

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

Taller Unidad 2 Backend

EDISON CAMILO ROSERO ACHICANOY

UNIVERSIDAD DE NARIÑO

INGENIERIA DE SISTEMAS

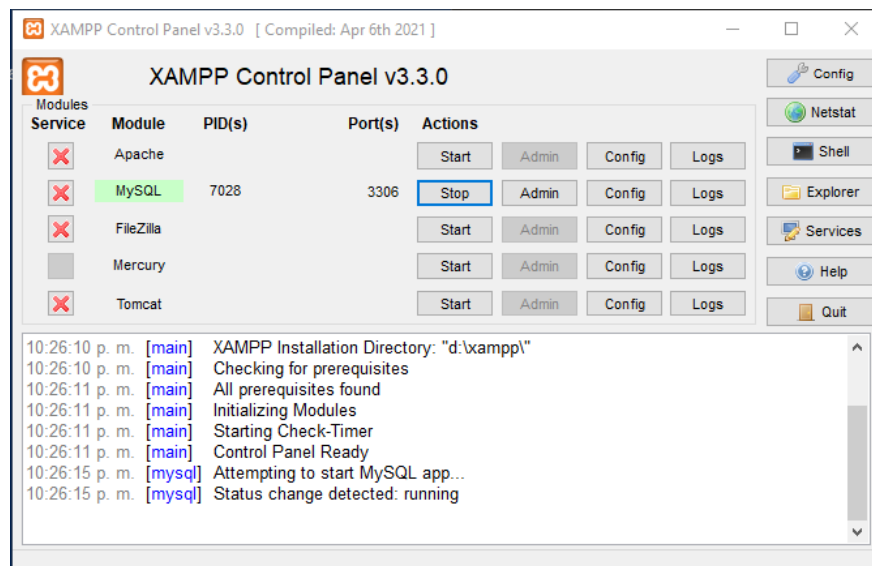
2023

Herramientas de software a utilizar.

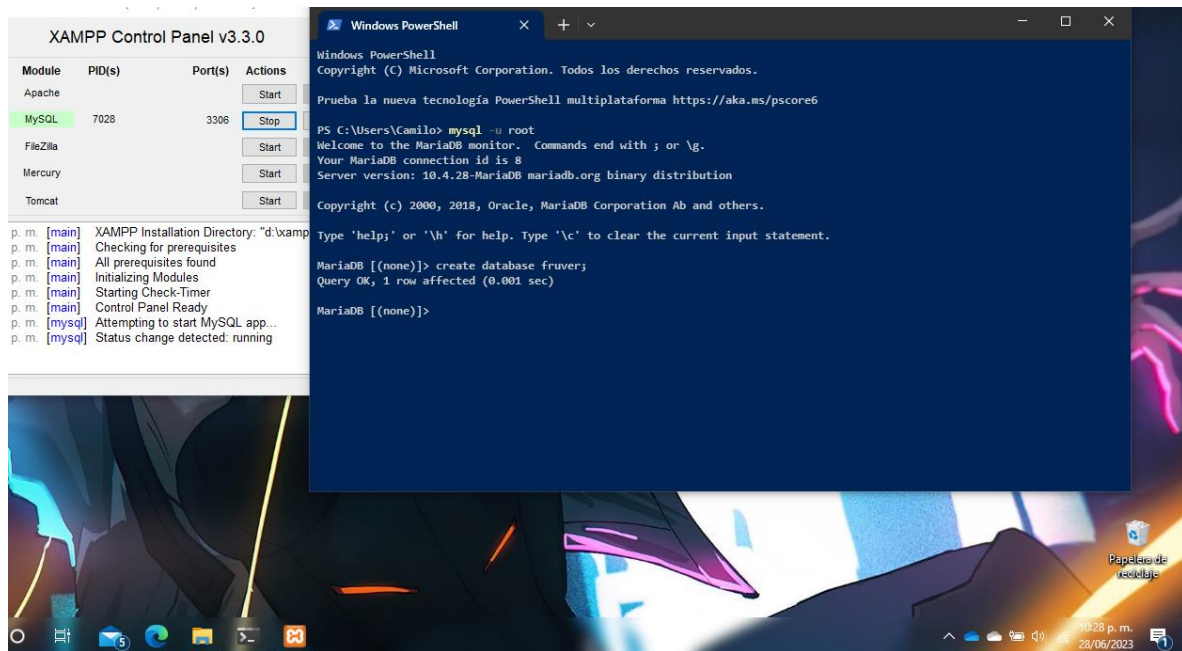
- XAMPP: Proporciona un entorno de desarrollo local, en este caso para administrar y configurar una base de datos con los servicios de apache y phpmyadmin.
- Postman: Permitirá probar a través de solicitudes http los diferentes endpoints del backend.
- Visual Studio Code: Editor de código.
- NodeJs: Entorno de ejecución de javascript del lado del servidor.
- Dbeaver: Para administrar bases de datos a través de una interfaz gráfica muy intuitiva.

1. Creación de la base de datos.

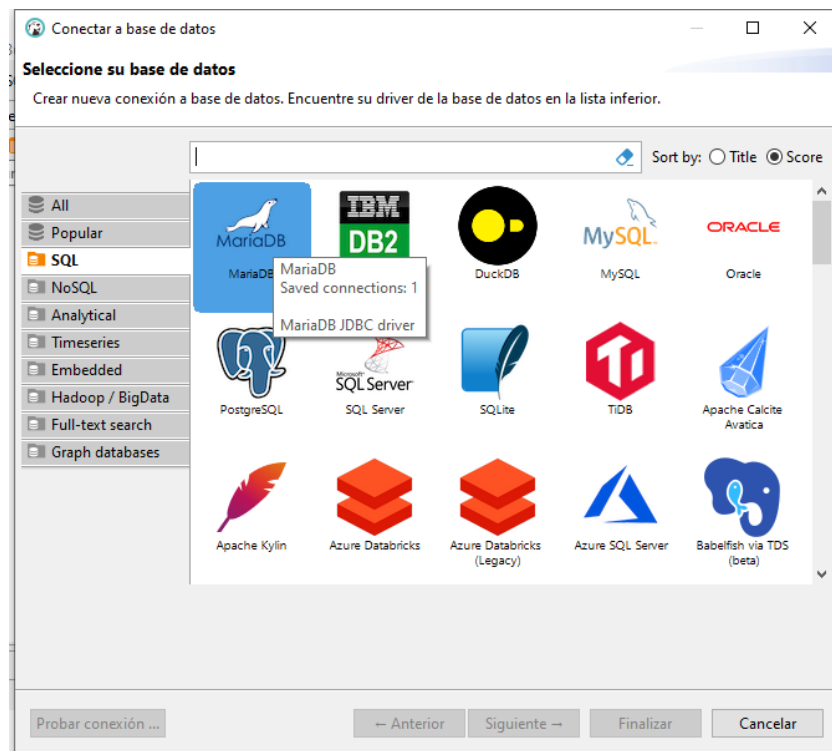
- Iniciar el servicio de mysql en xampp para creación y configuración de la base de datos



- Por medio de Powershell se realiza una conexión a mysql con el comando "mysql -u root".
- Después de iniciar sesión en mysql, se procede a crear la base de datos "Fruver" con el comando "create database fruver".

