

**DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA EL
DESARROLLO DE SOFTWARE.**

TALLER UNIDAD 2 BACKEND.

YESSENITH CAMILA YELA CABRERA

**UNIVERSIDAD DE NARIÑO.
INGENIERÍA DE SISTEMAS**

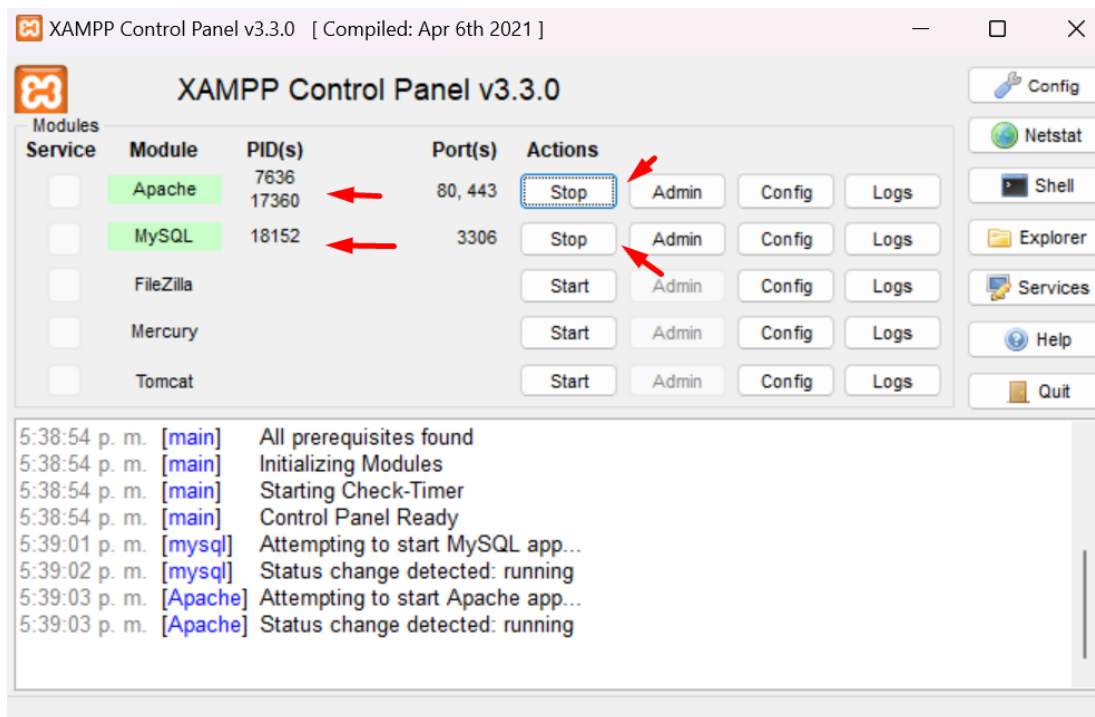
2023

DESARROLLO

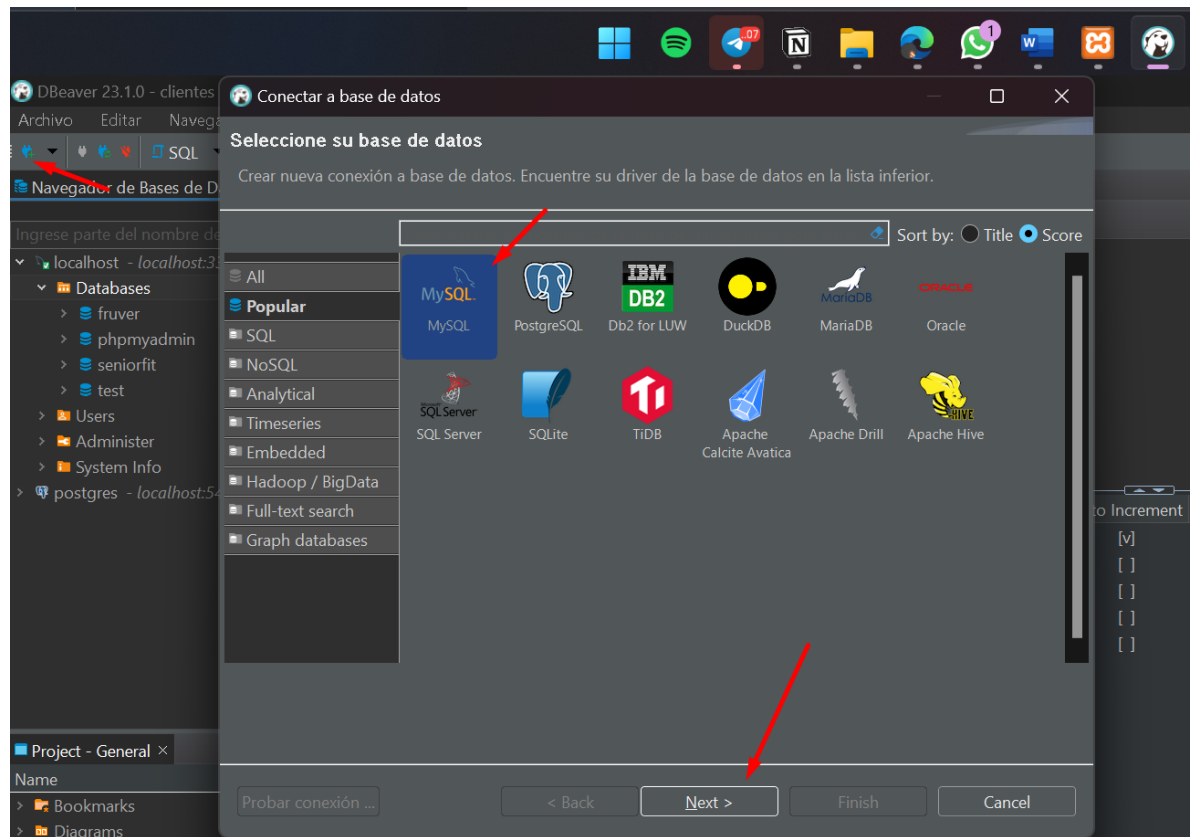
Para iniciar con el backend del proyecto FRUVER, tenemos que iniciar con la creación de la base de datos. Para esto utilizaremos las siguientes herramientas: XAMPP, Insomnia, Dbeaver, Visual Studio y Nodejs.

