

## **Informe BackEnd FruVerFresh**

**Presentado por:**

**David Santiago Reyes Lasso**

**Presentado a:**

**Ing. Vicente Aux Revelo**



**Universidad de Nariño**

**Facultad de Ingeniería**

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

**Pasto, 2023**

**1. Crear una base de datos MYSQL/MARIADB que permita llevar el registro de un FRUVER (FRUTAS Y VERDURAS), así como también el proceso de solicitud de compra de estas.**

### Base de datos FruVerFresh

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
pedidos	Examinar   Estructura   Buscar   Insertar   Vaciar   Eliminar	2	InnoDB	utf8mb4_general_ci	32,0 kB	-
productos	Examinar   Estructura   Buscar   Insertar   Vaciar   Eliminar	10	InnoDB	utf8mb4_general_ci	16,0 kB	-
usuarios	Examinar   Estructura   Buscar   Insertar   Vaciar   Eliminar	7	InnoDB	utf8mb4_general_ci	16,0 kB	-
3 tablas - Número de filas		19	InnoDB	utf8mb4_general_ci	64,0 kB	0 B

### Estructura de la tabla pedidos

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_pedido	int(10)			No	Ninguna		AUTO_INCREMENT	Cambiar    Eliminar   Más
2	nombre_cliente_pedido	varchar(45)	utf8mb4_general_ci		No	Ninguna		Cambiar    Eliminar   Más	
3	apellido_cliente_pedido	varchar(45)	utf8mb4_general_ci		No	Ninguna		Cambiar    Eliminar   Más	
4	telefono_cliente_pedido	varchar(12)	utf8mb4_general_ci		No	Ninguna		Cambiar    Eliminar   Más	
5	correo_cliente_pedido	varchar(45)	utf8mb4_general_ci		No	Ninguna		Cambiar    Eliminar   Más	
6	direccion_cliente_pedido	varchar(45)	utf8mb4_general_ci		No	Ninguna		Cambiar    Eliminar   Más	
7	producto_pedido	varchar(45)	utf8mb4_general_ci		No	Ninguna		Cambiar    Eliminar   Más	
8	cantidad_pedido	int(11)			No	Ninguna		Cambiar    Eliminar   Más	
9	precio_pedido	decimal(10,0)			No	Ninguna		Cambiar    Eliminar   Más	

## Estructura tabla productos

The screenshot shows the phpMyAdmin interface for the 'db\_fruver' database. The left sidebar lists databases and tables, including 'bdcolombiatravel', 'database', 'db\_fruver' (containing 'pedidos' and 'productos'), 'fruver', 'information\_schema', 'mysql', 'performance\_schema', 'phpmyadmin', 'test', and 'webtesis'. The main panel displays the structure of the 'productos' table:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_producto	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar  Eliminar  Más
2	nombre_producto	varchar(45)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
3	detalle_producto	varchar(800)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
4	precio_producto	decimal(10,0)			No	Ninguna			Cambiar  Eliminar  Más
5	foto_producto	varchar(255)	utf8mb4_general_ci		Sí	NULL			Cambiar  Eliminar  Más

Below the table structure, there are buttons for 'Imprimir', 'Planteamiento de la estructura de tabla', 'Mover columnas', 'Normalizar', 'Agregar', and 'Continuar'. The bottom section shows the index configuration:

Acción	Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
Editar  Renombrar  Eliminar	PRIMARY	BTREE	Sí	No	id_producto	10	A	No	

Buttons for 'Crear un índice en' and 'Continuar' are also present.

## Estructura tabla usuarios

The screenshot shows the phpMyAdmin interface for the 'db\_fruver' database. The left sidebar lists databases and tables, including 'bdcolombiatravel', 'database', 'db\_fruver' (containing 'pedidos' and 'usuarios'), 'fruver', 'information\_schema', 'mysql', 'performance\_schema', 'phpmyadmin', 'test', and 'webtesis'. The main panel displays the structure of the 'usuarios' table:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_usuario	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar  Eliminar  Más
2	nombre_usuario	varchar(255)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
3	apellido_usuario	varchar(255)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
4	edad_usuario	int(11)			No	Ninguna			Cambiar  Eliminar  Más
5	telefono_usuario	varchar(255)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
6	correo_usuario	varchar(255)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
7	contraseña_usuario	varchar(255)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
8	tipo_usuario	varchar(255)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más

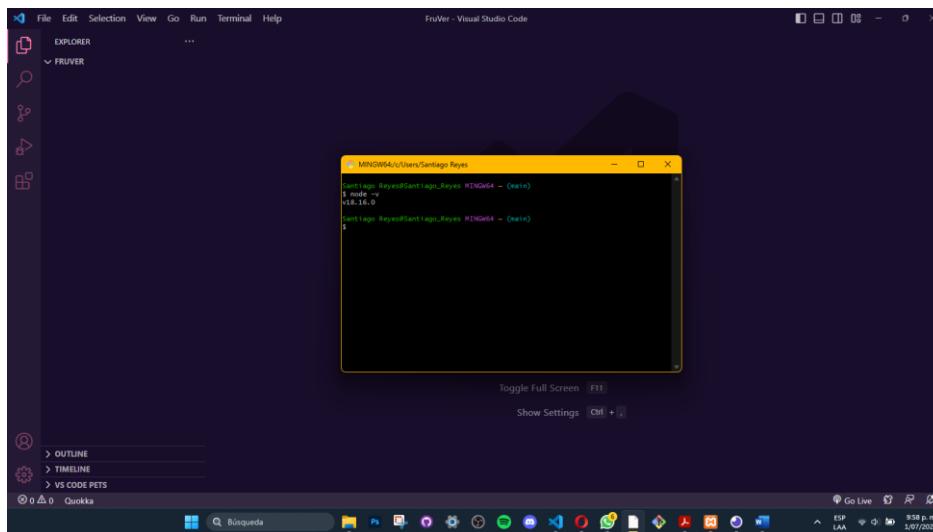
Below the table structure, there are buttons for 'Imprimir', 'Planteamiento de la estructura de tabla', 'Mover columnas', 'Normalizar', 'Agregar', and 'Continuar'. The bottom section shows the index configuration:

Acción	Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
Console  Renombrar  Eliminar	PRIMARY	BTREE	Sí	No	id_usuario	9	A	No	

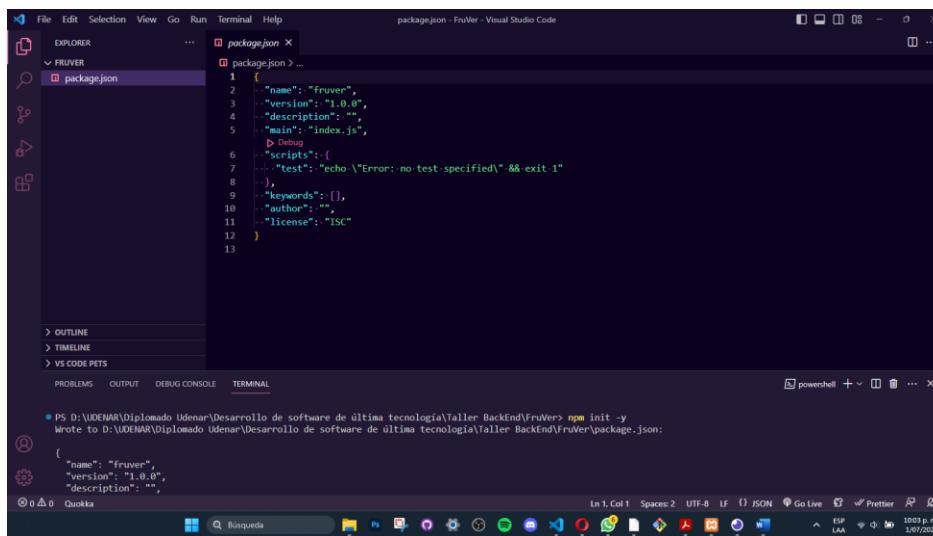
Buttons for 'Crear un índice en' and 'Continuar' are also present.

**2. Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS que haga uso de la base de datos del primer punto y que permita el desarrollo de todas las tareas asociadas al registro y administración de las frutas y verduras (La empresa debe contar con un nombre). Se debe hacer uso correcto de los verbos HTTP dependiendo de la tarea a realizar.**

### Versión NodeJs



### Inicializando proyecto con Node.js



## Instalando Express.js

```
package.json - FruVer - Visual Studio Code
package.json > author
1 {
2   "name": "fruver",
3   "version": "1.0.0",
4   "description": "Proyecto tienda virtual FruVer",
5   "main": "server.js",
6     ▶ Debug
7   "scripts": {
8     "test": "echo \\\"Error: no test specified\\\" && exit 1"
9   },
10  "keywords": [],
11  "author": "Santiago Reyes L.",
12  "license": "ISC",
13  "dependencies": {
14    "express": "^4.18.2"
15  }
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer> `npm i express`  
added 58 packages, and audited 59 packages in 4s  
8 packages are looking for funding  
run `'npm fund'` for details

Quokka

Búsqueda

Ln 10, Col 31 Spaces:2 UTF-8 LF JSON Go Live Prettier ESP LAA 10:10 p.m. 1/07/2023

## Instalando MySQL y MySQL2

```
package.json - FruVer - Visual Studio Code
package.json > author
1 {
2   "name": "fruver",
3   "version": "1.0.0",
4   "description": "Proyecto tienda virtual FruVer",
5   "main": "server.js",
6     ▶ Debug
7   "scripts": {
8     "test": "echo \\\"Error: no test specified\\\" && exit 1"
9   },
10  "keywords": [],
11  "author": "Santiago Reyes L.",
12  "license": "ISC",
13  "dependencies": {
14    "express": "^4.18.2",
15    "mysql": "^2.18.1",
16    "mysql2": "^2.18.1"
17  }
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

added 58 packages, and audited 59 packages in 4s  
PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer> `npm i mysql`  
added 12 packages, and audited 71 packages in 3s  
8 packages are looking for funding  
run `'npm fund'` for details

Quokka

Búsqueda

Ln 10, Col 31 Spaces:2 UTF-8 LF JSON Go Live Prettier ESP LAA 10:12 p.m. 1/07/2023

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "FRUVER".
- Editor:** Displays the content of the `package.json` file.
- Terminal:** Shows the command output:

```
added 11 packages, and audited 82 packages in 4s  
8 packages are looking for funding  
  run `npm fund` for details  
found 0 vulnerabilities
```
- Bottom Bar:** Includes icons for search, file operations, and system status.

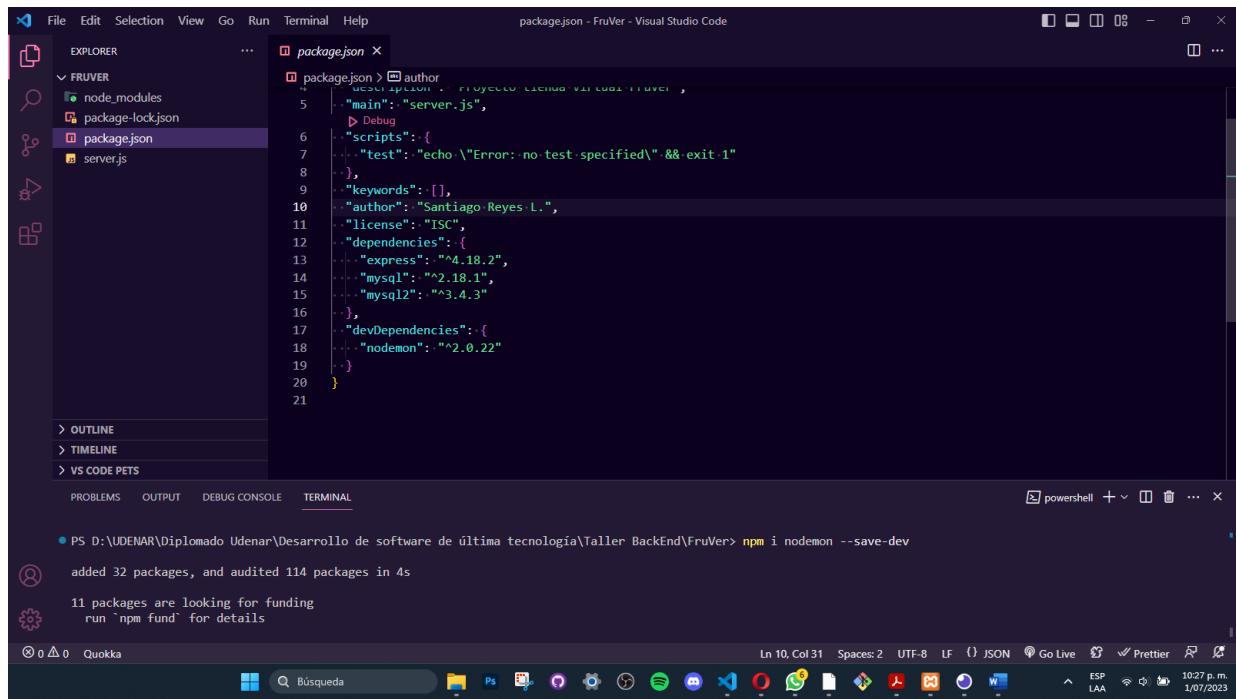
## Creando archivo server.js

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "FRUVER".
- Editor:** Displays the content of the `server.js` file.
- Terminal:** Shows the command output:

```
PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de ultima tecnologia\Taller BackEnd\FruVer>
```
- Bottom Bar:** Includes icons for search, file operations, and system status.