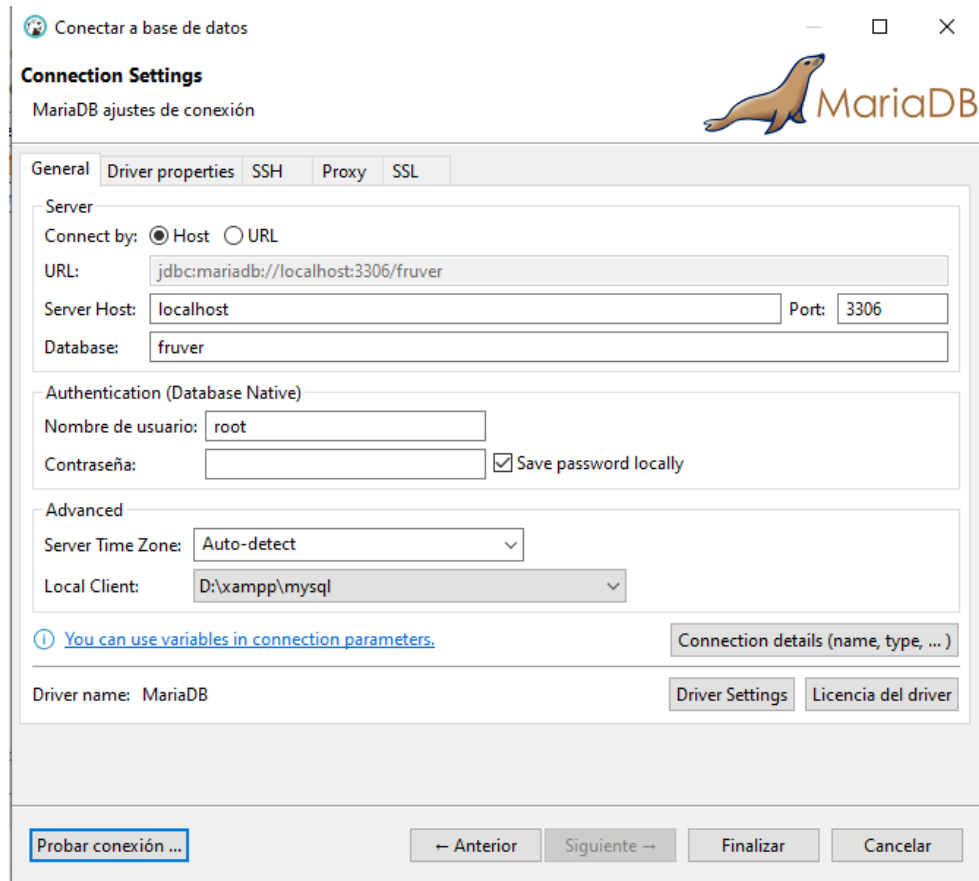


- Con el programa DBeaver se crea una conexión a la base de datos Fruver. Para este tipo de conexión se usará la base de datos MariaDb.



Se llena los campos a su consideración. En este caso será:

- Server Host: localhost (trabajar en un entorno local)
- Database: fruver (nombre de la base de datos)
- Nombre de usuario: root
- Contraseña: (ninguna)



The screenshot shows the 'Conectar a base de datos' (Connect to database) dialog box for MariaDB. The window title is 'Conectar a base de datos'. The main heading is 'Connection Settings' with the subtitle 'MariaDB ajustes de conexión'. The MariaDB logo is in the top right corner. The 'General' tab is selected, showing fields for 'Server', 'Authentication (Database Native)', and 'Advanced'. The 'Server' section has 'Connect by:' set to 'Host' (selected) and 'URL' set to 'jdbc:mariadb://localhost:3306/fruver'. The 'Server Host' is 'localhost' and the 'Port' is '3306'. The 'Database' is 'fruver'. The 'Authentication (Database Native)' section has 'Nombre de usuario:' set to 'root' and 'Contraseña:' is empty. The 'Save password locally' checkbox is checked. The 'Advanced' section has 'Server Time Zone' set to 'Auto-detect' and 'Local Client' set to 'D:\xampp\mysql'. At the bottom, there is a 'Probar conexión ...' button and navigation buttons: '← Anterior', 'Siguiente →', 'Finalizar', and 'Cancelar'.

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

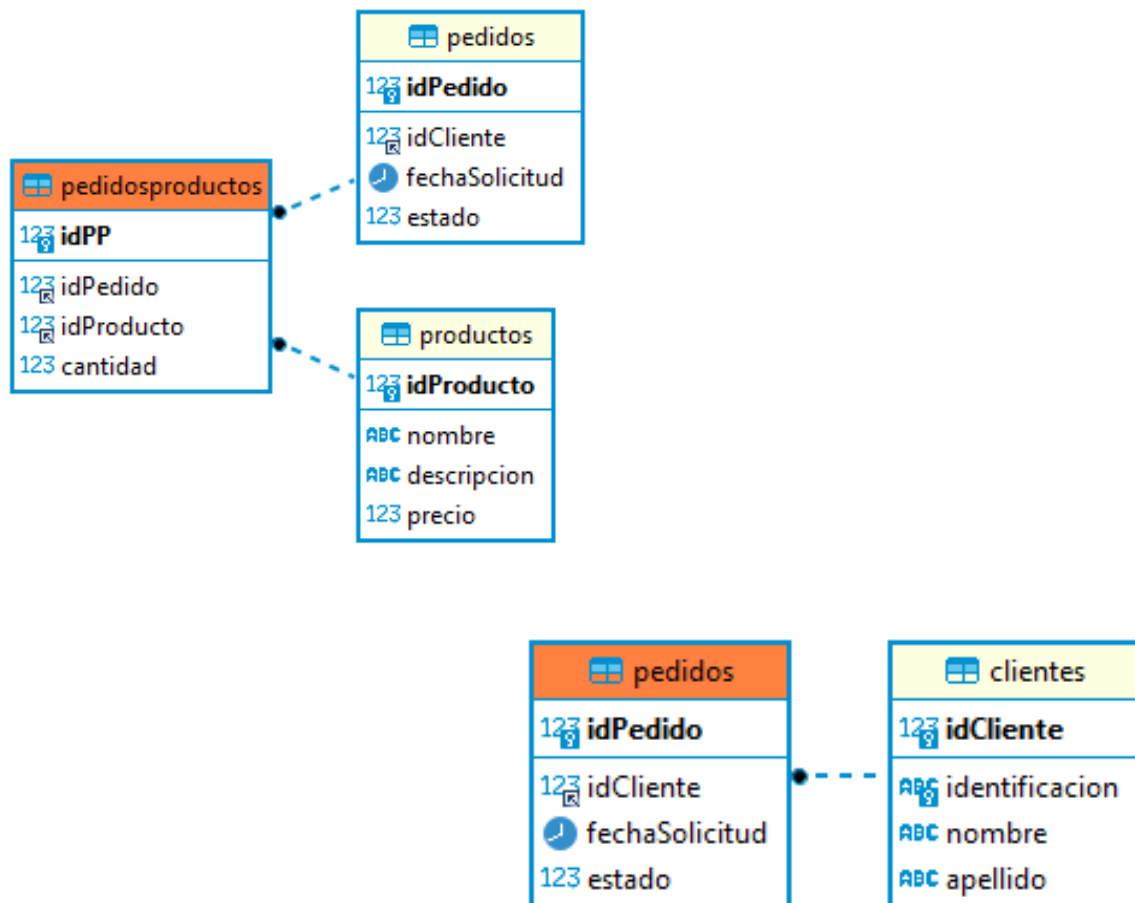
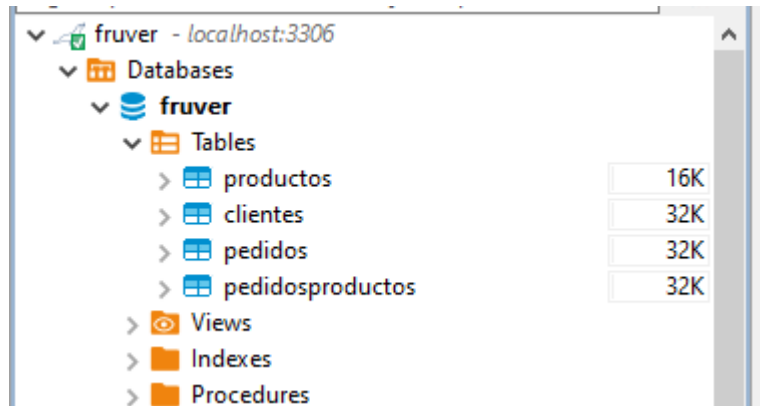
Local Client: D:\xampp\mysql