- ✓ XAMPP: Es una herramienta que proporciona un entorno de desarrollo local. Esto nos permitirá administrar y configurar una base de datos utilizando los servicios de Apache y phpMyAdmin.
- ✓ Insomnia: Es una herramienta que nos permitirá probar los diferentes endpoints del backend a través de solicitudes HTTP. Esto nos ayudará a verificar el funcionamiento correcto de nuestras API.
- ✓ Visual Studio Code: Es un editor de código muy popular y versátil. Lo utilizaremos para escribir y editar nuestro código del backend.
- ✓ Node.js: Es un entorno de ejecución de JavaScript del lado del servidor. Nos permitirá ejecutar nuestro código de backend y aprovechar las capacidades de JavaScript en el servidor.

1- Iniciamos abriendo xampp y encendiendo los servicios de apache y mysql.



2- Abrimos Dbeaver y vamos a establecer la conexión con mysql



Conectar a base de datos

MySQL ajustes de conexión

General Driver properties SSH Proxy SSL

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:mysql://localhost:3306/

Server Host: localhost Port: 3306

Database:

Authentication (Database Native)

Nombre de usuario: root

Contraseña: ☒ Save password locally

Advanced

Server Time Zone: Auto-detect

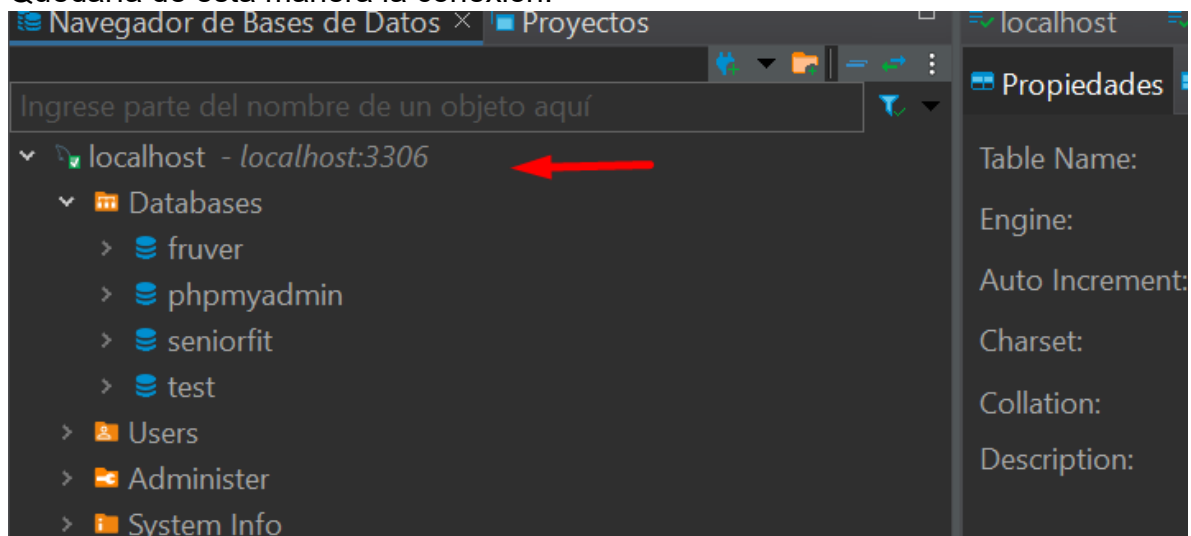
Local Client: C:\xampp\mysql

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

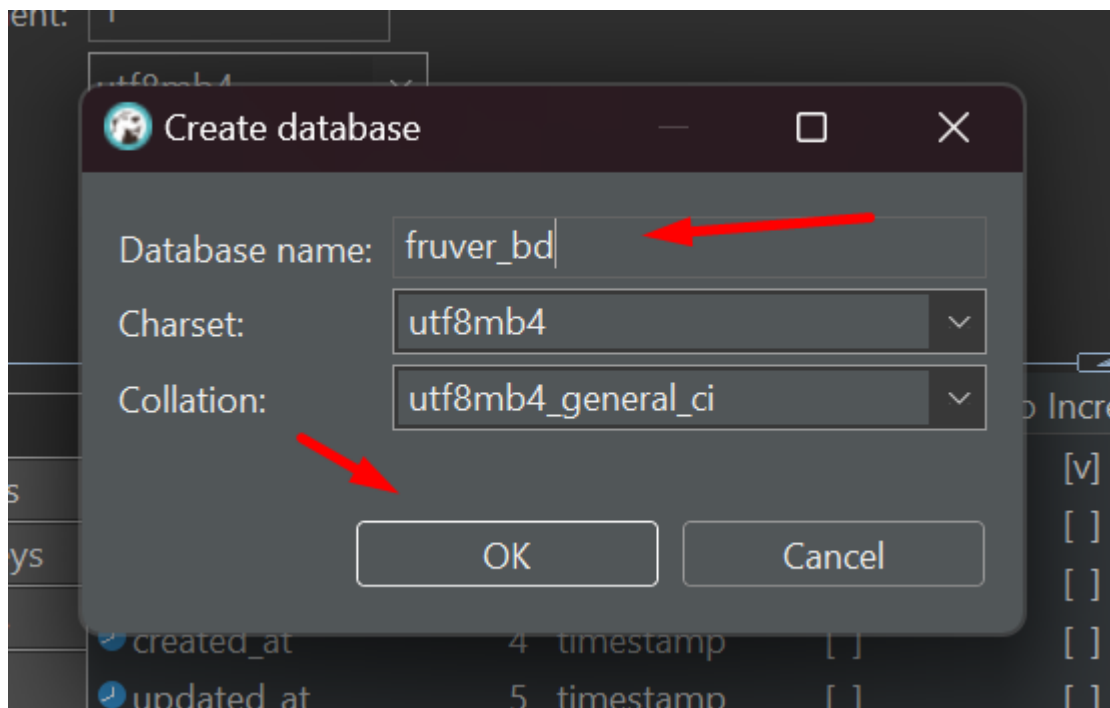
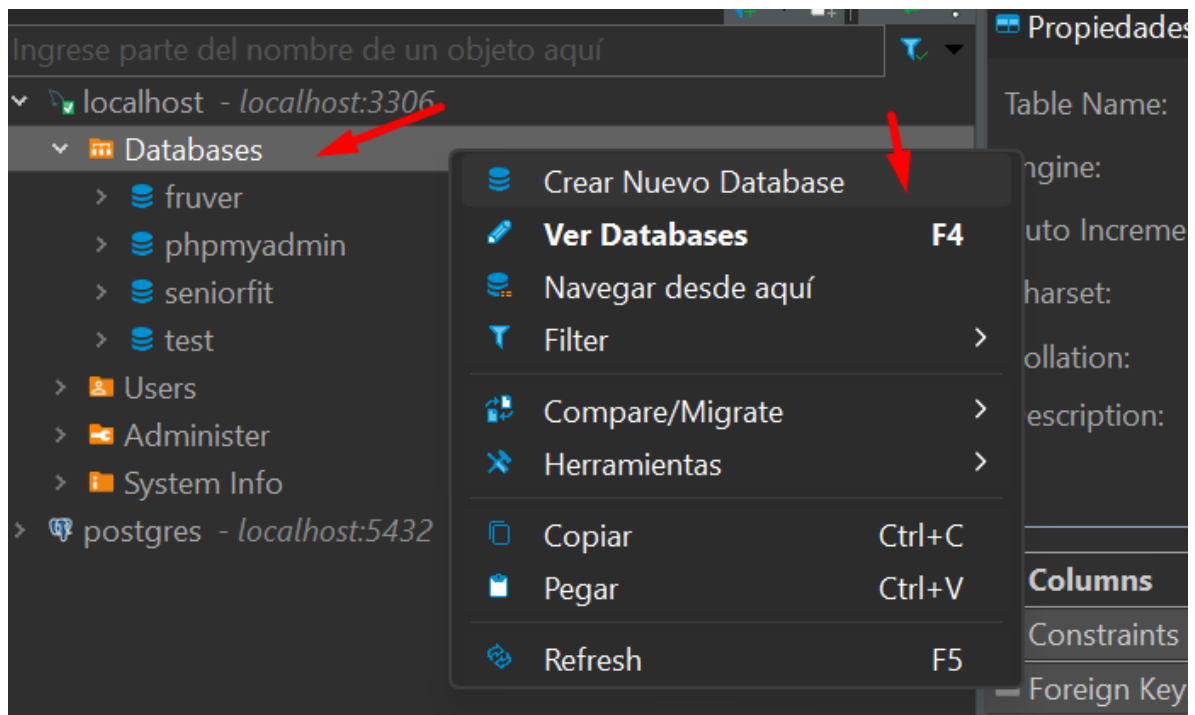
Driver name: MySQL Driver Settings Licencia del driver

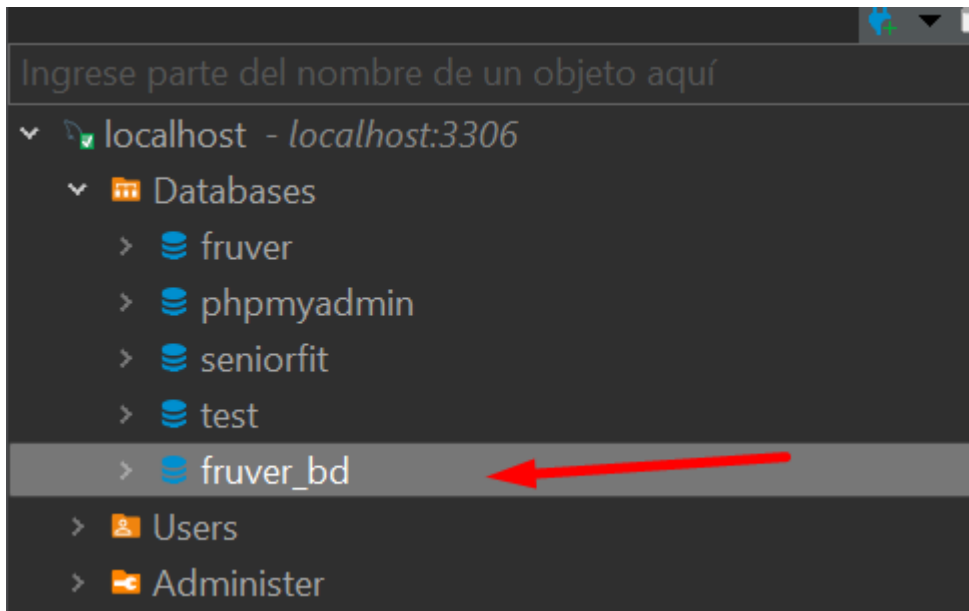
Probar conexión ... < Back Next > Finish Cancel

Quedaría de esta manera la conexión.