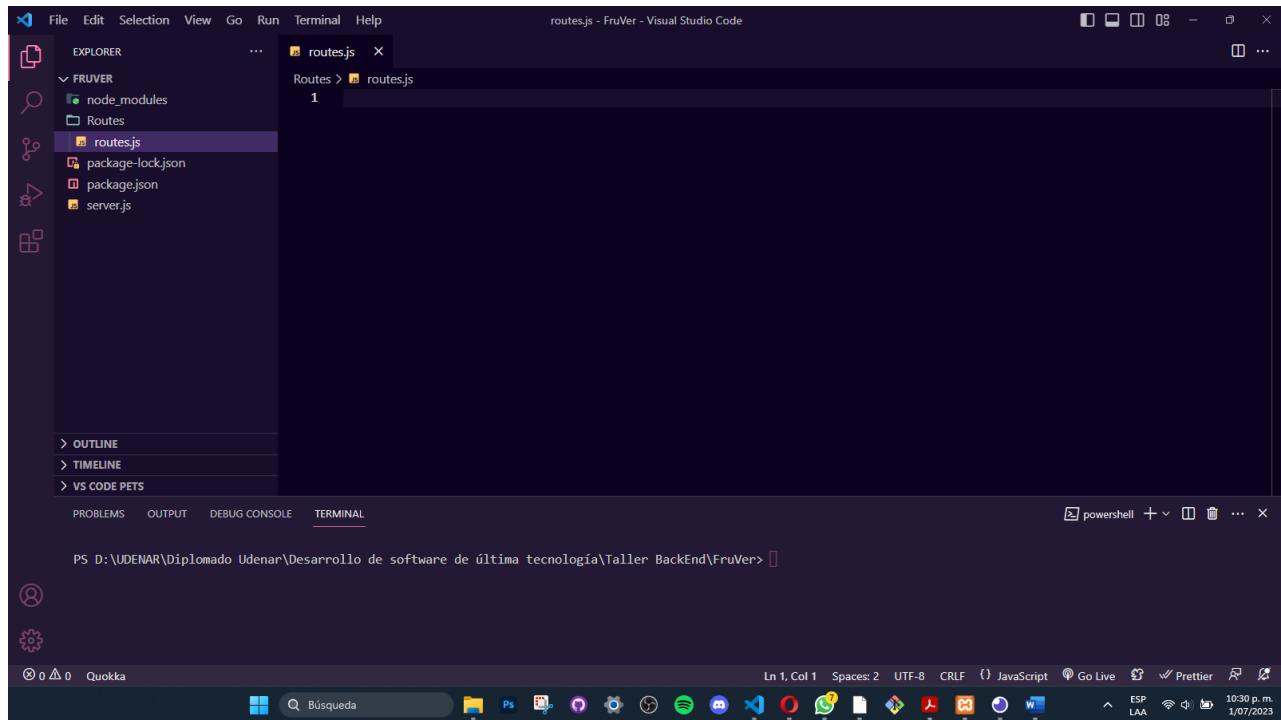
## Instalando Nodemon



```
package.json
{
  "name": "FRUVER",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "Santiago Reyes L.",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "mysql": "^2.18.1",
    "mysql2": "^3.4.3"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}

PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer> npm i nodemon --save-dev
added 32 packages, and audited 114 packages in 4s
11 packages are looking for funding
  run 'npm fund' for details
```

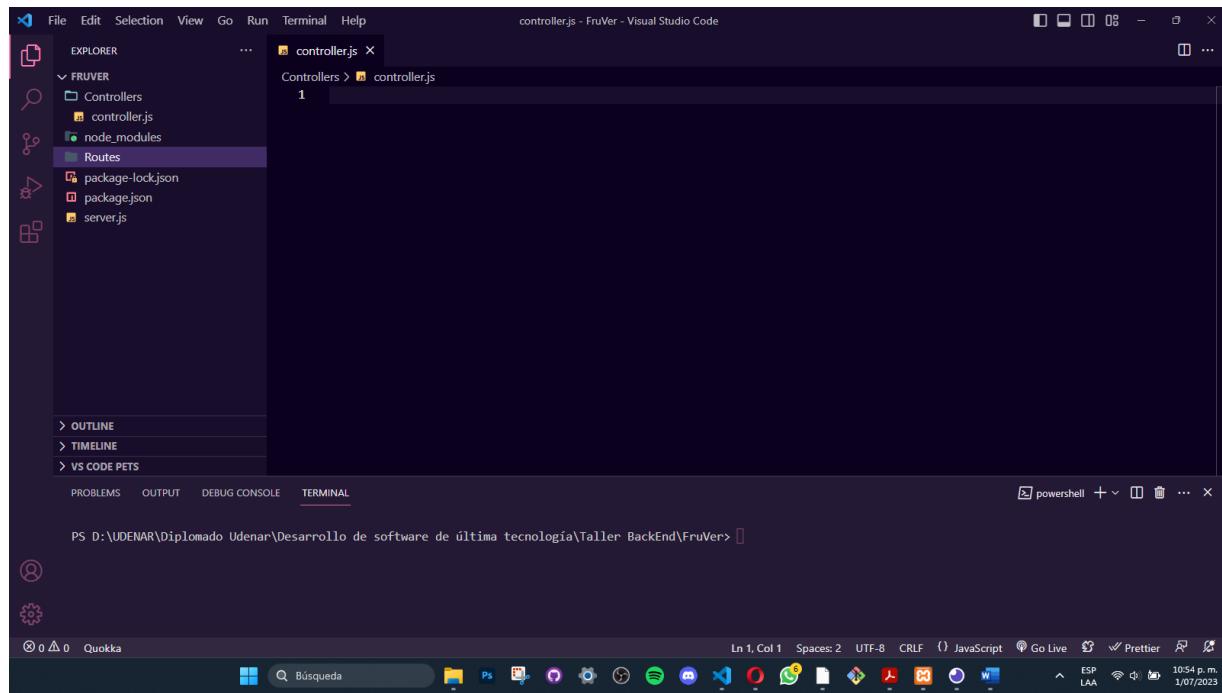
## Creando carpeta para rutas



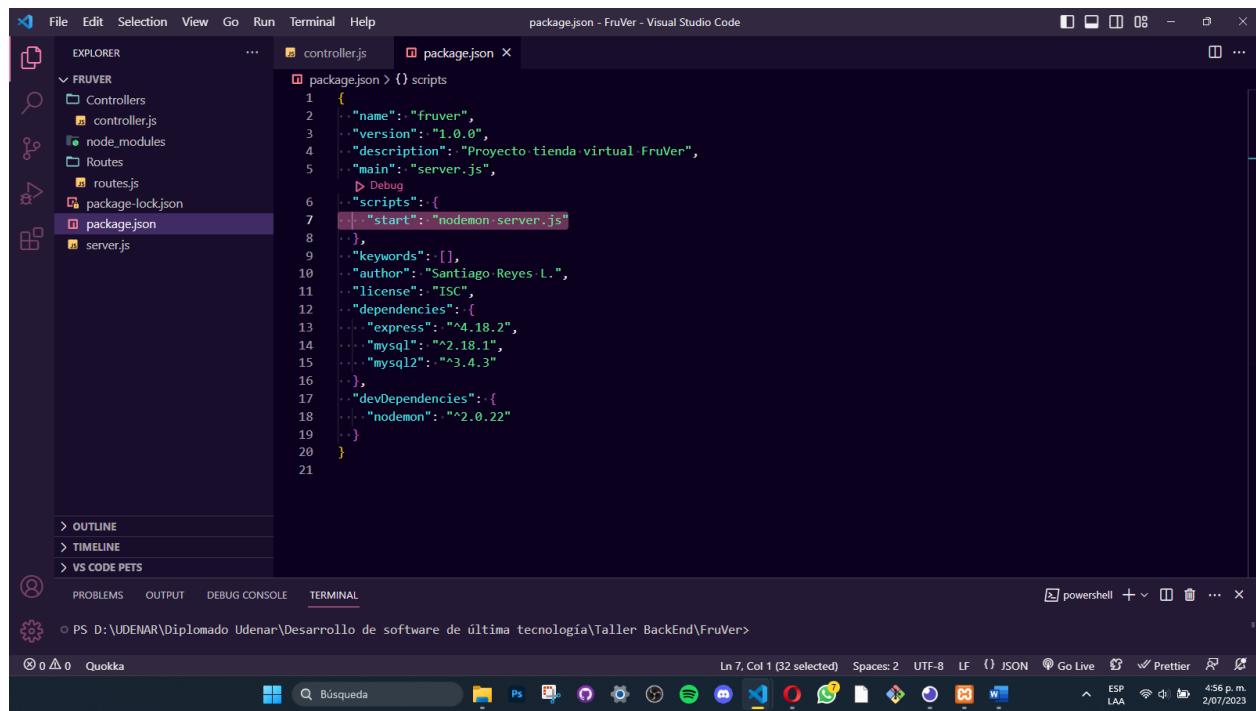
```
routes.js
1

PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer> npm i
```

## Creando carpeta para controladores



## Definiendo script Nodemon



## Habilitando los módulos

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under the folder "FRUVER".
- Editor:** Displays the contents of the `package.json` file.
- Code:**

```
1  {
2    . . .
3    . . .
4    . . .
5    . . .
6    "type": "module",
7    . . .
8    . . .
9    . . .
10   . . .
11   . . .
12   . . .
13   . . .
14   . . .
15   . . .
16   . . .
17   . . .
18   . . .
19   . . .
20   . . .
21 }
```
- Terminal:** Shows the command `PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer>`.
- Bottom Status Bar:** Includes icons for Quokka, PowerShell, and various system status indicators like battery level and signal strength.

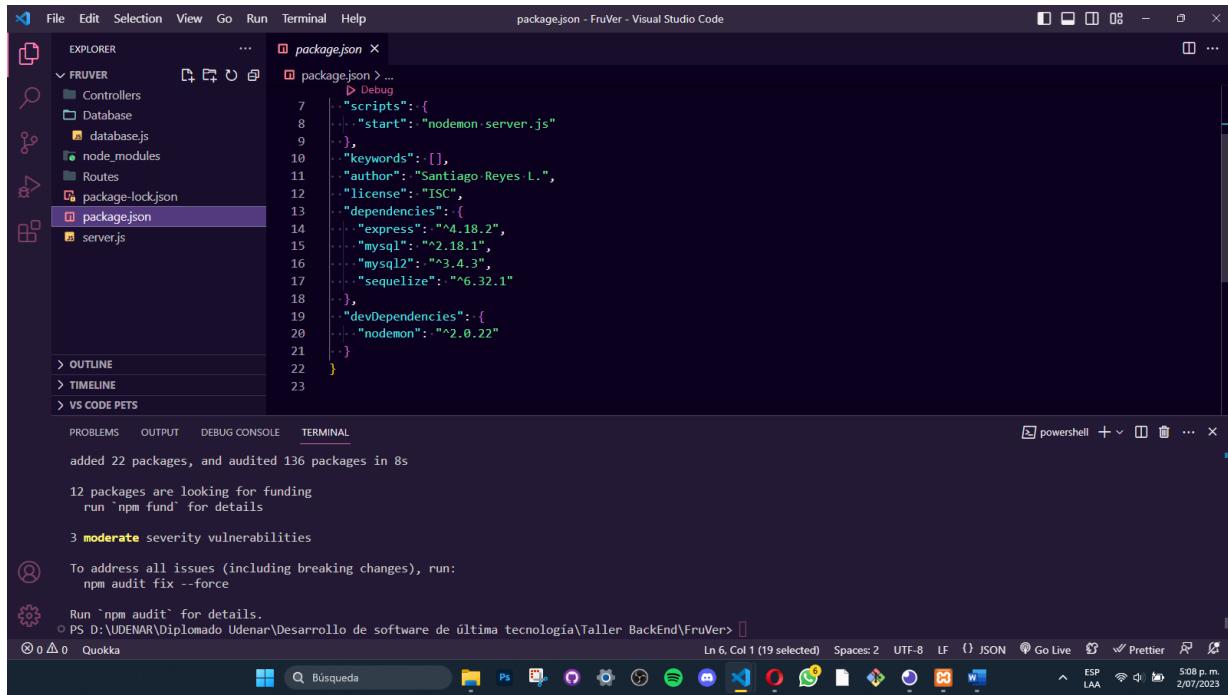
## Creando carpeta para la conexión a la BD

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under the folder "FRUVER". A new folder named "Database" has been created and is selected.
- Editor:** Displays the contents of the `database.js` file.
- Code:**

```
1
```
- Terminal:** Shows the command `PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer>`.
- Bottom Status Bar:** Includes icons for Quokka, PowerShell, and various system status indicators like battery level and signal strength.