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

Driver name: MariaDB Driver Settings Licencia del driver

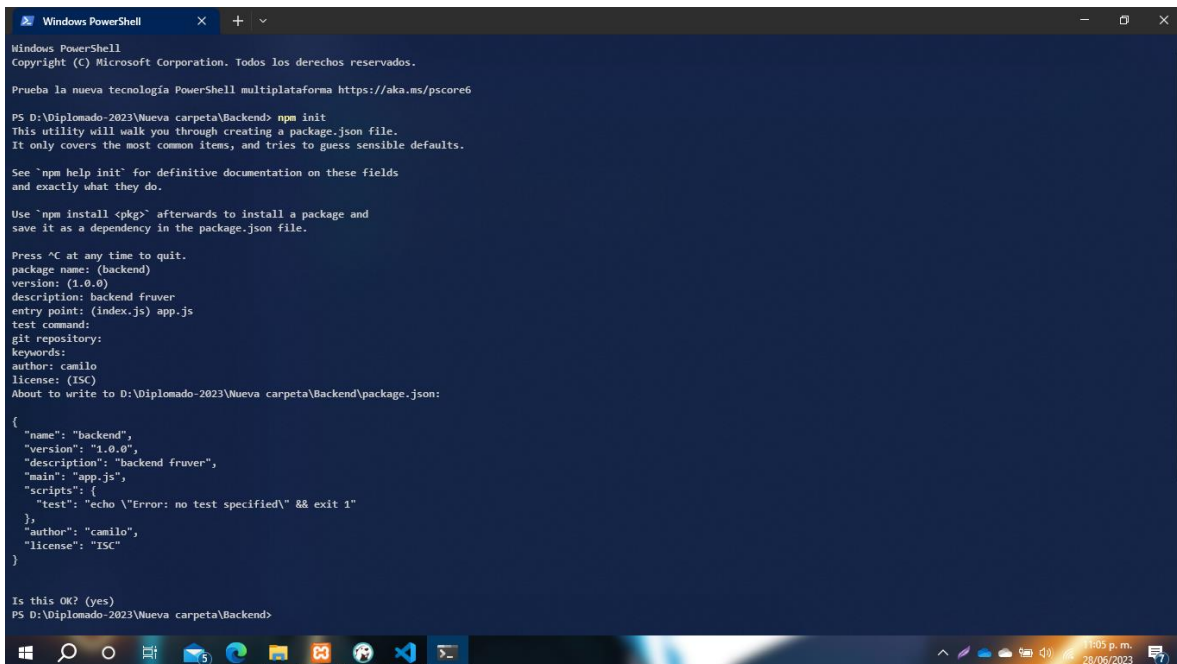
Probar conexión ... ← Anterior Siguiente → Finalizar Cancelar

- Después de crear la conexión se procede a crear las tablas, estableciendo sus llaves primarias y sus relaciones.



2. Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS. Es necesario ya tener instalado el programa Nodejs y un editor de código.

- Se crea una carpeta con el nombre que prefiera, dentro de la carpeta se abre una consola de comandos y se escribe el comando “npm init” para crear un proyecto de Node.js y generar un archivo package.json. En este caso el archivo desde el cual se inicia la aplicación (entry point) se llama app.js.



```
Windows PowerShell
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS D:\Diplomado-2023\Nueva carpeta\Backend> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

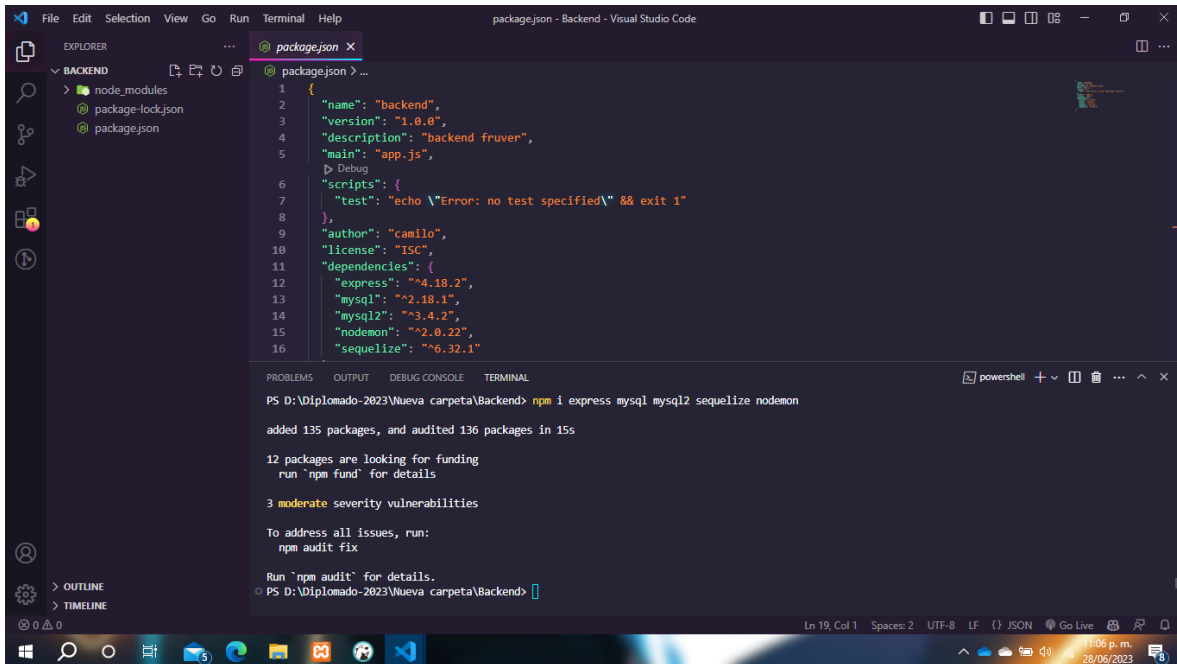
Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (backend)
version: (1.0.0)
description: backend fruver
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: camilo
license: (ISC)
About to write to D:\Diplomado-2023\Nueva carpeta\Backend\package.json:
{
  "name": "backend",
  "version": "1.0.0",
  "description": "backend fruver",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "camilo",
  "license": "ISC"
}

Is this OK? (yes)
PS D:\Diplomado-2023\Nueva carpeta\Backend>
```