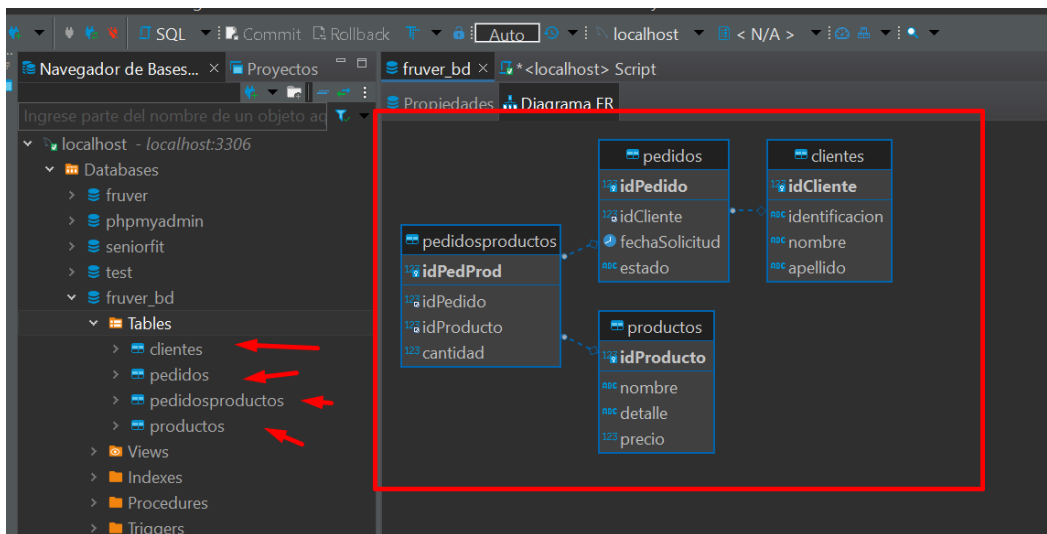
3- Vamos a iniciar creando una nueva base de datos.



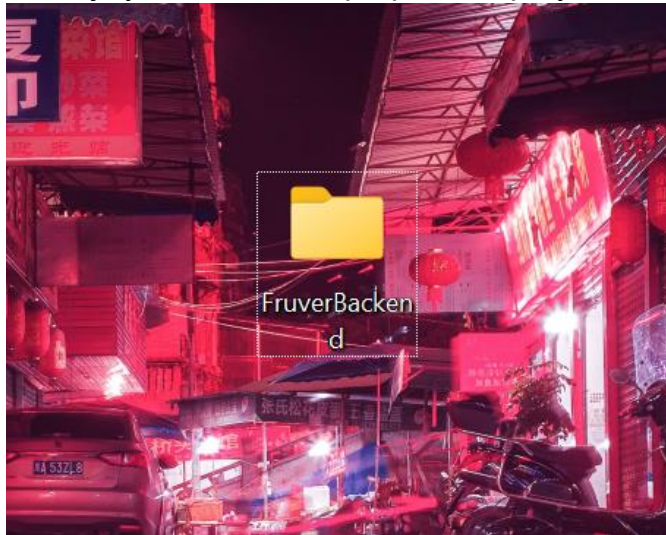


4- Con la base de datos creada y la conexión establecida, podemos iniciar a la construcción de esta. Vamos a establecer 4 tablas relacionadas para nuestra base de datos.

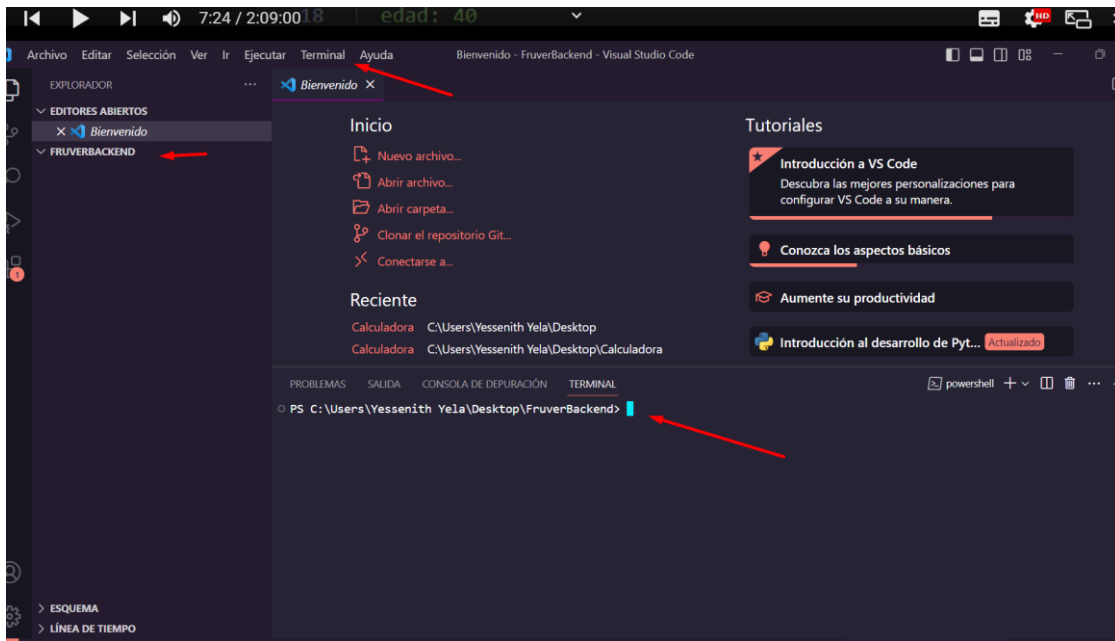
- ✓ Tabla productos, que contendrá la información de los productos de la tienda.
- ✓ Tabla pedidos, que contendrá información de los pedidos realizados en la tienda.
- ✓ Tabla clientes, va relacionada con la table pedidos y contiene información de los clientes que han realizado pedidos en la tienda.
- ✓ Tabla pedidos-productos, que relaciona la tabla pedidos y productos.