## Instalando ORM Sequelize



```
File Edit Selection View Go Run Terminal Help package.json - FruVer - Visual Studio Code

EXPLORER package.json ...
FRUVER
  Controllers
  Database
    database.js
  node_modules
  Models
  Routes
  package-lock.json
  package.json
  server.js

package.json > OUTLINE > TIMELINE > VS CODE PETS

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
added 22 packages, and audited 136 packages in 8s

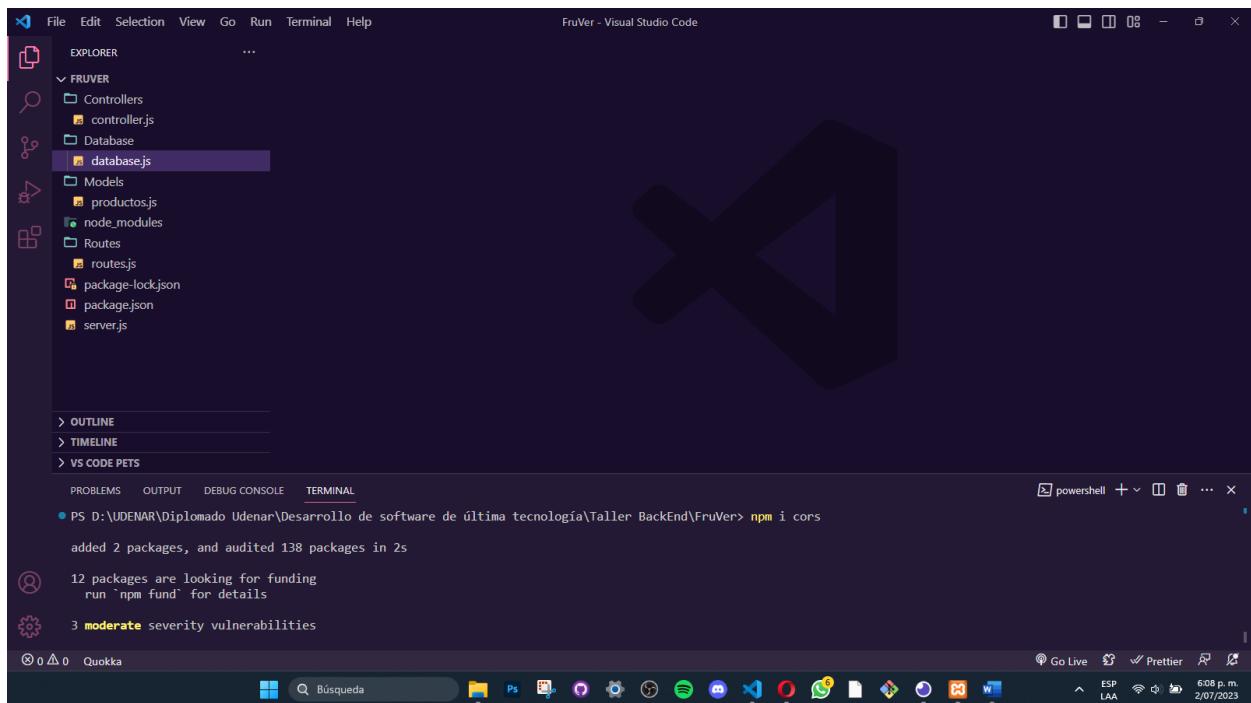
12 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
  ○ PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer> 
Ln 6, Col 1 (19 selected) Spaces: 2  UTF-8  LF  JSON  Go Live  Prettier  Quokka  5:08 p.m.  2/07/2023
```

## Instalando Cors



```
File Edit Selection View Go Run Terminal Help FruVer - Visual Studio Code

EXPLORER ...
FRUVER
  Controllers
    controller.js
  Database
    database.js
  Models
    productos.js
  node_modules
  Routes
    routes.js
  package-lock.json
  package.json
  server.js

database.js > OUTLINE > TIMELINE > VS CODE PETS

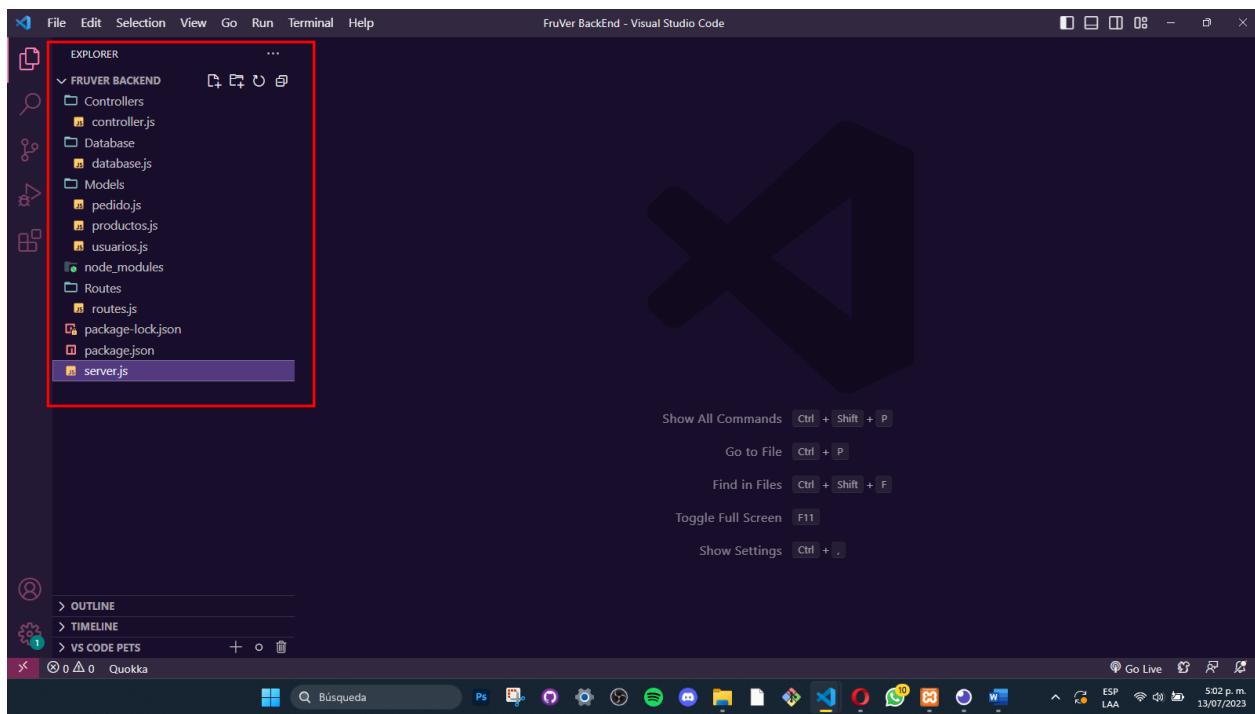
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
● PS D:\UDENAR\Diplomado Udenar\Desarrollo de software de última tecnología\Taller BackEnd\FruVer> npm i cors
added 2 packages, and audited 138 packages in 2s

12 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

Go Live  Prettier  Quokka  6:08 p.m.  2/07/2023
```

## Componentes BackEnd



## Estructura servidor

The screenshot shows the Visual Studio Code interface with the title bar "server.js - FruVer BackEnd - Visual Studio Code". The code editor displays the "server.js" file content:

```
1 import express from "express";
2 import { router } from "./Routes/routes.js";
3 import { sequelize } from "./Database/database.js";
4 import cors from "cors"; //IMPORTAR CORS PARA EL NAVEGADOR
5
6 const app = express();
7
8 // - CORS PARA BLOQUEO DE NAVEGADOR, CONTROL DE ACCESO
9 app.use(cors());
10
11 //Montar enrutador en app principal
12 app.use(express.json());
13 app.use(router);
14 app.set("port", 3000);
15
16 const testDB = async () => {
17   try {
18     await sequelize.sync();
19     console.log(`EXITO EN LA CONEXIÓN`);
20   } catch (error) {
21     console.log(`Error EN LA CONEXIÓN ${error}`);
22   }
23 };
24
25 testDB();
26
27 //Correr Servicio por puerto 3000
28 app.listen(app.get("port"), () => {
29   console.log(`Servidor Escuchando por puerto ${app.get("port")}`);
30 });
31
```

The code uses ES6 imports and async functions. The bottom status bar shows "Ln 1, Col 1" and the date/time: "5:07 p.m. 13/07/2023".

## Estructura rutas

The screenshot shows the Visual Studio Code interface with the file `routes.js` open. The code defines a router for a backend application, handling various routes for products, users, and orders.

```
File Edit Selection View Go Run Terminal Help
File Explorer routes.js
FRUVER BACKEND
  Controllers
    controller.js
  Database
    database.js
  Models
    pedido.js
    productos.js
    usuarios.js
  node_modules
  Routes
    routes.js
  package-lock.json
  package.json
  server.js

routes.js
Routes > routes.js > ...
1  import express from 'express';
2  const router = express.Router();
3
4  // Definir Rutas
5  router.get('/', (req, res) => {
6    res.send("Bienvenido a FruVerFresh");
7  });
8
9  // - RUTAS PARA PRODUCTOS
10 router.get('/productos', listarProductos);
11 router.get('/productos/:id_producto', getProducto);
12 router.post('/productos', registrarProductos);
13 router.put('/productos/:id_producto', actualizarProductos);
14 router.delete('/productos/:id_producto', eliminarProductos);
15 router.post('/buscar/', buscarProductos);
16
17 // - RUTAS PARA CLIENTES
18 router.get('/usuarios/:id_usuario', getUser);
19 router.post('/usuarios', registrarUsuarios);
20 router.post('/login', autenticarUsuarios);
21
22 // - RUTAS PEDIDO
23 router.get('/home-cliente/compras', listarPedidos);
24 router.post('/home-cliente/compras', registrarPedidos);
25 router.delete('/home-administrador/:id_pedido', eliminarPedidos);
26
27 export { router };
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

The code includes imports for `express` and `Router`, defines a `router`, and sets up various HTTP methods for different endpoints like listing products, getting a product by ID, and logging in users. It also includes sections for client and order management.

## Estructura modelo usuarios

The screenshot shows the Visual Studio Code interface with the file `usuarios.js` open. The code defines a Sequelize model for users, mapping to a `usuarios` table in the database.

```
File Edit Selection View Go Run Terminal Help
File Explorer usuarios.js
FRUVER BACKEND
  Controllers
    controller.js
  Database
    database.js
  Models
    pedido.js
    productos.js
    usuarios.js
  node_modules
  Routes
    routes.js
  package-lock.json
  package.json
  server.js

usuarios.js
Models > usuarios.js > Usuario > tipo_usuario
1 import { DataTypes } from 'sequelize';
2 import { sequelize } from '../Database/database.js';
3
4 const Usuario = sequelize.define(
5   'usuarios',
6   {
7     // Definicion de Atributos
8     id_usuario: {
9       type: DataTypes.INTEGER,
10      allowNull: false,
11      primaryKey: true,
12      autoIncrement: true,
13    },
14     nombre_usuario: [
15       type: DataTypes.STRING,
16       allowNull: false,
17     ],
18     apellido_usuario: [
19       type: DataTypes.STRING,
20       allowNull: false,
21     ],
22     edad_usuario: [
23       type: DataTypes.INTEGER,
24       allowNull: false,
25     ],
26     telefono_usuario: [
27       type: DataTypes.STRING,
28       allowNull: false,
29     ],
30     correo_usuario: [
31       type: DataTypes.STRING,
32       allowNull: false,
33     ],
34   }
35 );
36
37
38
39
40
41
42
```

The code uses Sequelize's `define` method to create a `Usuario` model with attributes `id_usuario`, `nombre_usuario`, `apellido_usuario`, `edad_usuario`, `telefono_usuario`, and `correo_usuario`. It specifies data types and allowNull settings for each attribute.

## Estructura productos

The screenshot shows the Visual Studio Code interface with the title bar "productos.js - FruVer BackEnd - Visual Studio Code". The Explorer sidebar on the left shows a project structure for "FRUVER BACKEND" with files like controller.js, database.js, Models (pedido.js, productos.js), usuarios.js, node\_modules, Routes (routes.js), package-lock.json, package.json, and server.js. The main editor area displays the code for "productos.js". The code defines a "Producto" model using Sequelize:

```
import { DataTypes } from "sequelize";
import { sequelize } from "../Database/database.js";

const Producto = sequelize.define(
  "producto",
  {
    // Definicion de Atributos
    id_producto: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    nombre_producto: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    precio_producto: {
      type: DataTypes.DECIMAL,
      allowNull: false,
    },
    foto_producto: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  },
  {
    timestamps: false,
  }
);
```

The status bar at the bottom shows "Ln 19, Col 30" and "5:09 p.m. 13/07/2023".

## Estructura pedidos

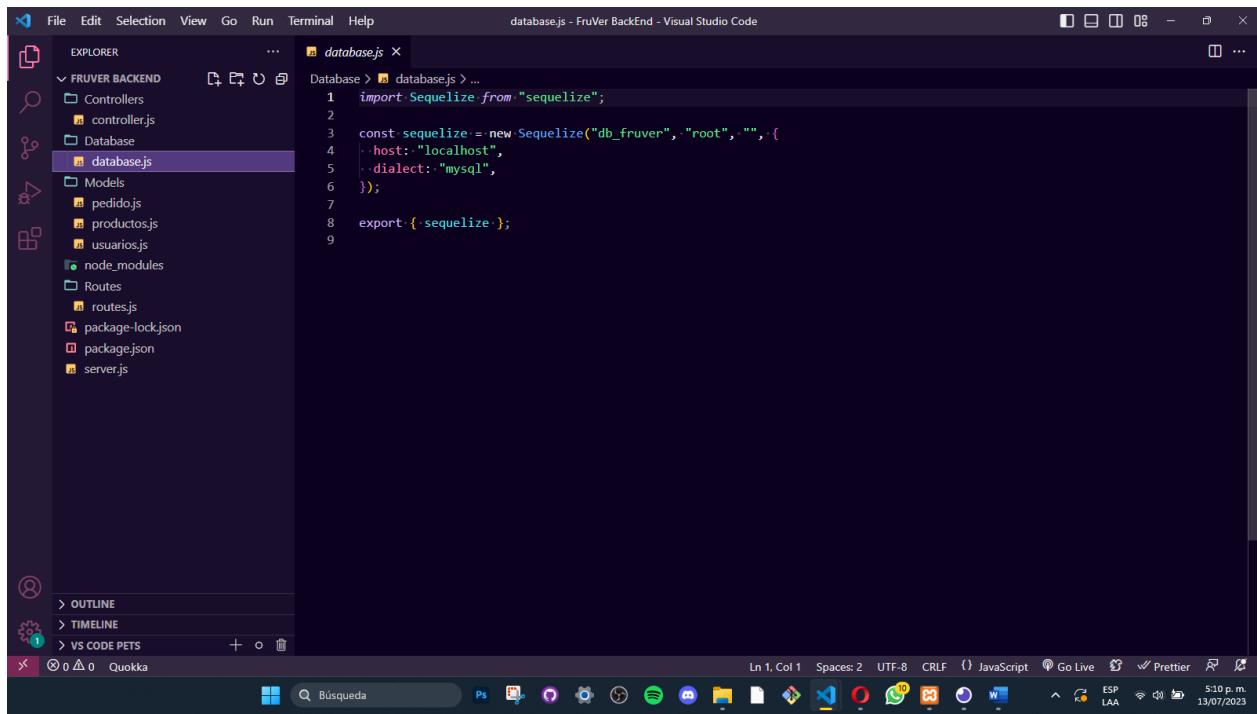
The screenshot shows the Visual Studio Code interface with the title bar "pedido.js - FruVer BackEnd - Visual Studio Code". The Explorer sidebar on the left shows a project structure for "FRUVER BACKEND" with files like controller.js, database.js, Models (pedido.js), usuarios.js, node\_modules, Routes (routes.js), package-lock.json, package.json, and server.js. The main editor area displays the code for "pedido.js". The code defines a "Pedido" model using Sequelize:

```
import { DataTypes } from "sequelize";
import { sequelize } from "../Database/database.js";

const Pedido = sequelize.define(
  "pedidos",
  {
    // Definicion de Atributos
    id_pedido: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    nombre_cliente_pedido: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    apellido_cliente_pedido: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    telefono_cliente_pedido: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    correo_cliente_pedido: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    direccion_cliente_pedido: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  }
);
```

The status bar at the bottom shows "Ln 25, Col 7" and "5:09 p.m. 13/07/2023".

## Estructura base de datos

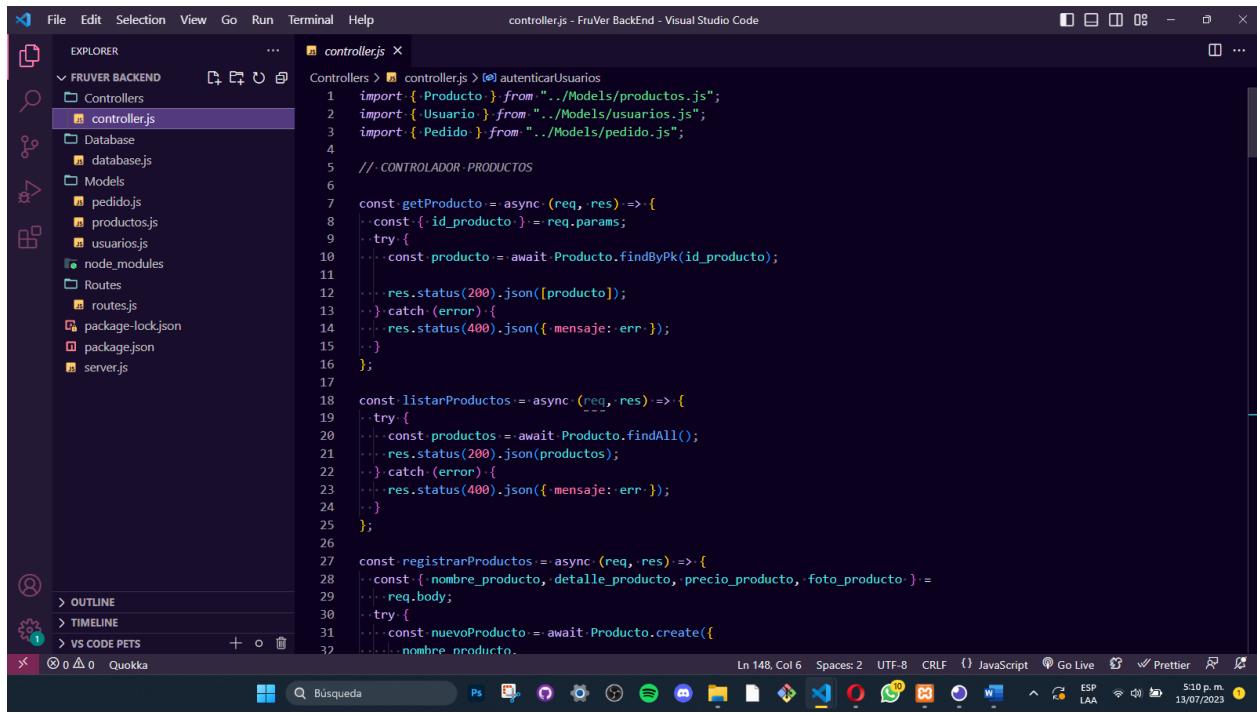


The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** database.js - Fruver BackEnd - Visual Studio Code
- Explorer:** Shows the project structure under "FRUVER BACKEND".
- Code Editor:** Displays the content of the database.js file.
- Bottom Bar:** Includes tabs for "Búsqueda" (Search), "Ps" (Performance), "P" (Profile), "O" (Output), "S" (Symbols), "C" (Coverage), "JavaScript", "Go Live", "Prettier", and date/time information (5:10 p.m., 13/07/2023).

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

## Estructura controlador



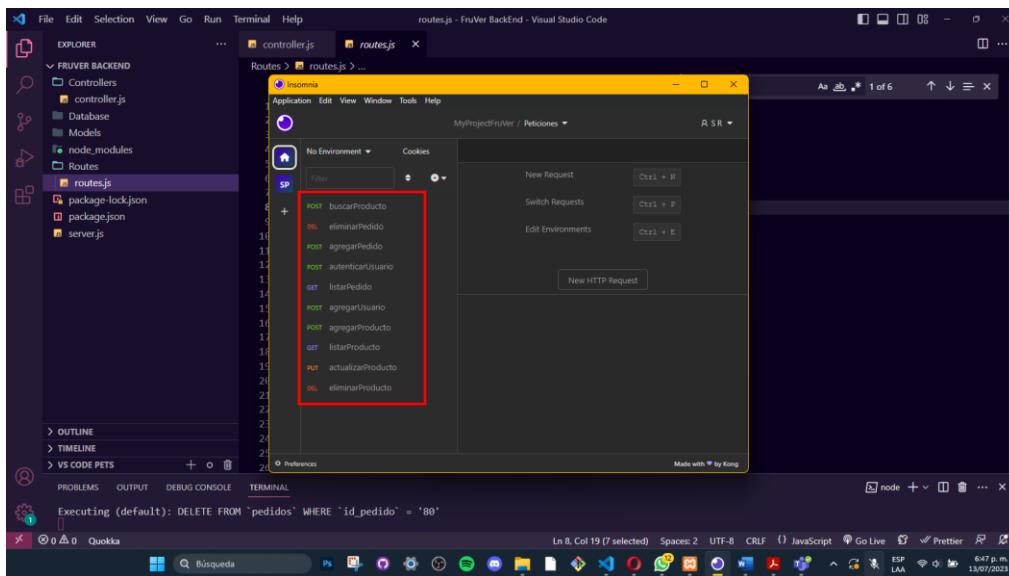
The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** controller.js - Fruver BackEnd - Visual Studio Code
- Explorer:** Shows the project structure under "FRUVER BACKEND".
- Code Editor:** Displays the content of the controller.js file.
- Bottom Bar:** Includes tabs for "Búsqueda" (Search), "Ps" (Performance), "P" (Profile), "O" (Output), "S" (Symbols), "C" (Coverage), "JavaScript", "Go Live", "Prettier", and date/time information (5:10 p.m., 13/07/2023).

```
1 import { Producto } from "../Models/productos.js";
2 import { Usuario } from "../Models/usuarios.js";
3 import { Pedido } from "../Models/pedido.js";
4
5 // CONTROLADOR PRODUCTOS
6
7 const getProducto = async (req, res) => {
8   const { id_producto } = req.params;
9   try {
10     const producto = await Producto.findByPk(id_producto);
11
12     res.status(200).json([producto]);
13   } catch (error) {
14     res.status(400).json({ mensaje: error });
15   }
16 };
17
18 const listarProductos = async (req, res) => {
19   try {
20     const productos = await Producto.findAll();
21     res.status(200).json(productos);
22   } catch (error) {
23     res.status(400).json({ mensaje: error });
24   }
25 };
26
27 const registrarProductos = async (req, res) => {
28   const { nombre_producto, detalle_producto, precio_producto, foto_producto } =
29     req.body;
30   try {
31     const nuevoProducto = await Producto.create({
32       nombre_producto,
```

**3. Realizar verificación de las diferentes operaciones a través de un cliente gráfico (Postman, Imnsomia, etc.), tomar capturas de pantalla que evidencien el resultado de las solicitudes realizadas.**

### Operaciones a realizar a través de Imnsomia



### Verificando operación agregar producto

```

const registrarProductos = async (req, res) => {
  const { nombre_producto, detalle_producto, precio_producto, foto_producto } =
    req.body;
  try {
    const nuevoProducto = await Producto.create({
      nombre_producto,
      detalle_producto,
      precio_producto,
      foto_producto,
    });
    res.status(200).json(nuevoProducto);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};

const actualizarProductos = async (req, res) => {
  const { id_producto } = req.params;
  const { nombre_producto, detalle_producto, precio_producto, foto_producto } =
    req.body;
  try {
    const productoAntiguo = await Producto.findById(id_producto);
    productoAntiguo.nombre_producto = nombre_producto;
    productoAntiguo.detalle_producto = detalle_producto;
    productoAntiguo.precio_producto = precio_producto;
    productoAntiguo.foto_producto = foto_producto;
    const modProducto = await productoAntiguo.save();
    res.json(modProducto);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
  
```

## Operación exitosa

The screenshot shows a developer's environment. On the left is the Visual Studio Code interface with an open file named 'controller.js'. The code contains a 'catch' block for an error. In the center is the Insomnia API client showing a successful POST request to 'http://localhost:3000/productos'. The JSON payload is:

```
1: {
2:   "id_producto": 95,
3:   "nombre_producto": "Papa",
4:   "detalle_producto": "Papa criolla seleccionada",
5:   "precio_producto": "15000",
6:   "foto_producto": "papa.jpg"
7: }
```

The response from Insomnia is a 200 OK status with a 49 ms response time. Below the Insomnia window is a terminal window showing MySQL queries and their execution results.

```
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'pedidos' AND TABLE_SCHEMA = 'db_fruver'
Executing (default): SHOW INDEX FROM `users` IN `db_fruver`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'users' AND TABLE_SCHEMA = 'db_fruver'
EXITO EN LA CONEXIÓN
Executing (default): INSERT INTO `productos` (`id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto`, `foto_producto`) VALUES (DEFAULT,?, ?, ?, ?)
$._store.books(*).author
```

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface connected to a MySQL database. The left sidebar shows the database structure with 'db\_fruver' selected. The main area displays the 'productos' table. A specific row for 'Papa' (id 95) is highlighted with a red border.

		id_producto	nombre_producto	detalle_producto	precio_producto	foto_producto
<input type="checkbox"/>	<a href="#">Editar</a>	84	Mango	El mango es una fruta tropical de origen asiático...	5000	assets/fotos/mango.png
<input type="checkbox"/>	<a href="#">Editar</a>	85	Piña	La piña madura tiene una fragancia muy singular. E...	35000	assets/fotos/piña.png
<input type="checkbox"/>	<a href="#">Editar</a>	86	Sandia	La sandia es un fruto grande y de forma más o meno...	15000	assets/fotos/sandia.png
<input type="checkbox"/>	<a href="#">Editar</a>	87	Naranja	La naranja es un fruto redondo, color naranja, con...	5000	assets/fotos/naranja.png
<input type="checkbox"/>	<a href="#">Editar</a>	88	Brócoli	El brócoli es también conocido por términos como b...	3000	assets/fotos/brócoli.png
<input type="checkbox"/>	<a href="#">Editar</a>	89	Mandarina	La mandarina es una fruta muy apreciada por su sab...	3800	assets/fotos/mandarina.png
<input type="checkbox"/>	<a href="#">Editar</a>	90	Zapallo	El zapallo es una hortaliza que se encuentra todas...	7000	assets/fotos/zapallo.png
<input type="checkbox"/>	<a href="#">Editar</a>	91	Melocotón	El melocotón es una típica drupa, un fruto carnos...	8000	assets/fotos/melocotón.png
<input type="checkbox"/>	<a href="#">Editar</a>	92	Pitahaya	En el aspecto nutricional, además de ser muy refe...	10000	assets/fotos/pitahaya.png
<input type="checkbox"/>	<a href="#">Editar</a>	93	Habichuela	Las habichuelas son una buena fuente de fibra, la ...	3500	assets/fotos/habichuela.png
<input type="checkbox"/>	<a href="#">Editar</a>	95	Papa	Papa criolla seleccionada	15000	papa.jpg

## Verificando operación listar productos