- Abrir la carpeta con el editor de código, en este caso es visual studio code, abrir la terminal integrada y escribir el comando “npm i express mysql mysql2 sequelize nodemon” el cual instalará las dependencias necesarias para el desarrollo de la actividad, las cuales son:
 - Express: Framework web para Node.js que te permite crear rápidamente aplicaciones web y APIs.
 - Mysql, Mysql2: Provee funcionalidades para conectarse y trabajar con una base de datos MySQL
 - Sequelize: Biblioteca que proporciona un ORM (Object-Relational Mapping) para trabajar con bases de datos relacionales, por ejemplo, mysql.

- Nodemon: herramienta que reinicia automáticamente el servidor de una aplicación Node.js cada vez que detecta cambios en los archivos del proyecto, solo es usado en entornos de desarrollo.

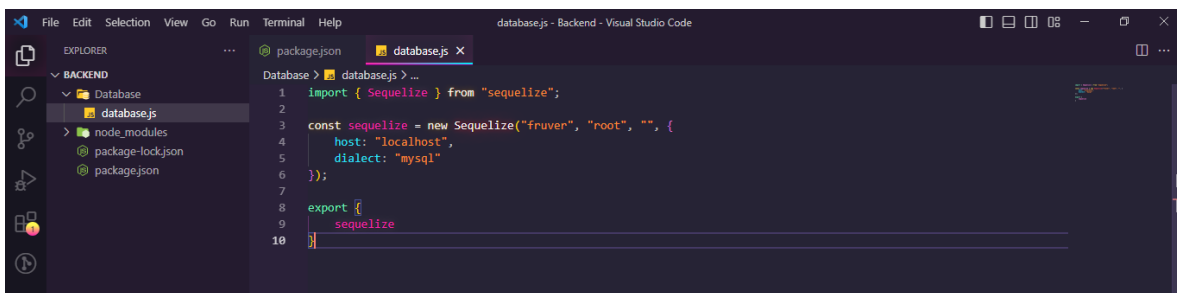


The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the project structure with folders like node_modules, package-lock.json, and package.json. The main editor area displays the package.json file with the following content:

```
1 {
2   "name": "backend",
3   "version": "1.0.0",
4   "description": "backend fruven",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "camilo",
10  "license": "ISC",
11  "dependencies": {
12    "express": "^4.18.2",
13    "mysql": "^2.18.1",
14    "mysql2": "^3.4.2",
15    "nodemon": "^2.0.22",
16    "sequelize": "^6.32.1"
17  }
18 }
```

The terminal window at the bottom shows the command `npm i express mysql mysql2 sequelize nodemon` being executed. The output indicates that 135 packages were added and 136 packages were audited in 15s. It also shows 12 packages looking for funding and 3 moderate severity vulnerabilities.

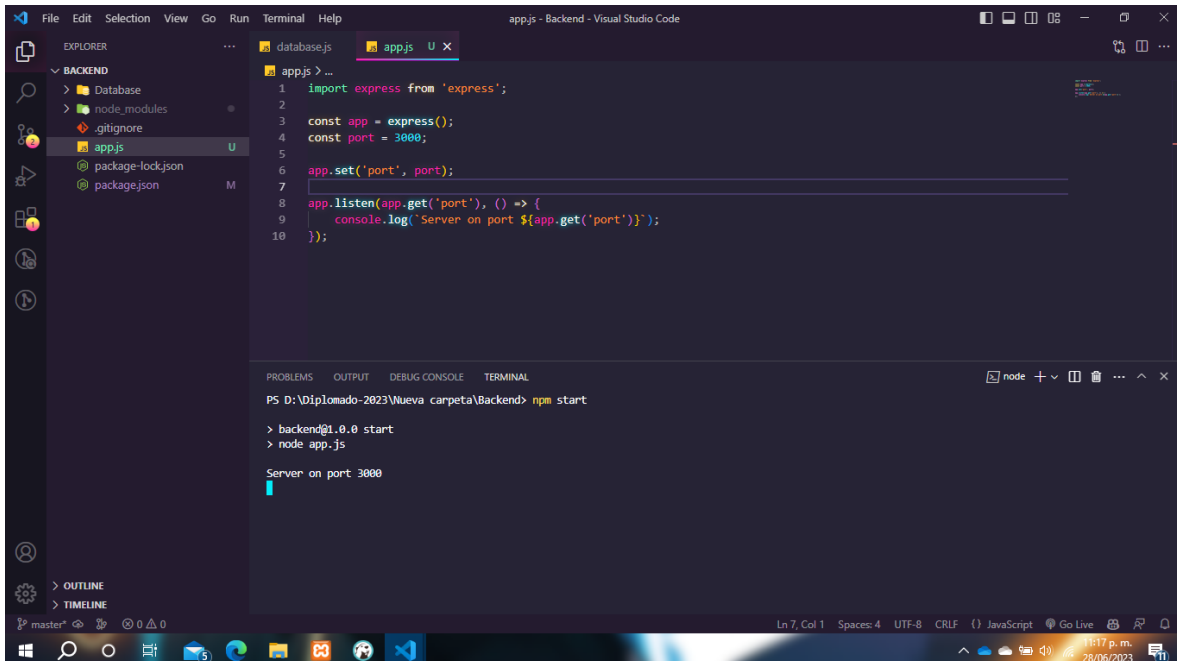
- Se crea un archivo database.js donde se configura la conexión con la base de datos.



The screenshot shows the Visual Studio Code interface with the Explorer panel on the left showing the project structure. The main editor area displays the database.js file with the following content:

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

- Crear el archivo que iniciará la aplicación node con el nombre que se definió al comienzo del punto 2 (app.js). En este archivo se define la lógica principal del proyecto.
 - se importa el módulo express, se define un puerto por el cual escuchara las solicitudes.
 - En la consola de vscode se escribe el comando “npm start” para iniciar la ejecución de una aplicación Node.js

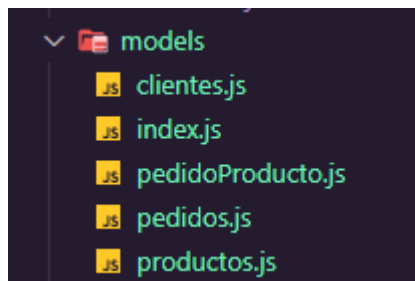


The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the project structure with files like database.js, app.js, node_modules, .gitignore, package-lock.json, and package.json. The main editor area displays the content of app.js, which imports express, sets a port of 3000, and listens for requests. The terminal at the bottom shows the command 'npm start' being executed, which runs 'node app.js', resulting in the output 'Server on port 3000'.

```
1 import express from 'express';
2
3 const app = express();
4 const port = 3000;
5
6 app.set('port', port);
7
8 app.listen(app.get('port'), () => {
9   console.log('Server on port ${app.get('port')}');
10 });
```

```
PS D:\Diplomado-2023\Nueva carpeta\Backend> npm start
> backend@1.0.0 start
> node app.js
Server on port 3000
```

- Crear los modelos de las tablas usando la biblioteca sequelize.