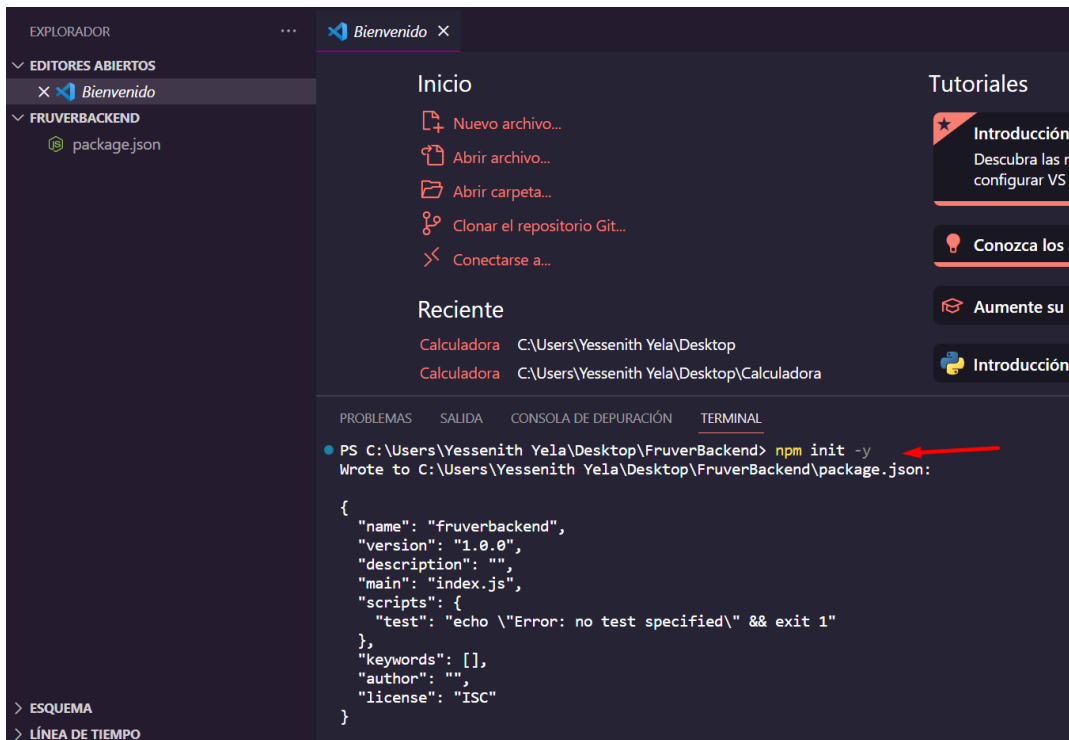
- 5- Con la base de datos establecida, vamos a comenzar creando una carpeta para el proyecto, esta se llamará FruverBackend, en esta integraremos Node.js y las funciones propias del proyecto.



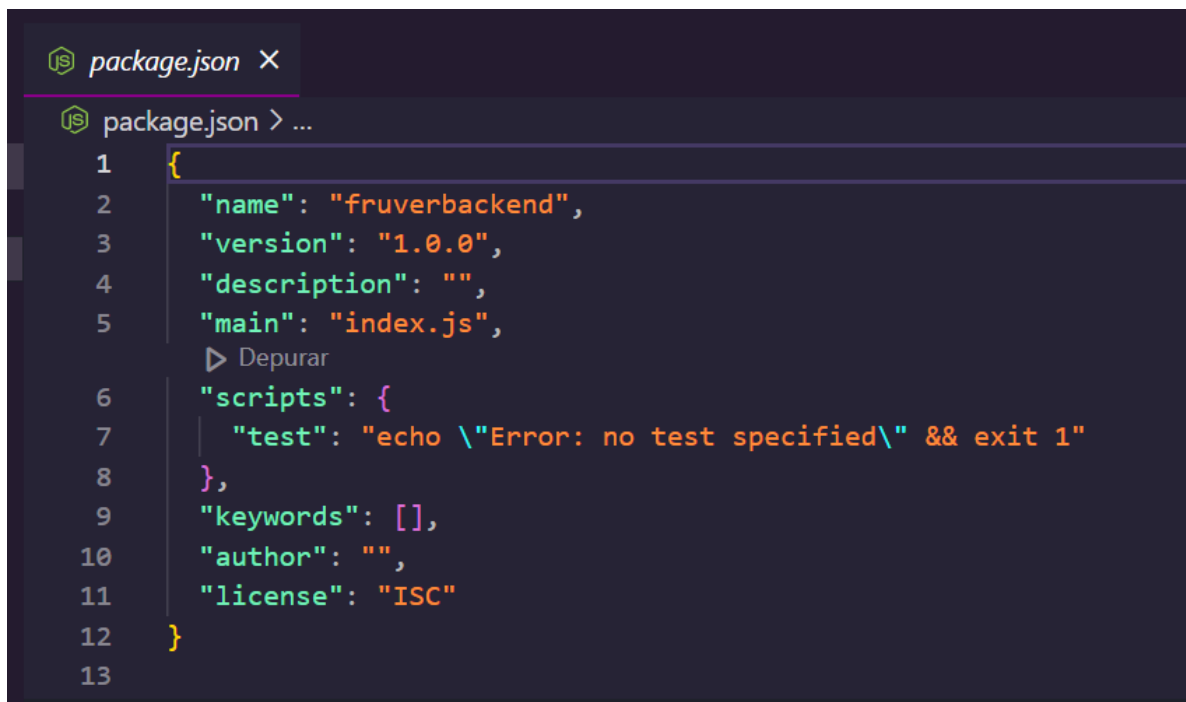
- 6- Continuamos abriendo la carpeta del proyecto con visual studio code. En visual studio code y para implementar node.js se abrirá una nueva terminal para ejecutar instrucciones.