```
controller.js - FruVer BackEnd - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER
FRUVER BACKEND
  Controllers
    controller.js
  Database
  Models
  node_modules
  Routes
  package-lock.json
  package.json
  server.js

Controllers > controller.js > actualizarProductos
9  .try {
10   const producto = await Producto.findByPk(id_producto);
11
12   res.status(200).json([producto]);
13 } catch (error) {
14   res.status(400).json({ mensaje: error });
15 }
16
17
18 const listarProductos = async (req, res) => {
19   try {
20     const productos = await Producto.findAll();
21     res.status(200).json(productos);
22   } catch (error) {
23     res.status(400).json({ mensaje: error });
24   }
25 }
26
27 const registrarProductos = async (req, res) => {
28   const { nombre_producto, detalle_producto, precio_producto, foto_producto } = req.body;
29   try {
30     const nuevoProducto = await Producto.create({
31       nombre_producto,
32       detalle_producto,
33       precio_producto,
34       foto_producto
35     });
36     res.status(201).json(nuevoProducto);
37   } catch (error) {
38     res.status(400).json({ mensaje: error });
39   }
40 }

Executing (default): SHOW INDEX FROM `usuarios`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'pedidos' AND TABLE_SCHEMA = 'db_fruver'
EXITO EN LA CONEXIÓN
Executing (default): INSERT INTO `productos` (`id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto`, `foto_producto`) VALUES (DEFAULT,?, ?, ?, ?);
Executing (default): SELECT `id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto` FROM `productos` AS `producto`;
Executing (default): SELECT `id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto` FROM `productos` AS `producto`;

Ln 47, Col 8 Spaces: 2 UTF-8 CRLF () JavaScript ⚡ Go Live ⚡ Prettier ⚡ node + ⚡ 5:29 p.m. 13/07/2023
```

## Operación exitosa

```
controller.js - FruVer BackEnd - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER
FRUVER BACKEND
  Controllers
    controller.js
  Database
  Models
  node_modules
  Routes
  package-lock.json
  package.json
  server.js

Controllers > controller.js > actualizarProductos
.try {
const producto = await Producto.findByPk(id_producto);

res.status(200).json([producto]);
} catch (error) {
res.status(400).json({ mensaje: error });
}

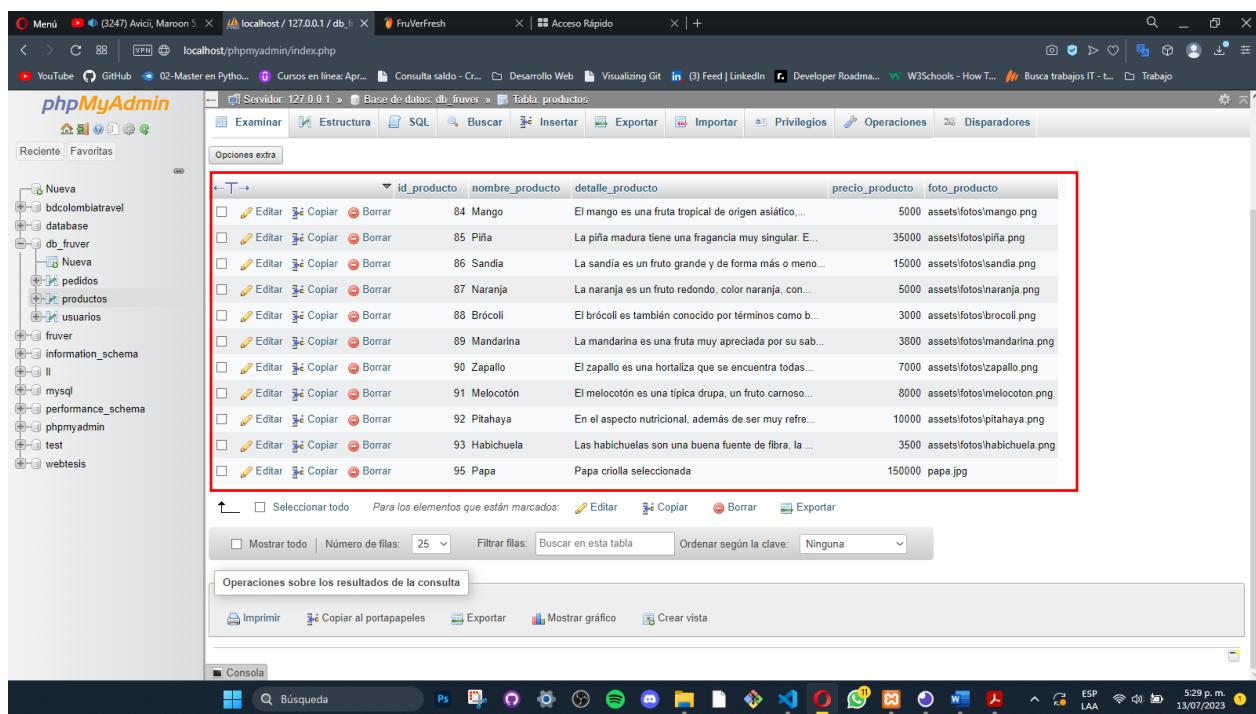
const listarProductos = async (req, res) => {
try {
const productos = await Producto.findAll();
res.status(200).json(productos);
} catch (error) {
res.status(400).json({ mensaje: error });
}
};

const registrarProductos = async (req, res) => {
const { nombre_producto, detalle_producto, precio_producto, foto_producto } = req.body;
try {
const nuevoProducto = await Producto.create({
nombre_producto,
detalle_producto,
precio_producto,
foto_producto
});
res.status(201).json(nuevoProducto);
} catch (error) {
res.status(400).json({ mensaje: error });
}
};

Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'pedidos' AND TABLE_SCHEMA = 'db_fruver'
EXITO EN LA CONEXIÓN
Executing (default): INSERT INTO `productos` (`id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto`, `foto_producto`) VALUES (DEFAULT,?, ?, ?, ?);
Executing (default): SELECT `id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto` FROM `productos` AS `producto`;
Executing (default): SELECT `id_producto`, `nombre_producto`, `detalle_producto`, `precio_producto` FROM `productos` AS `producto`;

Ln 47, Col 8 Spaces: 2 UTF-8 CRLF () JavaScript ⚡ Go Live ⚡ Prettier ⚡ node + ⚡ 5:29 p.m. 13/07/2023
```

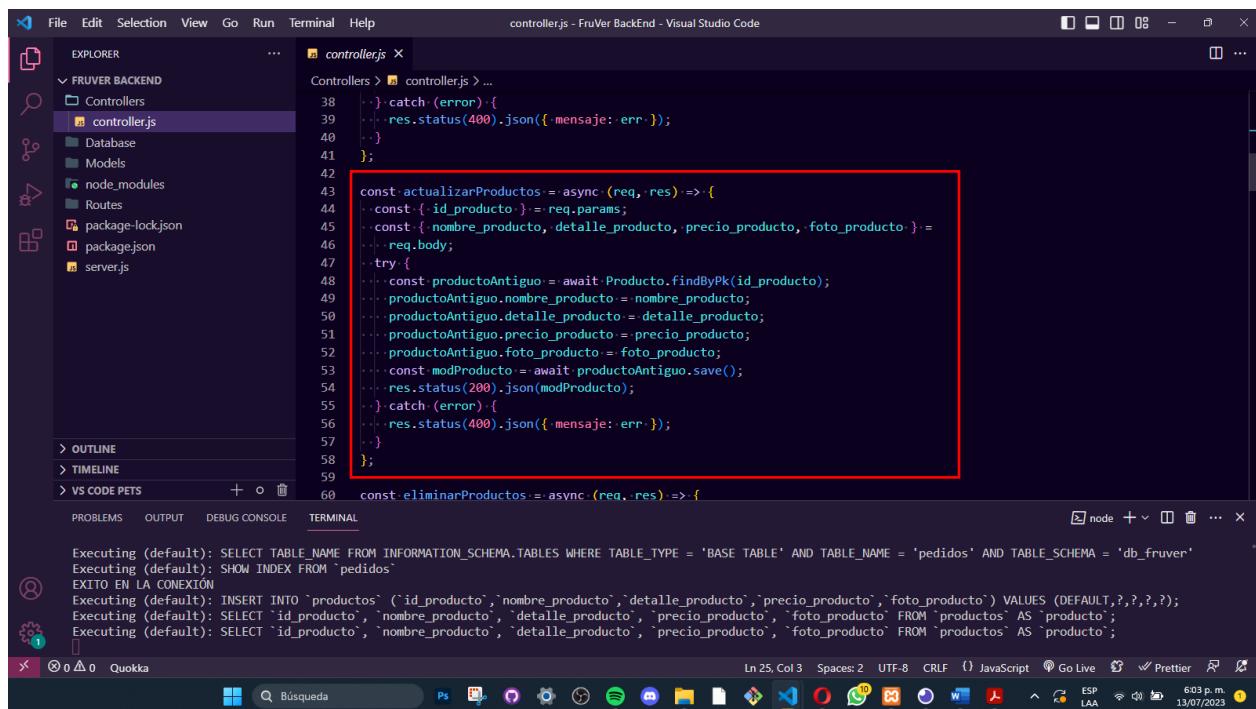
## Comprobando en la BD



The screenshot shows the phpMyAdmin interface connected to the 'db\_fruver' database. The 'products' table is selected, displaying 15 rows of fruit data. The columns are: id\_producto, nombre\_producto, detalle\_producto, precio\_producto, and foto\_producto. A red box highlights the data grid. Below the grid, there are buttons for selecting all rows, editing, copying, deleting, and exporting.

	id_producto	nombre_producto	detalle_producto	precio_producto	foto_producto
1	84	Mango	El mango es una fruta tropical de origen asiático,...	5000	assets/fotos/mango.png
2	85	Piña	La piña madura tiene una fragancia muy singular. E...	35000	assets/fotos/piña.png
3	86	Sandía	La sandía es un fruto grande y de forma más o meno...	15000	assets/fotos/sandia.png
4	87	Naranja	La naranja es un fruto redondo, color naranja, con...	5000	assets/fotos/naranja.png
5	88	Brócoli	El brócoli es también conocido por términos como b...	3000	assets/fotos/brócoli.png
6	89	Mandarina	La mandarina es una fruta muy apreciada por su sab...	3800	assets/fotos/mandarina.png
7	90	Zapallo	El zapallo es una hortaliza que se encuentra todas...	7000	assets/fotos/zapallo.png
8	91	Melocotón	El melocotón es una típica drupa, un fruto carnos...	8000	assets/fotos/melocoton.png
9	92	Pitahaya	En el aspecto nutricional, además de ser muy refre...	10000	assets/fotos/pitahaya.png
10	93	Habichuela	Las habichuelas son una buena fuente de fibra, la ...	3500	assets/fotos/habichuela.png
11	95	Papa	Papa criolla seleccionada	150000	papa.jpg

## Verificando operación actualizar producto



The screenshot shows the Visual Studio Code editor with the 'controller.js' file open. The 'controller.js' file contains code for a Node.js application. A red box highlights the 'actualizarProductos' function, which is an asynchronous function that finds a product by ID, creates a new object with updated values, and saves it back to the database. The function returns a status 200 if successful or 400 if there's an error.

