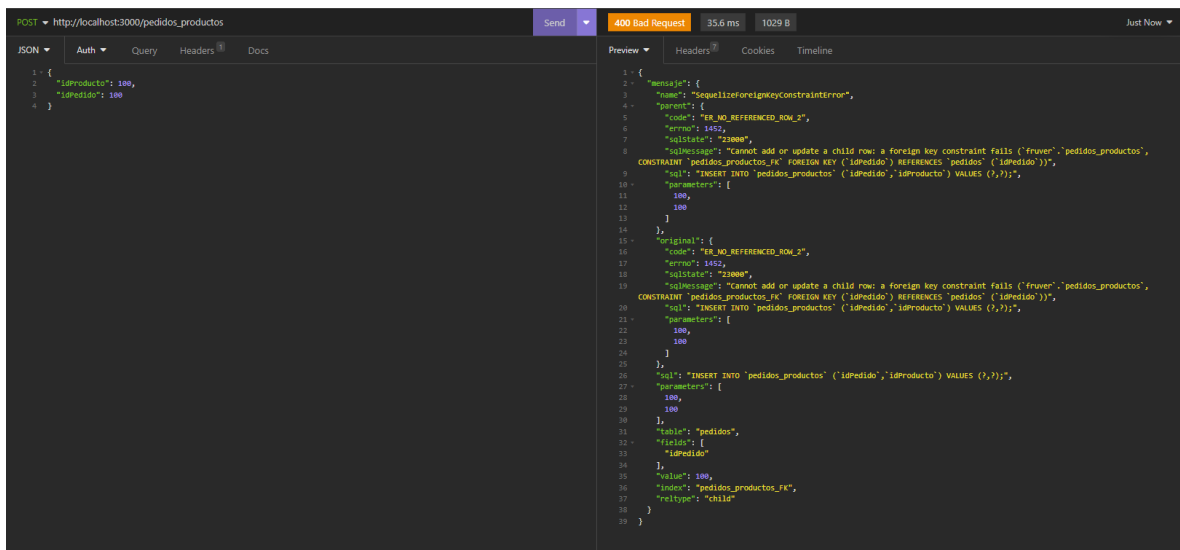


### 4. Borrar pedido

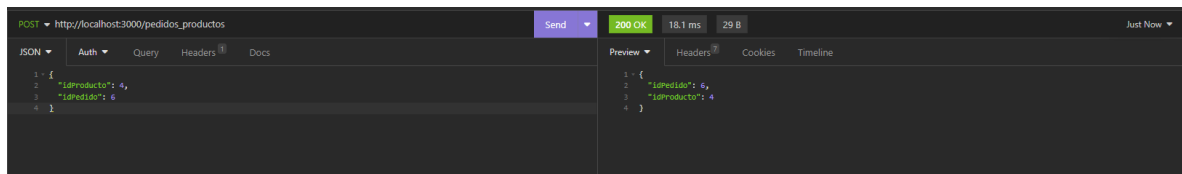


## Pedidos productos

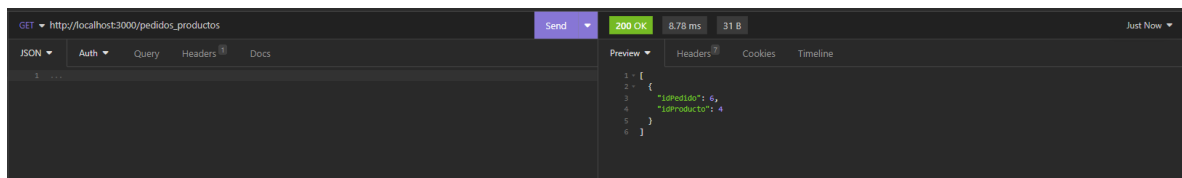
Para las pruebas de gestión de la tabla Pedidos producto se necesita tener el id de un pedido y el id de un producto ya registrados por la referencia que se hace hacia la tabla pedido y productos, de lo contrario ocurrirá un error.



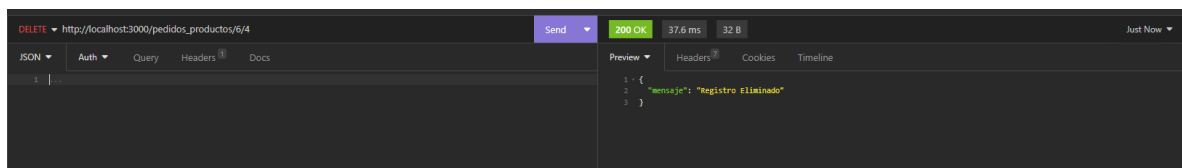
### 1. Crear pedido producto con el idPedido 6 y el idProducto 4



### 2. Obtener pedido producto

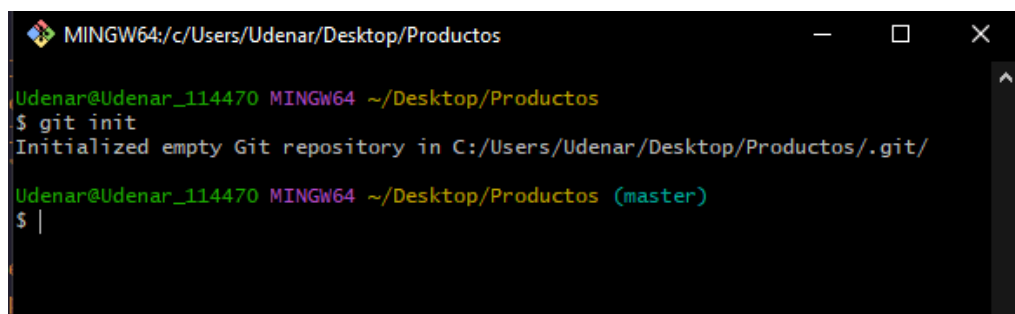


### 3. Borrar pedido producto



## GIT

### 1. Inicializamos un repositorio local



### 2. Creamos un repositorio remoto en github y lo sincronizamos con nuestro proyecto

```
MINGW64/c/Users/Udenar/Desktop/Productos
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$ git remote add origin https://github.com/tatianaurbano/Backend.git
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$ git remote -v
origin  https://github.com/tatianaurbano/Backend.git (fetch)
origin  https://github.com/tatianaurbano/Backend.git (push)
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/tatianaurbano/Backend.git'
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$ git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the
next time Git touches it
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$ git commit -m "primer commit"
[main (root-commit) 553d498] primer commit
11 files changed, 1830 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Controllers/controller.js
 create mode 100644 Database/database.js
 create mode 100644 Models/clientes.js
 create mode 100644 Models/pedidos.js
 create mode 100644 Models/pedidos_productos.js
 create mode 100644 Models/productos.js
 create mode 100644 Routes/routes.js
 create mode 100644 package-lock.json
 create mode 100644 package.json
 create mode 100644 server.js
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (17/17), 17.84 KiB | 4.46 MiB/s, done.
Total 17 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/tatianaurbano/Backend.git
 * [new branch]      main -> main
Udenar@Udenar_114470 MINGW64 ~/Desktop/Productos (main)
$
```

### 3. Enlace git

<https://github.com/tatianaurbano/Backend.git>