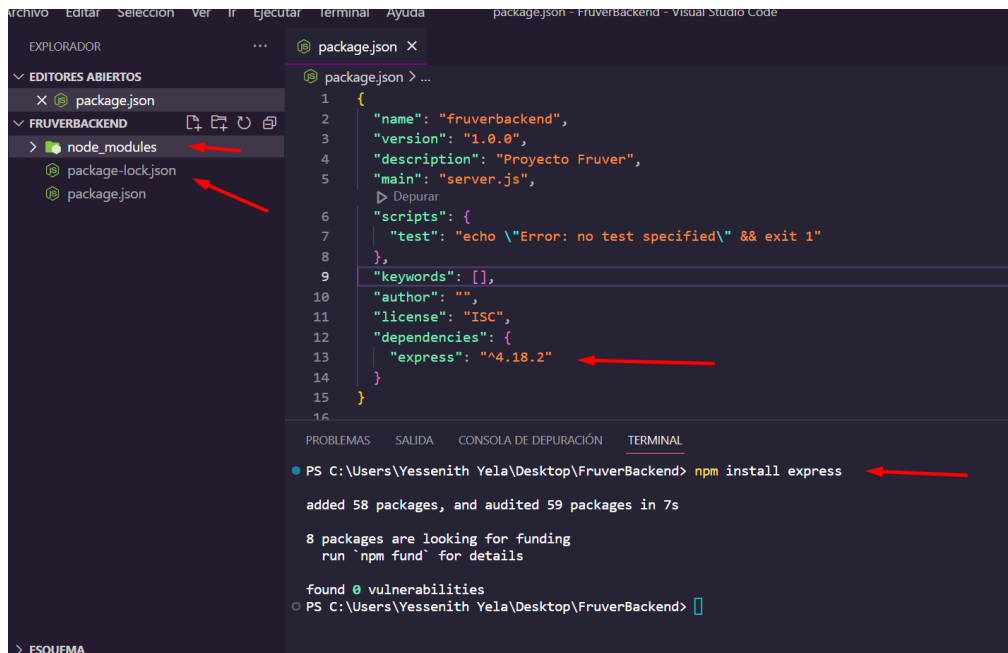
- 7- Instalaremos el npm, que es la herramienta de línea de comandos para node y ayudara para instalación de paquetes o comandos propios de node. Con el comando siguiente inicializaremos un proyecto en node.



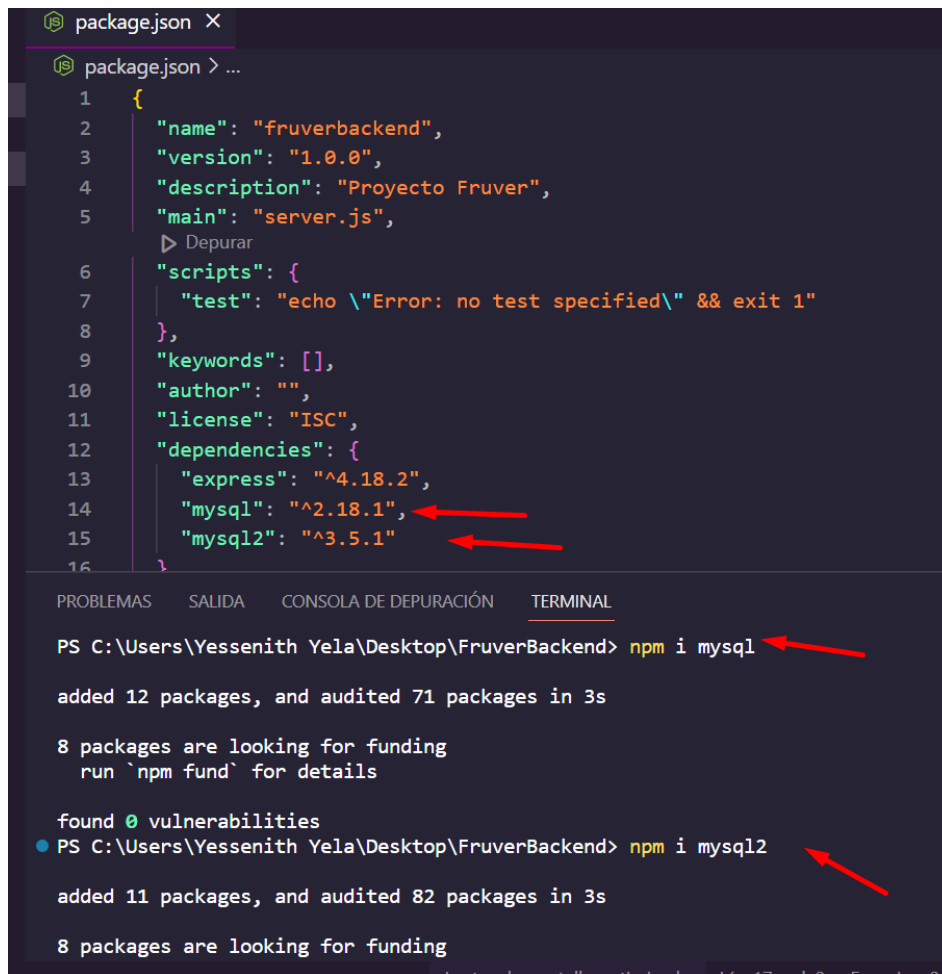
- 8- Al ejecutar el iniciador de node, se crea este archivo “package.json” que permite administrar los módulos del proyecto..



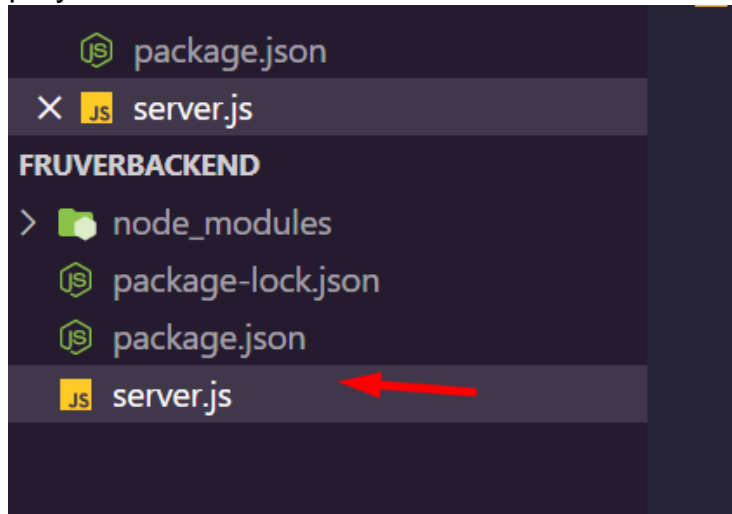
- 9- Vamos a instalar un modificador de paquetes para node, que nos va a permitir desarrollar de una manera más ágil el proyecto.