○ Clientes

```
models > clientes.js > ...
1 import { DataTypes } from 'sequelize';
2 import sequelize from '../Database/database.js';
3
4 const Cliente = sequelize.define('clientes', {
5   idCliente: {
6     type: DataTypes.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  identificacion: {
11    type: DataTypes.STRING(100),
12    allowNull: false
13  },
14  nombre: {
15    type: DataTypes.STRING(100),
16    allowNull: false
17  },
18  apellido: {
19    type: DataTypes.STRING(100),
20    allowNull: false
21  }
22 }, {
23   timestamps: false
24 });
25
26
27 export default Cliente;
28
29
```

○ Productos

```
models > productos.js > (e) default
1 import { DataTypes } from 'sequelize';
2 import sequelize from '../Database/database.js';
3
4 const Producto = sequelize.define('productos', {
5   idProducto: {
6     type: DataTypes.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  nombre: {
11    type: DataTypes.STRING(100),
12    allowNull: false
13  },
14  descripcion: {
15    type: DataTypes.STRING(100),
16    allowNull: false
17  },
18  precio: {
19    type: DataTypes.FLOAT,
20    allowNull: false
21  }
22 }, {
23   timestamps: false
24 });
25
26
27 export default Producto;
```

- Pedidos

```
models > pedidosjs > default
1 import { DataTypes } from 'sequelize';
2 import sequelize from '../Database/database.js';
3
4 const Pedido = sequelize.define('pedidos', {
5   idPedido: {
6     type: DataTypes.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  idCliente: {
11    type: DataTypes.INTEGER,
12    allowNull: false
13  },
14  fechaSolicitud: {
15    type: DataTypes.DATE,
16    allowNull: false
17  },
18  estado: {
19    type: DataTypes.INTEGER,
20    allowNull: false
21  }
22 }, {
23   timestamps: false
24 });
25
26
27 export default Pedido;
```

- Pedidosproductos

```
models > pedidoProductojs > default
1 import { DataTypes } from 'sequelize';
2 import sequelize from '../Database/database.js';
3
4 const PedidoProducto = sequelize.define('pedidosproductos', {
5   idPP: {
6     type: DataTypes.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  idPedido: {
11    type: DataTypes.INTEGER,
12    unique: true,
13  },
14  idProducto: {
15    type: DataTypes.INTEGER,
16    unique: true,
17  },
18  cantidad: {
19    type: DataTypes.INTEGER,
20    allowNull: false
21  }
22 }, {
23   timestamps: false
24 });
25
26
27 export default PedidoProducto;
```

- Creación de un archivo index.js dentro de la carpeta models donde se definen las relaciones entre las diferentes tablas y se exportan todos los modelos dentro de un objeto para un desarrollar la actividad de forma cómoda.

```
models > index.js > ...
1
2 import Cliente from './clientes.js';
3 import Producto from './productos.js';
4 import Pedido from './pedidos.js';
5 import PedidoProducto from './pedidoProducto.js';
6
7 // Relaciones
8 // cliente - pedido
9 Cliente.hasMany(Pedido, {
10   foreignKey: 'idCliente',
11   onDelete: 'CASCADE',
12   hooks: true
13 });
14
15 Pedido.belongsTo(Cliente, {
16   foreignKey: 'idCliente'
17 });
18
19 // pedido - pedidoProducto
20 Pedido.hasMany(PedidoProducto, {
21   foreignKey: 'idPedido',
22   onDelete: 'CASCADE',
23   hooks: true
24 });
25
26 PedidoProducto.belongsTo(Pedido, {
27   foreignKey: 'idPedido'
28 });
29
30 // producto - pedidoProducto
31 Producto.hasMany(PedidoProducto, {
32   foreignKey: 'idProducto'
```

```
31 // producto - pedidoProducto
32 Producto.hasMany(PedidoProducto, {
33   foreignKey: 'idProducto',
34   onDelete: 'CASCADE',
35   hooks: true
36 });
37
38 PedidoProducto.belongsTo(Producto, {
39   foreignKey: 'idProducto'
40 });
41
42
43
44 export default {
45   Cliente,
46   Producto,
47   Pedido,
48   PedidoProducto
49 };
```

- Creación de los controladores que interactuaran con la base de datos.

```

  controllers
  ├── clientes.controller.js
  ├── pedidos.controller.js
  ├── pedidosProd.controller.js
  └── productos.controller.js

```

- Clientes

```
controllers > clientes.controller.js > getClientes
1 import Models from '../models/index.js';
2
3 const getClientes = async (req, res) => {
4   try {
5     const clientes = await Models.Cliente.findAll();
6     res.status(200).json(clientes);
7   } catch (error) {
8     console.log(error);
9     res.status(500).json(error);
10  }
11 }
12
13 const postCliente = async (req, res) => {
14   try {
15     const cliente = await Models.Cliente.create(req.body);
16     res.status(200).json(cliente);
17   } catch (error) {
18     console.log(error);
19     res.status(500).json(error);
20   }
21 }
22
23 const putCliente = async (req, res) => {
24   try {
25     let idCliente = req.params.id;
26     let cliente = await Models.Cliente.findByPk(idCliente);
27     cliente = await cliente.update(req.body);
28     res.status(200).json(cliente);
29   } catch (error) {
30     console.log(error);
31     res.status(500).json(error);
32   }
33 }
34
35 const deleteCliente = async (req, res) => {
36   try {
37     let idCliente = req.params.id;
38     let cliente = await Models.Cliente.findByPk(idCliente);
39     cliente = await cliente.destroy();
40     res.status(200).json(cliente);
41   } catch (error) {
42     console.log(error);
43     res.status(500).json(error);
44   }
45 }
46
47
48 export default {
49   getClientes,
50   postCliente,
51   putCliente,
52   deleteCliente
53 };
```

Ln 49, Col 17 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live 10:40 p. m. 3/07/2023

○ Productos

```
controllers > productos.controller.js > postProducto
1  import Models from '../models/index.js';
2
3  const getProductos = async (req, res) => {
4    try {
5      const productos = await Models.Producto.findAll();
6      res.status(200).json(productos);
7    } catch (error) {
8      res.status(500).json(error);
9      console.log(error);
10   }
11 }
12
13 const postProducto = async (req, res) => {
14   try {
15     const producto = await Models.Producto.create(req.body);
16     res.status(200).json(producto);
17   } catch (error) {
18     res.status(500).json(error);
19     console.log(error);
20   }
21 }
22
```

```
controllers > productos.controller.js > deleteProducto
23 const putProducto = async (req, res) => {
24   try {
25     let idProducto = req.params.id;
26     let producto = await Models.Producto.findByPk(idProducto);
27     producto = await producto.update(req.body);
28     res.status(200).json(producto);
29   } catch (error) {
30     res.status(500).json(error);
31     console.log(error);
32   }
33 }
34
35
36 const deleteProducto = async (req, res) => {
37   try {
38     let idProducto = req.params.id;
39     let producto = await Models.Producto.findByPk(idProducto);
40     producto = await producto.destroy();
41     res.status(200).json(producto);
42   } catch (error) {
43     res.status(500).json(error);
44     console.log(error);
45   }
46 }
47
48 export default {
49   getProductos,
50   postProducto,
51   putProducto,
52   deleteProducto
53 };