```
const actualizarProductos = async (req, res) => {
  const { id_producto } = req.params;
  const { nombre_producto, detalle_producto, precio_producto, foto_producto } =
    req.body;
  try {
    const productoAntiguo = await Producto.findByPk(id_producto);
    productoAntiguo.nombre_producto = nombre_producto;
    productoAntiguo.detalle_producto = detalle_producto;
    productoAntiguo.precio_producto = precio_producto;
    productoAntiguo.foto_producto = foto_producto;
    const modProducto = await productoAntiguo.save();
    res.status(200).json(modProducto);
  } catch (error) {
    res.status(400).json({ mensaje: err });
  }
};
```

## Operación exitosa

The screenshot shows the Visual Studio Code interface with the controller.js file open in the editor. The Insomnia application window is overlaid on the code editor, displaying a POST request to update a product. The request body contains the following JSON:

```
1  {
2     "id_producto": 95,
3     "nombre_producto": "Papa criolla",
4     "detalle_producto": "Papa criolla seleccionada",
5     "precio_producto": "150000",
6     "foto_producto": "papa.jpg"
7 }
```

The response from Insomnia is a 200 OK status with a timestamp of Just Now. Below the response, the Headers tab shows the updated product details.

VS Code status bar: Executing (default): UPDATE `productos` SET `nombre\_producto`=? WHERE `id\_producto` = ?

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface connected to the db\_fruver database. The products table is selected, displaying a list of fruit products. One row, specifically the one for 'Papa criolla' (id 95), is highlighted with a red border. The table structure includes columns for id\_producto, nombre\_producto, detalle\_producto, precio\_producto, and foto\_producto.

	id_producto	nombre_producto	detalle_producto	precio_producto	foto_producto
<input type="checkbox"/>	84	Mango	El mango es una fruta tropical de origen asiático....	5000	assets/fotos/mango.png
<input type="checkbox"/>	85	Piña	La piña madura tiene una fragancia muy singular. E...	35000	assets/fotos/piña.png
<input type="checkbox"/>	86	Sandia	La sandía es un fruto grande y de forma más o meno...	15000	assets/fotos/sandia.png
<input type="checkbox"/>	87	Naranja	La naranja es un fruto redondo, color naranja, con...	5000	assets/fotos/naranja.png
<input type="checkbox"/>	88	Brócoli	El brócoli es también conocido por términos como b...	3000	assets/fotos/brócoli.png
<input type="checkbox"/>	89	Mandarina	La mandarina es una fruta muy apreciada por su sab...	3800	assets/fotos/mandarina.png
<input type="checkbox"/>	90	Zapallo	El zapallo es una hortaliza que se encuentra todas...	7000	assets/fotos/zapallo.png
<input type="checkbox"/>	91	Melocotón	El melocotón es una típica drupa, un fruto carnos...	8000	assets/fotos/melocotón.png
<input type="checkbox"/>	92	Pitahaya	En el aspecto nutricional, además de ser muy refre...	10000	assets/fotos/pitahaya.png
<input type="checkbox"/>	93	Habichuela	Las habichuelas son una buena fuente de fibra, la ...	3500	assets/fotos/habichuela.png
<input checked="" type="checkbox"/>	95	Papa criolla	Papa criolla seleccionada	150000	papa.jpg

## Verificando operación eliminar producto



```
controller.js

Controllers > controller.js > ...
54     .res.status(200).json(modProducto);
55   } catch (error) {
56     .res.status(400).json({ mensaje: error });
57   }
58 }

59 const eliminarProductos = async (req, res) => {
60   const { id_producto } = req.params;
61   try {
62     const respuesta = await Producto.destroy({
63       where: {
64         id_producto,
65       },
66     });
67     res.status(200).json({ mensaje: "Registro Eliminado" });
68   } catch (error) {
69     res.status(400).json({ mensaje: "Registro No Eliminado" + error });
70   }
71 }
72 }

73 const buscarProductos = async (req, res) => {
74   const { nombre_producto } = req.body;
75 }

76
```

## **Operación exitosa**

A screenshot of a Microsoft Word document titled "Informe BackEnd DruVerFresh.docx - Word". The Word ribbon is visible at the top. An Insomnia REST client window is overlaid on the Word interface. The Insomnia window shows a list of API endpoints for a "MyProjectFruVer" service, including methods like GET, POST, PUT, and DELETE. One endpoint, "eliminarProducto", is selected and its response is displayed as a JSON object: { "mensaje": "Registro Eliminado" }. The status bar at the bottom of the screen shows "Página 20 de 21" and "304 palabras".

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface connected to the 'db\_fruver' database. The left sidebar lists databases like 'bdcolombiatravel', 'database', and 'db\_fruver'. The 'db\_fruver' database has tables 'pedidos', 'productos', and 'usuarios'. The main area displays the 'productos' table with 10 rows of fruit data. The columns are: id\_producto, nombre\_producto, detalle\_producto, precio\_producto, and foto\_producto. A red box highlights the first few rows of the table.

	id_producto	nombre_producto	detalle_producto	precio_producto	foto_producto
1	84	Mango	El mango es una fruta tropical de origen asiático...	5000	assets/fotos/mango.png
2	85	Piña	La piña madura tiene una fragancia muy singular. E...	35000	assets/fotos/piña.png
3	86	Sandía	La sandía es un fruto grande y de forma más o meno...	15000	assets/fotos/sandia.png
4	87	Naranja	La naranja es un fruto redondo, color naranja, con...	5000	assets/fotos/naranja.png
5	88	Brócoli	El brócoli es también conocido por términos como b...	3000	assets/fotos/brócoli.png
6	89	Mandarina	La mandarina es una fruta muy apreciada por su sab...	3800	assets/fotos/mandarina.png
7	90	Zapallo	El zapallo es una hortaliza que se encuentra todas...	7000	assets/fotos/zapallo.png
8	91	Melocotón	El melocotón es una típica drupa, un fruto carnos...	8000	assets/fotos/melocoton.png
9	92	Pitahaya	En el aspecto nutricional, además de ser muy refre...	10000	assets/fotos/pitahaya.png
10	93	Habichuela	Las habichuelas son una buena fuente de fibra, la ...	3500	assets/fotos/habichuela.png

## Verificando operación buscar producto

The screenshot shows the Visual Studio Code interface with the file 'controller.js' open. The code defines a function 'buscarProductos' that searches for a product by name. A red box highlights the search logic. The terminal at the bottom shows database queries being executed.

```
const buscarProductos = async (req, res) => {
  const { nombre_producto } = req.body;

  try {
    const producto = await Producto.findOne({
      where: {
        nombre_producto: nombre_producto,
      },
    });

    if (!producto) {
      return res.status(401).json({ mensaje: "Producto no encontrado" });
    }

    if (producto.nombre_producto !== nombre_producto) {
      return res.status(401).json({ mensaje: "Producto no encontrado" });
    }

    res.status(200).json(producto);
  } catch (error) {
    res.status(400).json({ mensaje: error.message });
  }
};
```

Executing (default): SHOW INDEX FROM `pedidos`  
EXITO EN LA CONEXIÓN  
Executing (default): SELECT `id\_producto`, `nombre\_producto`, `detalle\_producto`, `precio\_producto`, `foto\_producto` FROM `productos` AS `producto` WHERE `producto`.`nombre\_producto` = 'papa' LIMIT 1;  
Executing (default): SELECT `id\_producto`, `nombre\_producto`, `detalle\_producto`, `precio\_producto`, `foto\_producto` FROM `productos` AS `producto` WHERE `producto`.`nombre\_producto` = 'Piña' LIMIT 1;

## Operación exitosa

The screenshot shows the Visual Studio Code interface for the 'FRUVER BACKEND' project. The 'controller.js' file is open in the editor. In the bottom right corner of the editor, there is a status bar indicating 'Made with ❤ by Kong'. An Insomnia REST client window is overlaid on the code editor, showing a successful POST request to 'http://localhost:3000/buscar' with JSON data containing 'nombre\_producto': 'Piña'. The response is a 200 OK status with a duration of 6.11 ms and a size of 589 B. The response body contains detailed information about the Piña fruit.

The screenshot shows the Visual Studio Code interface for the 'FRUVER FRONTEND' project. The 'main.ts' file is open in the editor. In the bottom right corner of the editor, there is a status bar indicating 'Made with ❤ by Kong'. An Insomnia REST client window is overlaid on the code editor, showing an unauthorized POST request to 'http://localhost:3000/buscar' with JSON data containing 'nombre\_producto': 'Papa'. The response is a 401 Unauthorized status with a duration of 7.26 ms and a size of 36 B. The response body indicates that the product was not found.

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface connected to a database named 'db\_fruver'. The left sidebar lists databases like 'bdcolombiatravel', 'database', 'db\_fruver', and 'fruver'. The main area displays a table titled 'productos' with the following data:

	id_producto	nombre_producto	detalle_producto	precio_producto	foto_producto
<input type="checkbox"/>	84	Mango	El mango es una fruta tropical de origen asiático...	5000	assets/fotos/mango.png
<input checked="" type="checkbox"/>	85	Piña	La piña madura tiene una fragancia muy singular. E...	35000	assets/fotos/piña.png
<input type="checkbox"/>	86	Sandía	La sandía es un fruto grande y de forma más o meno...	15000	assets/fotos/sandia.png
<input type="checkbox"/>	87	Naranja	La naranja es un fruto redondo, color naranja, con...	5000	assets/fotos/naranja.png
<input type="checkbox"/>	88	Brócoli	El brócoli es también conocido por términos como b...	3000	assets/fotos/brócoli.png
<input type="checkbox"/>	89	Mandarina	La mandarina es una fruta muy apreciada por su sab...	3800	assets/fotos/mandarina.png
<input type="checkbox"/>	90	Zapallo	El zapallo es una hortaliza que se encuentra todas...	7000	assets/fotos/zapallo.png
<input type="checkbox"/>	91	Melocotón	El melocotón es una típica drupa, un fruto carnos...	8000	assets/fotos/melocoton.png
<input type="checkbox"/>	92	Pitahaya	En el aspecto nutricional, además de ser muy refre...	10000	assets/fotos/pitahaya.png
<input type="checkbox"/>	93	Habichuela	Las habichuelas son una buena fuente de fibra, la ...	3500	assets/fotos/habichuela.png

## Verificando operación registrar usuario

The screenshot shows the Visual Studio Code editor with the file 'controller.js' open. The code defines a function 'registerUsuarios' that creates a new user in a database:

```
const registerUsuarios = async (req, res) => {
  const {
    nombre_usuario,
    apellido_usuario,
    edad_usuario,
    telefono_usuario,
    correo_usuario,
    contraseña_usuario,
    tipo_usuario,
  } = req.body;
  try {
    const nuevoUsuario = await Usuario.create({
      nombre_usuario,
      apellido_usuario,
      edad_usuario,
      telefono_usuario,
      correo_usuario,
      contraseña_usuario,
      tipo_usuario,
    });
    res.status(200).json(nuevoUsuario);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

The code is highlighted with a red rectangle. The terminal at the bottom shows the command 'Executing (default): SELECT `id\_producto`, `nombre\_producto`, `detalle\_producto`, `precio\_producto`, `foto\_producto` FROM `productos` AS `producto` WHERE `producto`.`nombre\_producto` = 'Papa' LIMIT 1;

## Operación exitosa

The screenshot shows a Visual Studio Code interface with an Insomnia REST client window. The Insomnia window displays a successful POST request to `http://localhost:3000/usuarios`. The JSON payload sent was:

```
1. {
2.   "id_usuario": 1,
3.   "nombre_usuario": "Juanito",
4.   "apellido_usuario": "Suarez",
5.   "edad_usuario": 35,
6.   "telefono_usuario": "3111117896",
7.   "correo_usuario": "juanito@gmail.com",
8.   "contrasena_usuario": "12345",
9.   "tipo_usuario": "2"
10. }
```

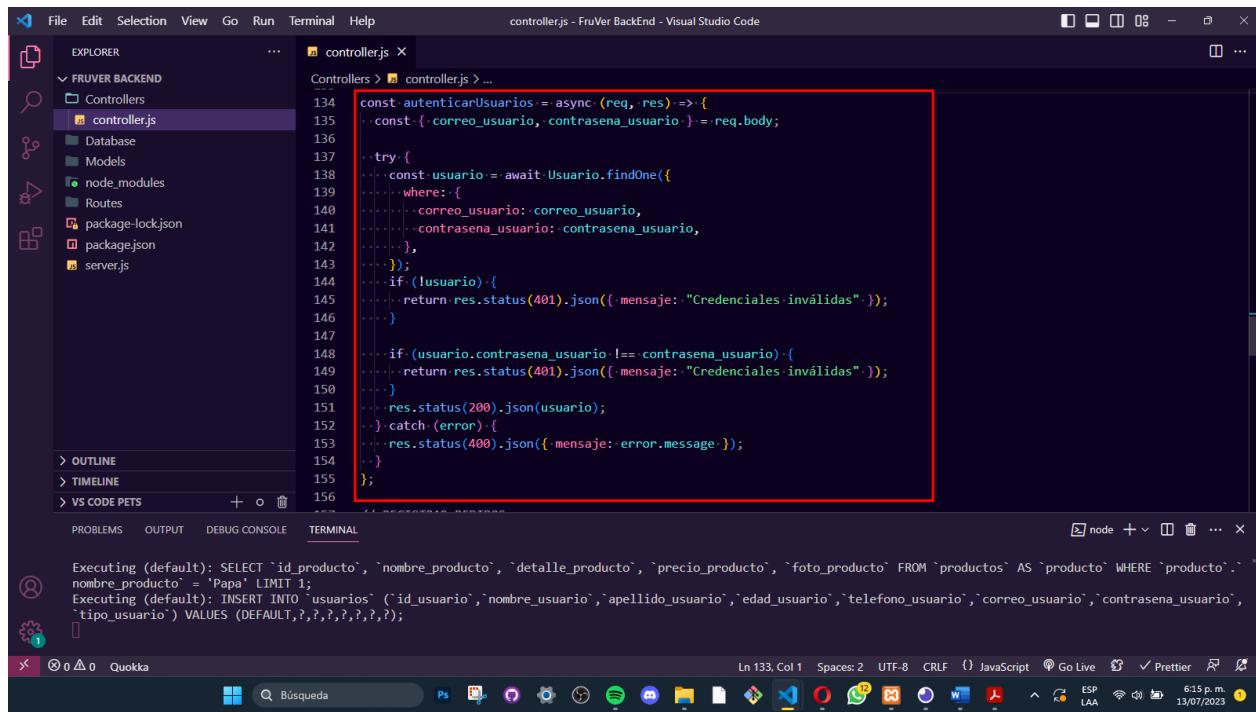
The response from the server was a `200 OK` status with a response time of 15.8 ms and a size of 207 B. The timeline shows the request was made just now.