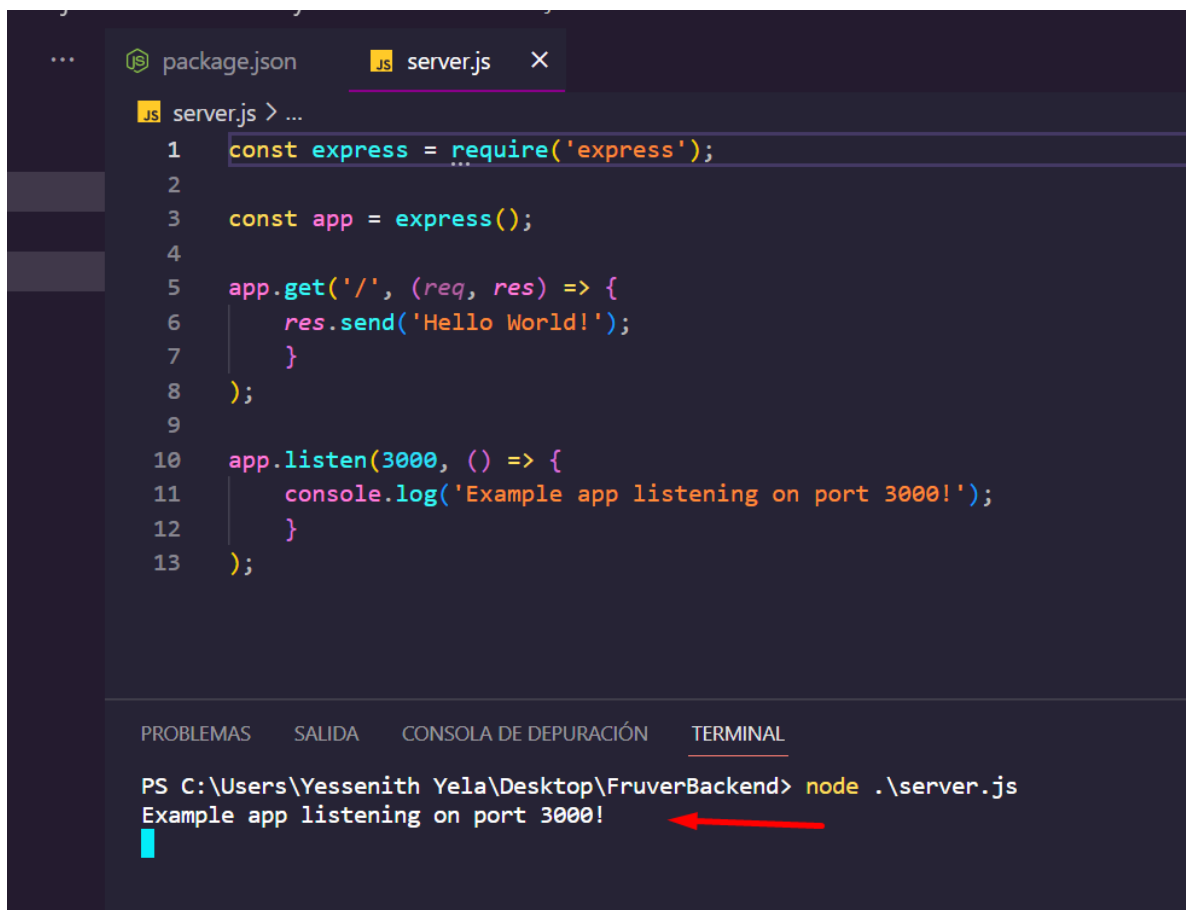
10-Continuamos instalando módulos para manejo de base de datos.



11-Como establecí que la raíz de mi proyecto es server.js creo el archivo. Aquí en el futuro se establecerá la conexión con el puerto y las rutas del proyecto.



12- Comprobamos la conexión con el puerto y corremos node



- 13- Para no tener que estar reiniciando el servicio y que se detecten los cambios del código. Estos cambios son a nivel de desarrollo. Por esto vamos a instalar nodemon.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS C:\Users\Yessenith Yela\Desktop\FruverBackend> npm i nodemon --save-dev
[.....] \ idealTree:FruverBackend: sill idealTree buildDeps
```

- 14-Con el complemento instalado, modificamos el archivo package.js para poder correr el servicio llamando a nodemon.