```

Ln 46, Col 2 Spaces: 4 UTF-8 CRLF JavaScript Go Live 10:42 p. m. 3/07/2023

○ Pedidos

```
controllers > pedidos.controller.js > getPedidos
1 import Models from '../models/index.js';
2
3 const getPedidos = async (req, res) => {
4   try {
5     const pedidos = await Models.Pedido.findAll({
6       include: [
7         {
8           model: Models.Cliente
9         }
10      ]
11    });
12    res.status(200).json(pedidos);
13  } catch (error) {
14    console.log(error);
15    res.status(500).json(error);
16  }
17 }
18
19 const postPedido = async (req, res) => {
20   try {
21     const pedido = await Models.Pedido.create(req.body);
22     res.status(200).json(pedido);
23   } catch (error) {
24     console.log(error);
25     res.status(500).json(error);
26   }
27 }
28
```

```
controllers > pedidos.controller.js > deletePedido
29 const putPedido = async (req, res) => {
30   try {
31     let idPedido = req.params.id;
32     let pedido = await Models.Pedido.findByPk(idPedido);
33     pedido = await pedido.update(req.body);
34     res.status(200).json(pedido);
35   } catch (error) {
36     console.log(error);
37     res.status(500).json(error);
38   }
39 }
40
41 const deletePedido = async (req, res) => {
42   try {
43     let idPedido = req.params.id;
44     let pedido = await Models.Pedido.findByPk(idPedido);
45     pedido = await pedido.destroy();
46     res.status(200).json(pedido);
47   } catch (error) {
48     console.log(error);
49     res.status(500).json(error);
50   }
51 }
52
53 export default {
54   getPedidos,
55   postPedido,
56   putPedido,
57   deletePedido
58 };
59
```

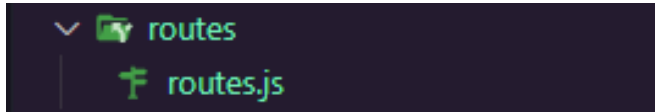
Ln 44, Col 61 Spaces: 4 UTF-8 CRLF JavaScript Go Live 10:46 p. m. 3/07/2023

- Pedidos_productos

```
controllers > | pedidosProd.controller.js > | postPedidoProd
1  import Models from '../models/index.js';
2
3  const getPedidosProd = async (req, res) => {
4    try {
5      const pedidosProd = await Models.PedidoProducto.findAll({
6        include: [
7          {
8            model: Models.Pedido
9          },
10         {
11           model: Models.Producto
12         }
13       ]
14     });
15     res.status(200).json(pedidosProd);
16   } catch (error) {
17     console.log(error);
18     res.status(500).json(error);
19   }
20 }
21
22 const postPedidoProd = async (req, res) => {
23   try {
24     const pedidoProd = await Models.PedidoProducto.create(req.body);
25     res.status(200).json(pedidoProd);
26   } catch (error) {
27     console.log(error);
28     res.status(500).json(error);
29   }
30 }
31
```

```
controllers > | pedidosProd.controller.js > ...
31
32 const putPedidoProd = async (req, res) => {
33   try {
34     let idPedidoProd = req.params.id;
35     let pedidoProd = await Models.PedidoProducto.findByPk(idPedidoProd);
36     pedidoProd = await pedidoProd.update(req.body);
37     res.status(200).json(pedidoProd);
38   } catch (error) {
39     console.log(error);
40     res.status(500).json(error);
41   }
42 }
43
44
45 const deletePedidoProd = async (req, res) => {
46   try {
47     let idPedidoProd = req.params.id;
48     let pedidoProd = await Models.PedidoProducto.findByPk(idPedidoProd);
49     pedidoProd = await pedidoProd.destroy();
50     res.status(200).json(pedidoProd);
51   } catch (error) {
52     console.log(error);
53     res.status(500).json(error);
54   }
55 }
56
57 export default {
58   getPedidosProd,
59   postPedidoProd,
60   putPedidoProd,
61   deletePedidoProd
62 };
Ln 56, Col 1  Spaces: 4  UTF-8  CRLF  () JavaScript  Go Live  10:48 p. m.
3/07/2023
```


- Crear un archivo routes.js, donde se gestionarán las rutas y las solicitudes entrantes asociando las URL de una aplicación con las funciones o controladores correspondientes.



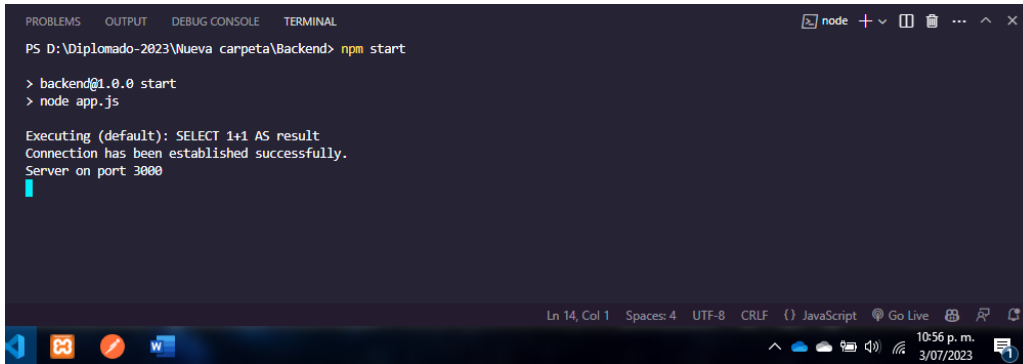
```
routes > routes.js > ...
1  import { Router } from 'express';
2  import pedidosController from '../controllers/pedidos.controller.js';
3  import pedidosProdController from '../controllers/pedidosProd.controller.js';
4  import productosController from '../controllers/productos.controller.js';
5  import clientesController from '../controllers/clientes.controller.js';
6
7  const router = Router();
8
10 // clientes
11 router.get('/clientes', clientesController.getClientes);
12 router.post('/clientes', clientesController.postCliente);
13 router.put('/clientes/:id', clientesController.putCliente);
14 router.delete('/clientes/:id', clientesController.deleteCliente);
15
16 // productos
17 router.get('/productos', productosController.getProductos);
18 router.post('/productos', productosController.postProducto);
19 router.put('/productos/:id', productosController.putProducto);
20 router.delete('/productos/:id', productosController.deleteProducto);
21
22 // pedidos
23 router.get('/pedidos', pedidosController.getPedidos);
24 router.post('/pedidos', pedidosController.postPedido);
25 router.put('/pedidos/:id', pedidosController.putPedido);
26 router.delete('/pedidos/:id', pedidosController.deletePedido);
27
28 // pedidosProd
29 router.get('/pedidosProd', pedidosProdController.getPedidosProd);
30 router.post('/pedidosProd', pedidosProdController.postPedidoProd);
31 router.put('/pedidosProd/:id', pedidosProdController.putPedidoProd);
32 router.delete('/pedidosProd/:id', pedidosProdController.deletePedidoProd);
33
34
35 export default router;
```

Ln 35, Col 23 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live 10:52 p. m. 3/07/2023

3. Verificación.

Usando el programa Postman se realizarán solicitudes verificando las operaciones de GET, POST, PUT y DELETE de cada uno de los modelos.