## Comprobando en la BD

The screenshot shows the `usuarios` table in the `db_fruver` database via phpMyAdmin. The table has 8 rows of data, each representing a user. The last row, which corresponds to the newly created user 'Juanito', is highlighted with a red border.

	id_usuario	nombre_usuario	apellido_usuario	edad_usuario	telefono_usuario	correo_usuario	contrasena_usuario	tipo_usuario
1	1	Administrador	FruVerFresh	21	7299225	adminfruverfresh@admin.com	adminfruverfresh11	1
2	2	Luis	Suarez Páez	35	3111117896	paez@gmail.com	12345	2
3	3	Alejandra	Valencia Mora	22	3214374523	alejandro@gmail.com	12345	2
4	4	Armando	Peralta	35	3214396352	armando@gmail.com	12345	2
5	5	Fredy	Gonzales	20	3178963652	freddy@gmail.com	12345	2
6	6	Clemente	Reyes	47	3189632545	clemente@gmail.com	12345	2
7	10	Paula	Jaramillo	36	3205902356	paula@gmail.com	paula123	2
8	11	Juanito	Suarez	35	3111117896	juanito@gmail.com	12345	2

## Verificando operación autenticar usuario



```
const autenticarUsuarios = async (req, res) => {
  const { correo_usuario, contrasena_usuario } = req.body;

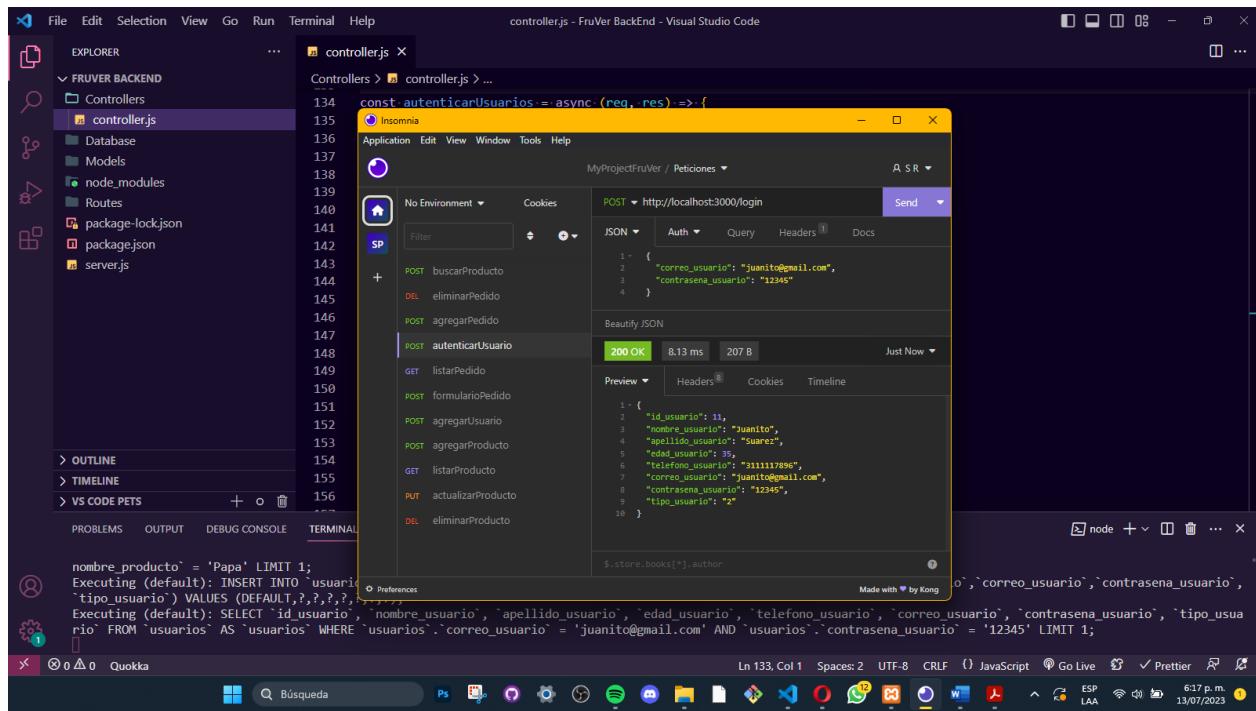
  try {
    const usuario = await Usuario.findOne({
      where: {
        correo_usuario: correo_usuario,
        contrasena_usuario: contrasena_usuario,
      },
    });

    if (!usuario) {
      return res.status(401).json({ mensaje: "Credenciales inválidas" });
    }

    if (usuario.contrasena_usuario !== contrasena_usuario) {
      return res.status(401).json({ mensaje: "Credenciales inválidas" });
    }

    res.status(200).json(usuario);
  } catch (error) {
    res.status(400).json({ mensaje: error.message });
  }
};
```

## Operación exitosa



The screenshot shows the Insomnia REST client integrated into the Visual Studio Code interface. A POST request is being made to `http://localhost:3000/login` with the following JSON body:

```
{
  "correo_usuario": "juanito@gmail.com",
  "contrasena_usuario": "12345"
}
```

The response is a 200 OK status with a response time of 8.13 ms and a size of 207 B. The response body is a JSON object containing user information:

```
{
  "id_usuario": 11,
  "correo_usuario": "Juanito",
  "apellido_usuario": "Suarez",
  "edad_usuario": 35,
  "telefono_usuario": "311117896",
  "correo_usuario": "juanito@gmail.com",
  "contrasena_usuario": "12345",
  "tipo_usuario": "2"
}
```

The screenshot shows a Visual Studio Code interface with an Insomnia REST client window overlaid. The Insomnia window displays a POST request to `http://localhost:3000/login` with the following JSON body:

```

1: {
2:   "correo_usuario": "juanito@gmail.com",
3:   "contrasena_usuario": "12345"
4: }

```

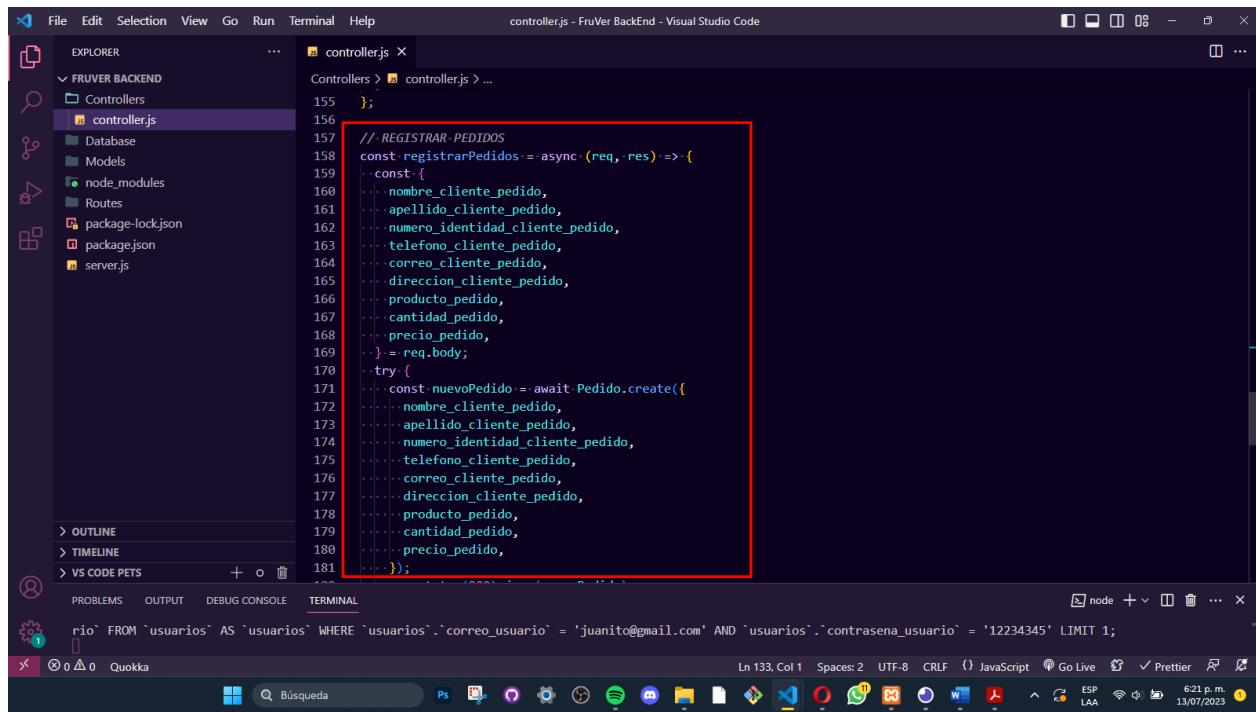
The response is a **401 Unauthorized** status with a duration of 6.57 ms and a size of 37 B, received just now. The response body contains the message: `"mensaje": "Credenciales inválidas"`.

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface connected to the `localhost/127.0.0.1/db_fruver` database. The left sidebar shows databases like `bdcolumbatravel`, `database`, `db_fruver` (selected), `fruver`, `information_schema`, `mysql`, `performance_schema`, `phpmyadmin`, `test`, and `webtis`. The main area shows the `usuarios` table with the following data:

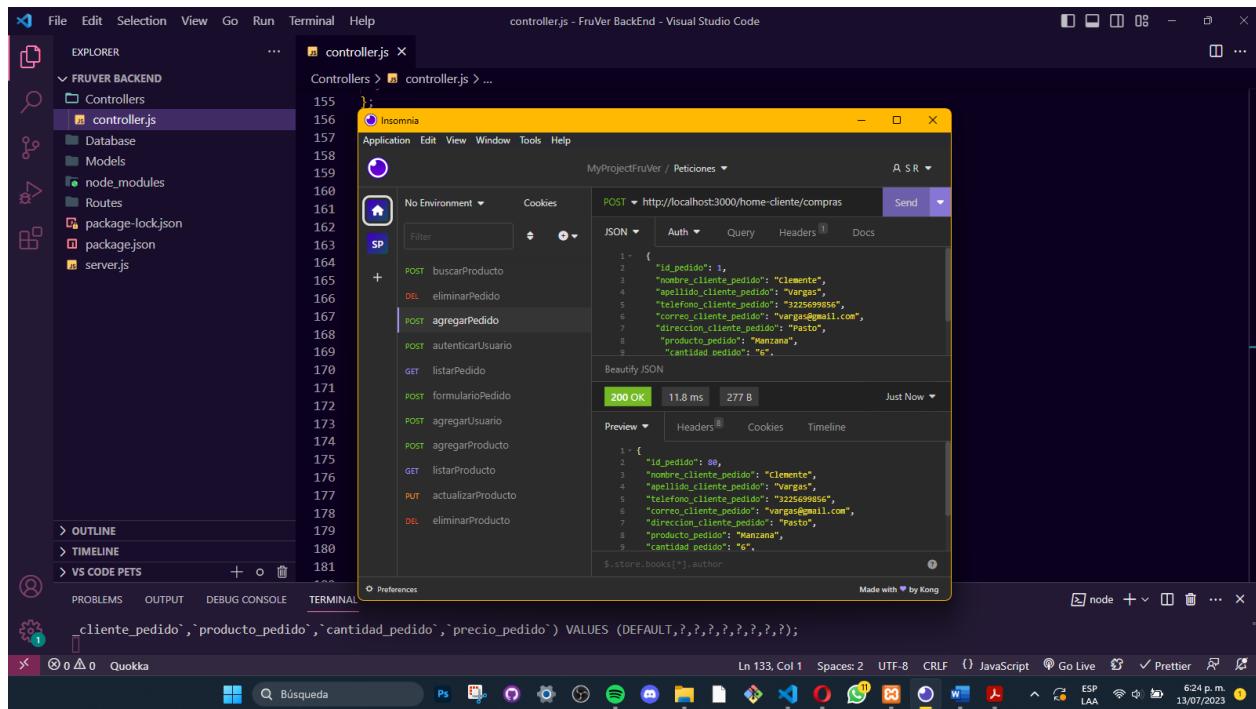
	<code>id_usuario</code>	<code>nombre_usuario</code>	<code>apellido_usuario</code>	<code>edad_usuario</code>	<code>telefono_usuario</code>	<code>correo_usuario</code>	<code>contrasena_usuario</code>	<code>tipo_usuario</code>
<input type="checkbox"/>	1	Administrador	FruVerFresh	21	7299225	adminfruverfresh@admin.com	adminfruverfresh11	1
<input type="checkbox"/>	2	Luis	Suarez Páez	35	3111117896	paez@gmail.com	12345	2
<input type="checkbox"/>	3	Alejandra	Valencia Mora	22	3214374523	alejandra@gmail.com	12345	2
<input type="checkbox"/>	4	Armando	Peralta	35	3214896352	armando@gmail.com	12345	2
<input type="checkbox"/>	5	Fredy	Gonzales	20	3178963652	fredy@gmail.com	12345	2
<input type="checkbox"/>	6	Clemente	Reyes	47	3189632545	clemente@gmail.com	12345	2
<input type="checkbox"/>	10	Paula	Jaramillo	36	3205902356	paula@gmail.com	paula123	2
<input type="checkbox"/>	11	Juanito	Suarez	35	3111117896	juanito@gmail.com	12345	2

## Verificando operación registrar pedido



```
//- REGISTRAR_PEDIDOS
const registrarPedidos = async (req, res) => {
  const {
    nombre_cliente_pedido,
    apellido_cliente_pedido,
    numero_identidad_cliente_pedido,
    telefono_cliente_pedido,
    correo_cliente_pedido,
    direccion_cliente_pedido,
    producto_pedido,
    cantidad_pedido,
    precio_pedido,
  } = req.body;
  try {
    const nuevoPedido = await Pedido.create({
      nombre_cliente_pedido,
      apellido_cliente_pedido,
      numero_identidad_cliente_pedido,
      telefono_cliente_pedido,
      correo_cliente_pedido,
      direccion_cliente_pedido,
      producto_pedido,
      cantidad_pedido,
      precio_pedido,
    });
  
```

## Operación exitosa



The screenshot shows the Insomnia REST client integrated into Visual Studio Code. A POST request is being made to `http://localhost:3000/home-cliente/compras`. The request body is a JSON object representing a new purchase:

```
{
  "id_pedido": 1,
  "nombre_cliente_pedido": "Clemente",
  "apellido_cliente_pedido": "Vargas",
  "numero_identidad_cliente_pedido": "3225699856",
  "telefono_cliente_pedido": "juanito@gmail.com",
  "direccion_cliente_pedido": "Pasto",
  "producto_pedido": "Manzana",
  "cantidad_pedido": "6"
}
```

The response is a 200 OK status with a response time of 11.8 ms and a size of 277 B. The response body is identical to the request body.