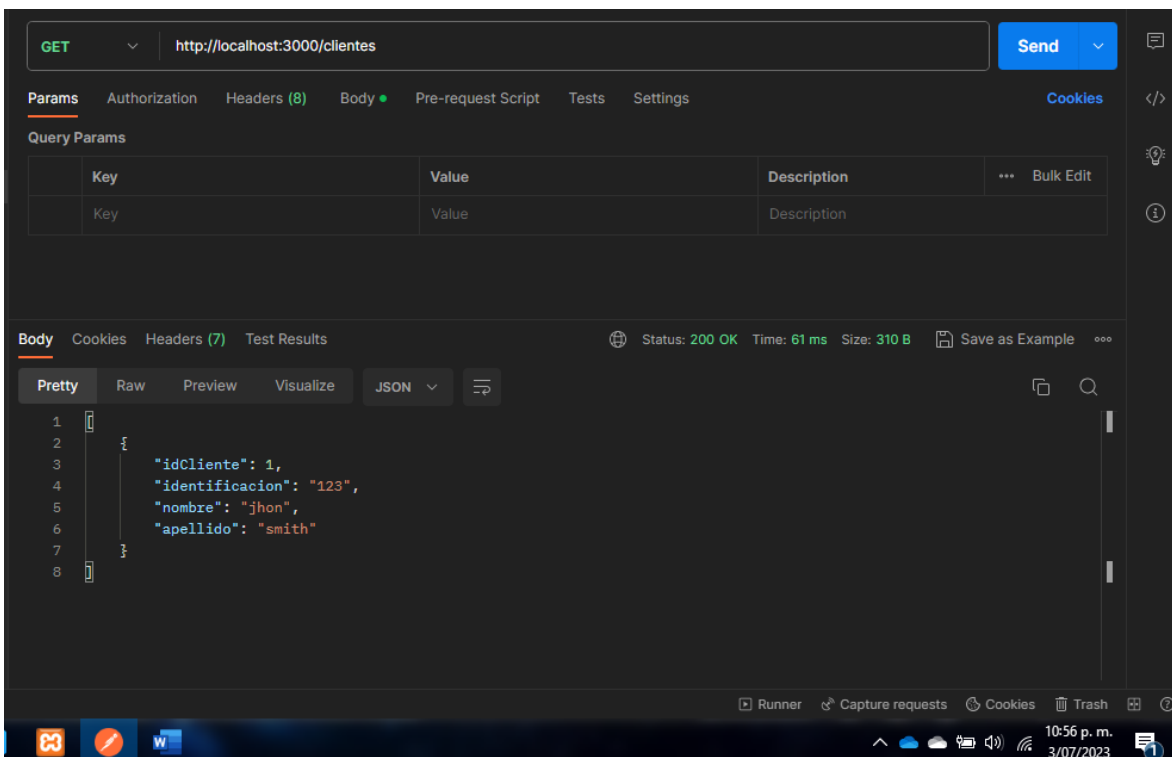
Iniciar el servidor con el comando npm start.



```
PS D:\Diplomado-2023\Nueva carpeta\Backend> npm start
> backend@1.0.0 start
> node app.js

Executing (default): SELECT 1+1 AS result
Connection has been established successfully.
Server on port 3000
```

- **Clientes**



GET Send

Params Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 61 ms Size: 310 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "idCliente": 1,
3   "identificacion": "123",
4   "nombre": "jhon",
5   "apellido": "smith"
6 }
7
8
```

POST http://localhost:3000/clientes Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1
2
3
4
5
6
[{"identificacion": "789",
  "nombre": "Edison",
  "apellido": "Rosero"}]
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 249 ms Size: 311 B Save as Example

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
{"idCliente": 2,
  "identificacion": "789",
  "nombre": "Edison",
  "apellido": "Rosero"}
```

Runner Capture requests Cookies Trash 10:59 p. m. 3/07/2023

PUT http://localhost:3000/clientes/2 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1
2
3
4
[{"nombre": "Edison Camilo"}]
```

Body Cookies Headers (7) Test Results 200 OK 43 ms 318 B Save as Example

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
{"idCliente": 2,
  "identificacion": "789",
  "nombre": "Edison Camilo",
  "apellido": "Rosero"}
```

Runner Capture requests Cookies Trash 11:01 p. m. 3/07/2023

DELETE ▼ http://localhost:3000/clientes/2 Send ▼

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON ▼ Beautify

```
1
```

Body Cookies Headers (7) Test Results 200 OK 23 ms 318 B Save as Example ⋮

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "idCliente": 2,
3   "identificacion": "789",
4   "nombre": "Edison Camilo",
5   "apellido": "Rosero"
6 }
```

Runner Capture requests Cookies Trash

11:02 p. m. 3/07/2023

- **Productos**

GET ▼ http://localhost:3000/productos Send ▼

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 200 OK Time: 30 ms Size: 384 B Save as Example ⋮

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   {
3     "idProducto": 1,
4     "nombre": "manzanas",
5     "descripcion": "verdes",
6     "precio": 5000
7   },
8   {
9     "idProducto": 2,
10    "nombre": "manzanas",
11    "descripcion": "rojas",
12    "precio": 4000
13  }
14 }
```

Runner Capture requests Cookies Trash

11:14 p. m. 3/07/2023

POST http://localhost:3000/productos Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1
2
3
4
5
6
[{"id": 3, "nombre": "Uva", "descripcion": "Merlot", "precio": 10000}]
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 41 ms Size: 304 B Save as Example

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
{"idProducto": 3, "nombre": "Uva", "descripcion": "Merlot", "precio": 10000}
```

Runner Capture requests Cookies Trash 11:15 p. m. 3/07/2023

PUT http://localhost:3000/productos/1 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1
2
3
4
[{"id": 1, "nombre": "manzanas", "descripcion": "Reineta", "precio": 5000}]
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 24 ms Size: 309 B Save as Example

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
{"idProducto": 1, "nombre": "manzanas", "descripcion": "Reineta", "precio": 5000}
```

Runner Capture requests Cookies Trash 11:16 p. m. 3/07/2023

DELETE ▼ http://localhost:3000/productos/1 Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies </>

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 22 ms 309 B Save as Example ...

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "idProducto": 1,
3   "nombre": "manzanas",
4   "descripcion": "Reineta",
5   "precio": 5000
6 }
```

Runner Capture requests Cookies Trash 11:18 p. m. 3/07/2023

- Pedidos

GET ▼ http://localhost:3000/pedidos Send ▼

Params Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies </>

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 200 OK Time: 27 ms Size: 416 B Save as Example ...

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "idPedido": 3,
3   "idCliente": 3,
4   "fechaSolicitud": "2023-07-03T00:00:00.000Z",
5   "estado": 0,
6   "cliente": {
7     "idCliente": 3,
8     "identificacion": "1085",
9     "nombre": "Edison Camilo",
10    "apellido": "Rosero"
11   }
12 }
13
14 }
```

Runner Capture requests Cookies Trash 11:21 p. m. 3/07/2023

POST http://localhost:3000/pedidos

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "idCliente": "4",
3   "fechaSolicitud": "2023-07-12",
4   "estado": 0
5 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 17 ms Size: 320 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "idPedido": 4,
3   "idCliente": "4",
4   "fechaSolicitud": "2023-07-12T00:00:00.000Z",
5   "estado": 0
6 }
```

Runner Capture requests Cookies Trash 11:22 p.m. 3/07/2023

PUT http://localhost:3000/pedidos/5

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

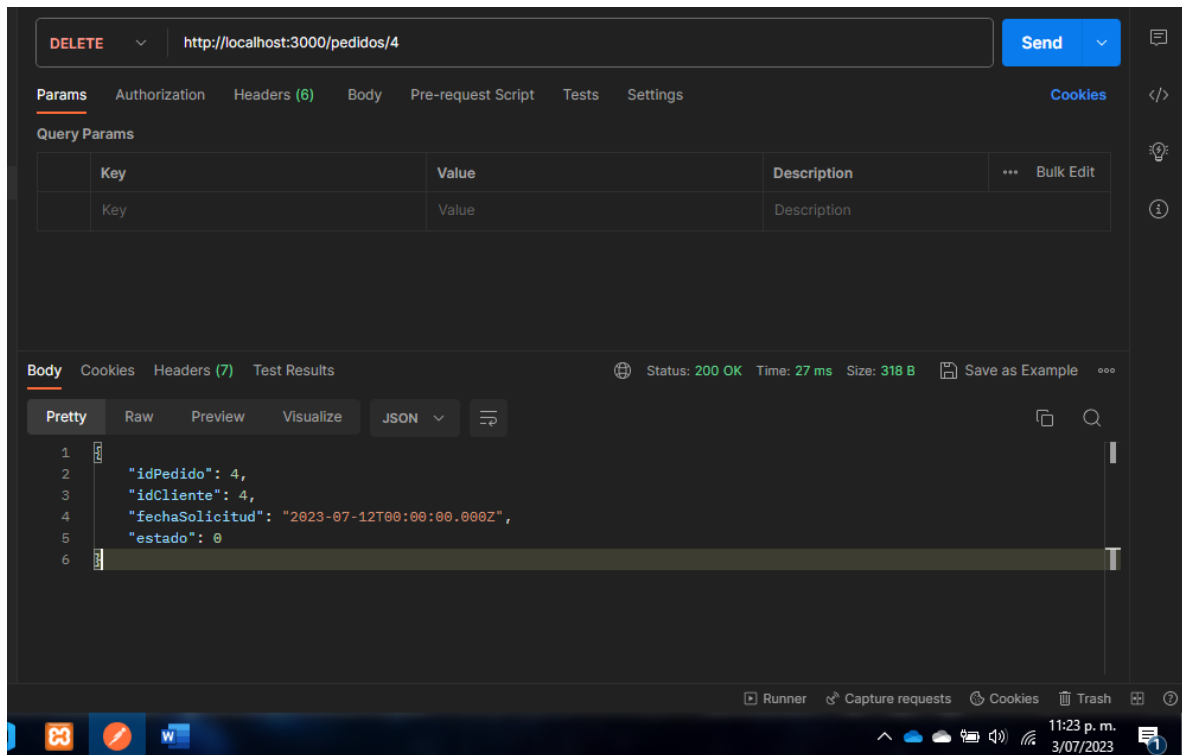
```
1 {
2   "estado": 2
3 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 44 ms Size: 318 B Save as Example

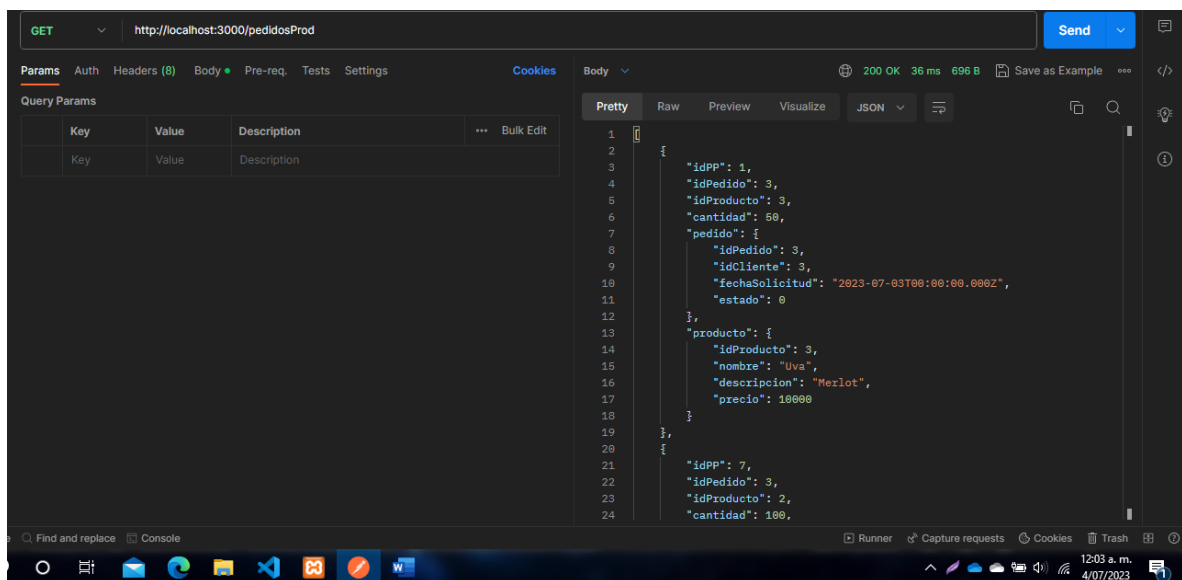
Pretty Raw Preview Visualize JSON

```
1 {
2   "idPedido": 5,
3   "idCliente": 4,
4   "fechaSolicitud": "2023-05-12T00:00:00.000Z",
5   "estado": 2
6 }
```

Runner Capture requests Cookies Trash 12:29 a.m. 4/07/2023



- Pedidos producto



POST http://localhost:3000/pedidosProd

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   ... "idPedido": 5,
3   ... "idProducto": 4,
4   "cantidad": 20
5 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 24 ms Size: 287 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "idPP": 8,
3   "idPedido": 5,
4   "idProducto": 4,
5   "cantidad": 20
6 }
```

Runner Capture requests Cookies Trash 12:08 a.m. 4/07/2023

PUT http://localhost:3000/pedidosProd/1

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   ... "cantidad": 1000
3 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 24 ms Size: 289 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "idPP": 1,
3   "idPedido": 3,
4   "idProducto": 3,
5   "cantidad": 1000
6 }
```

Runner Capture requests Cookies Trash 12:10 a.m. 4/07/2023

DELETE

http://localhost:3000/pedidosProd/7

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

Status: 200 OK

Time: 42 ms

Size: 288 B

Save as Example

Pretty

Raw

Preview

Visualize

JSON

```
1  [
2    "idPP": 7,
3    "idPedido": 3,
4    "idProducto": 2,
5    "cantidad": 100
6  ]
```

Runner

Capture requests

Cookies

Trash

12:11 a.m.

4/07/2023