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface connected to the 'db\_fruver' database. The 'pedidos' table is selected, displaying three rows of data:

	id_pedido	nombre_cliente_pedido	apellido_cliente_pedido	telefono_cliente_pedido	correo_cliente_pedido	direccion_cliente_pedido	producto_pedido	cantidad_pedido	pre
1	78	Benito	Reyes	23456	benit@gmail.com	Pasto	Piña	1	
2	79	Puala	Jaramillo	3205902369	paula@gmail.com	Pasto	Naranja	6	
3	80	Clemente	Vargas	3226698056	vargas@gmail.com	Pasto	Manzana	6	

## Verificando operación listar pedidos

The screenshot shows the Visual Studio Code editor with the 'controller.js' file open. The 'listarPedidos' function is highlighted with a red box:

```
const listarPedidos = async (req, res) => {
  try {
    const pedido = await Pedido.findAll();
    res.status(200).json(pedido);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

## Operación exitosa

The screenshot shows a Visual Studio Code interface with several windows open. On the left, the Explorer sidebar shows a project structure for 'FRUVER BACKEND' with files like 'controller.js', 'Database', 'Models', 'Routes', 'node\_modules', 'package-lock.json', 'package.json', and 'server.js'. The 'controller.js' file is open in the main editor area, showing code for handling requests to '/home-cliente/compras'. A terminal window at the bottom shows a successful HTTP request to 'localhost:3000/home-cliente/compras' with a status of 200 OK. The Insomnia API client window shows a list of endpoints and a detailed response for the 'listarPedido' endpoint, which returns a JSON object with fields like 'id\_pedido', 'nombre\_cliente\_pedido', 'apellido\_cliente\_pedido', etc. The response body contains three rows of data.

```
res.status(400).json({mensaje: err});
```

```
200 OK 9.22 ms 816 B Just Now
```

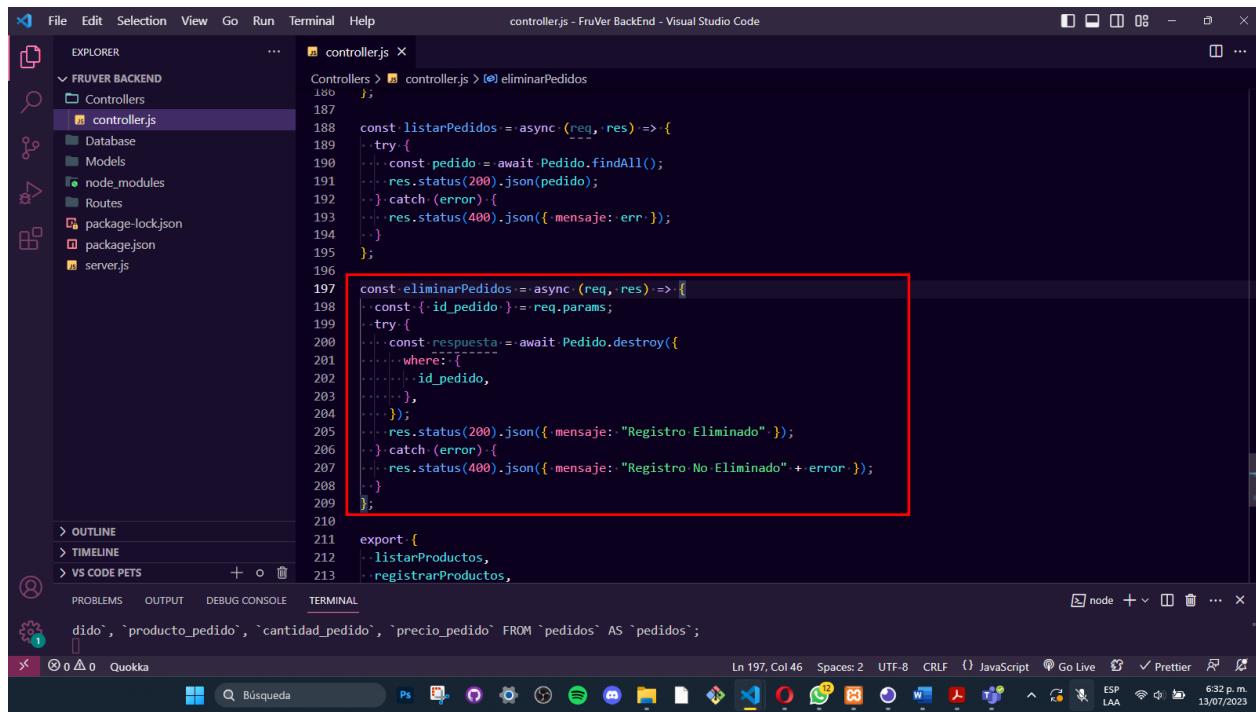
```
[{"id_pedido": 80, "nombre_cliente_pedido": "Clemente", "apellido_cliente_pedido": "Vargas", "telefono_cliente_pedido": "322569856", "correo_cliente_pedido": "vargas@gmail.com", "direccion_cliente_pedido": "Pasto", "producto_pedido": "Manzana", "cantidad_pedido": 6, "precio_pedido": "1500"}, {"id_pedido": 81, "nombre_cliente_pedido": "Benito", "apellido_cliente_pedido": "Reyes", "telefono_cliente_pedido": "23456", "correo_cliente_pedido": "benit@gmail.com", "direccion_cliente_pedido": "Pasto", "producto_pedido": "Piña", "cantidad_pedido": 1, "precio_pedido": "1000"}, {"id_pedido": 82, "nombre_cliente_pedido": "Paula", "apellido_cliente_pedido": "Jaramillo", "telefono_cliente_pedido": "3205902369", "correo_cliente_pedido": "paula@gmail.com", "direccion_cliente_pedido": "Pasto", "producto_pedido": "Naranja", "cantidad_pedido": 6, "precio_pedido": "5000"}]
```

## Comprobando en la BD

The screenshot shows a browser window for phpMyAdmin connected to 'localhost:127.0.0.1/db\_fruver'. The left sidebar shows databases like 'bdcolombiatravel', 'db\_fruver', 'fruver', and 'information\_schema'. The 'db\_fruver' database is selected, and the 'pedidos' table is currently viewed. The table has columns: id\_pedido, nombre\_cliente\_pedido, apellido\_cliente\_pedido, telefono\_cliente\_pedido, correo\_cliente\_pedido, direccion\_cliente\_pedido, producto\_pedido, cantidad\_pedido, and precio\_pedido. The data grid shows three rows of data corresponding to the JSON response from the API. Below the table, there are buttons for actions like 'Copiar al portapapeles', 'Exportar', 'Mostrar gráfico', and 'Crear vista'.

id_pedido	nombre_cliente_pedido	apellido_cliente_pedido	telefono_cliente_pedido	correo_cliente_pedido	direccion_cliente_pedido	producto_pedido	cantidad_pedido	precio_pedido
80	Benito	Reyes	23456	benit@gmail.com	Pasto	Piña	1	1000
81	Paula	Jaramillo	3205902369	paula@gmail.com	Pasto	Naranja	6	5000
82	Clemente	Vargas	322569856	vargas@gmail.com	Pasto	Manzana	6	1500

## Verificando operación eliminar pedidos

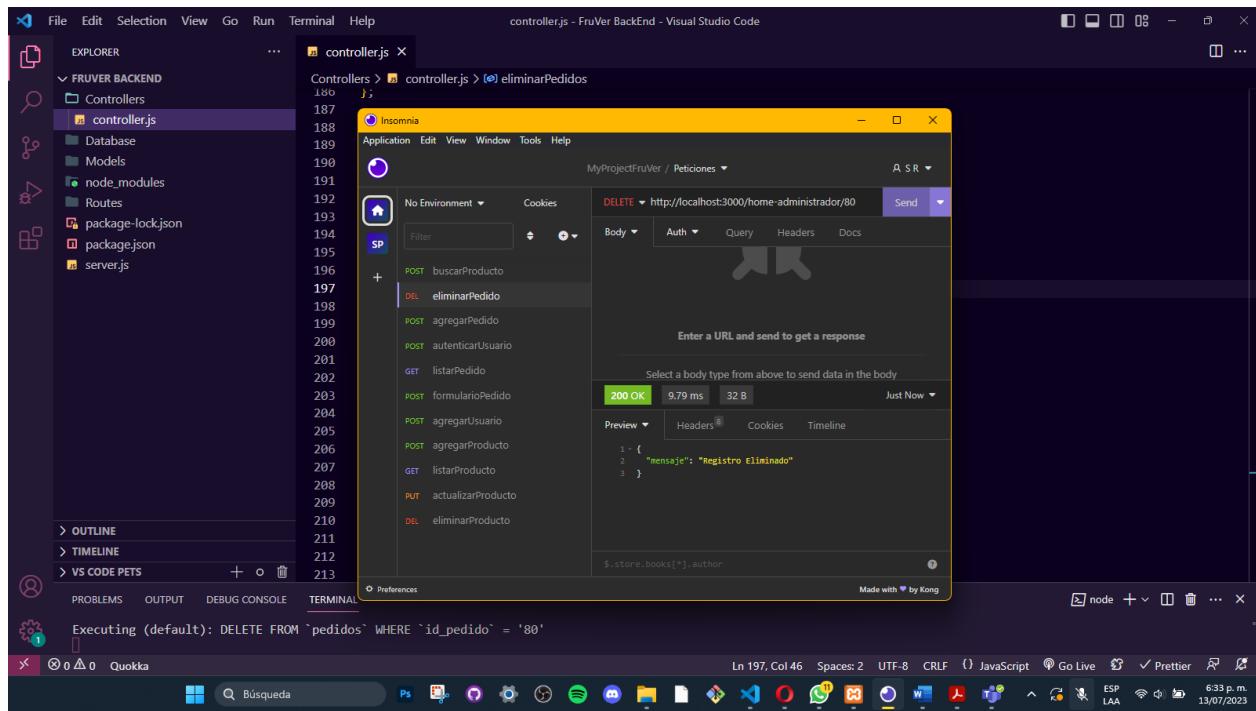


```
File Edit Selection View Go Run Terminal Help controller.js - FruVer BackEnd - Visual Studio Code

EXPLORER
FRUVER BACKEND
  Controllers
    controller.js
      Database
      Models
      node_modules
      Routes
      package-lock.json
      package.json
      server.js

controller.js
...
186   };
187
188   const listarPedidos = async (req, res) => {
189     try {
190       const pedido = await Pedido.findAll();
191       res.status(200).json(pedido);
192     } catch (error) {
193       res.status(400).json({ mensaje: error });
194     }
195   };
196
197   const eliminarPedidos = async (req, res) => {
198     const { id_pedido } = req.params;
199     try {
200       const respuesta = await Pedido.destroy({
201         where: {
202           id_pedido,
203         },
204       });
205       res.status(200).json({ mensaje: "Registro Eliminado" });
206     } catch (error) {
207       res.status(400).json({ mensaje: "Registro No Eliminado" + error });
208     }
209   };
210
211   export {
212     listarProductos,
213     registrarProductos,
214   };
215
216 dido`, `producto_pedido`, `cantidad_pedido`, `precio_pedido` FROM `pedidos` AS `pedidos`;
```

## Operación exitosa



The screenshot shows the Visual Studio Code interface with the controller.js file open. A red box highlights the code for the `eliminarPedidos` function. Below the code editor, the terminal window shows a successful execution of a DELETE command:

```
Executing (default): DELETE FROM `pedidos` WHERE `id_pedido` = '80'
```

A modal window from the Insomnia REST client is overlaid on the code editor. It displays a successful `DELETE` request to `http://localhost:3001/home-administrador/80`. The response status is `200 OK`, the response time is `9.79 ms`, and the response body is:

```
{ "mensaje": "Registro Eliminado" }
```

## Comprobando en la BD

The screenshot shows the phpMyAdmin interface for the 'db\_fruver' database. The left sidebar lists databases and tables, including 'pedidos'. The main area displays the 'pedidos' table with two rows of data:

	id_pedido	nombre_cliente_pedido	apellido_cliente_pedido	telefono_cliente_pedido	correo_cliente_pedido	direccion_cliente_pedido	producto_pedido	cantidad_pedido	precio_pedido
1	78	Benito	Reyes	23456	benit@gmail.com	Pasto	Piña	1	10000
2	79	Puala	Jaramillo	3205902369	paula@gmail.com	Pasto	Naranja	6	60000

Below the table, there are buttons for 'Copiar al portapapeles', 'Exportar', 'Mostrar gráfico', and 'Crear vista'.