```
package.json x  server.js

package.json > ...
1  {
2    "name": "fruverbackend",
3    "version": "1.0.0",
4    "description": "Proyecto Fruver",
5    "main": "server.js",
6    "scripts": {
7      "start": "nodemon server.js",
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "express": "^4.18.2",
14     "mysql": "^2.18.1",
15     "mysql2": "^3.5.1"
16   }
17 }
```

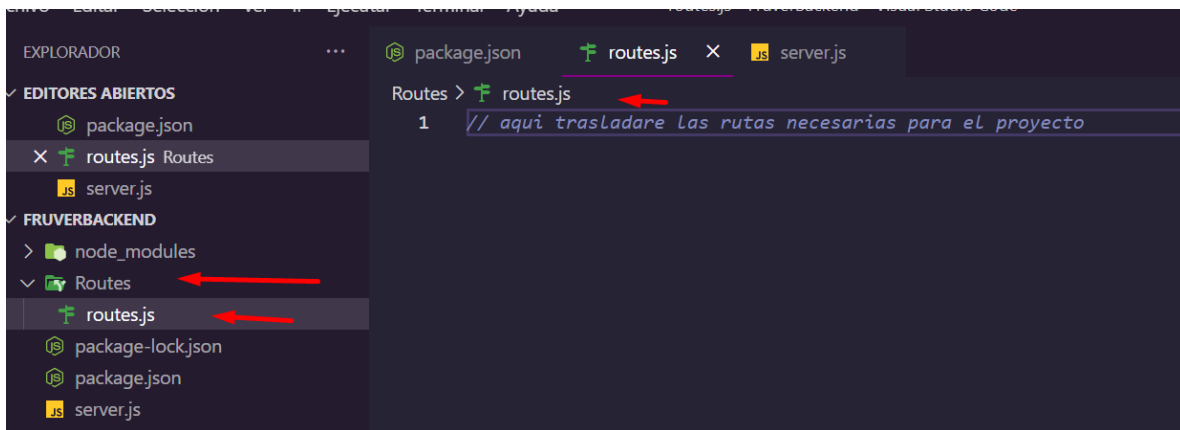
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS C:\Users\Yessenith Yela\Desktop\FruverBackend> npm run start

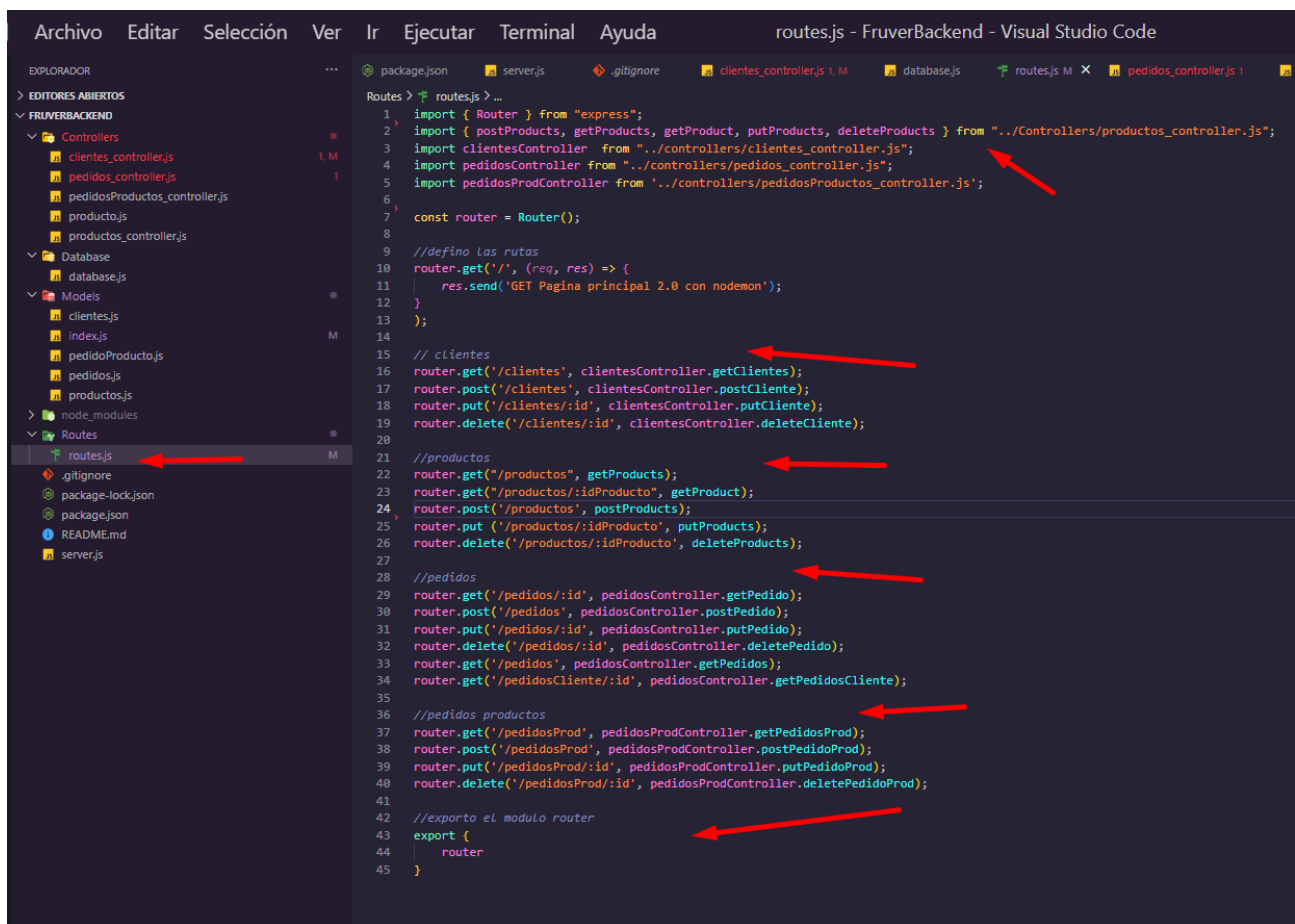
> fruverbackend@1.0.0 start
> nodemon server.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Example app listening on port 3000!
```

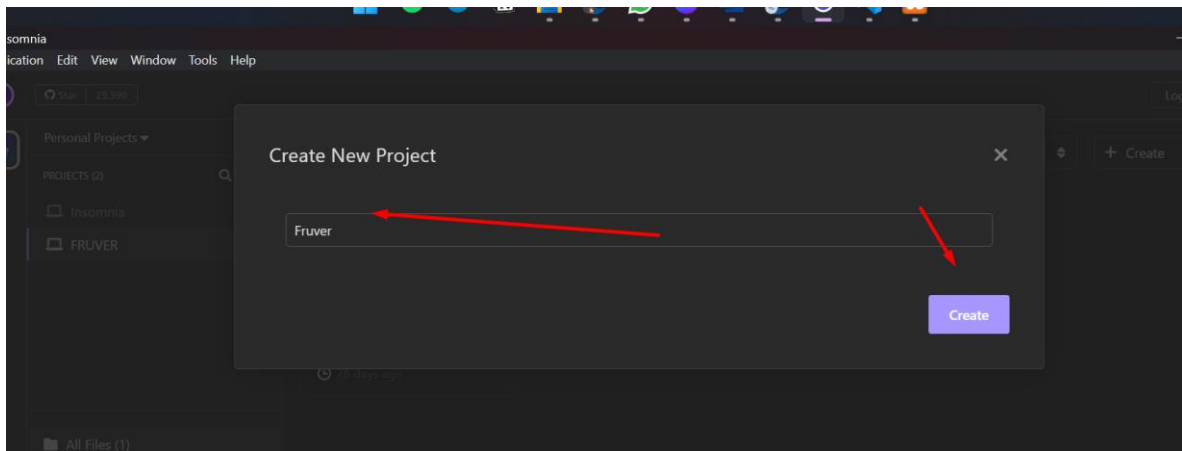
15-Voy a crear una carpeta dentro de mi proyecto de rutas, para poder manejar de manera más eficiente.



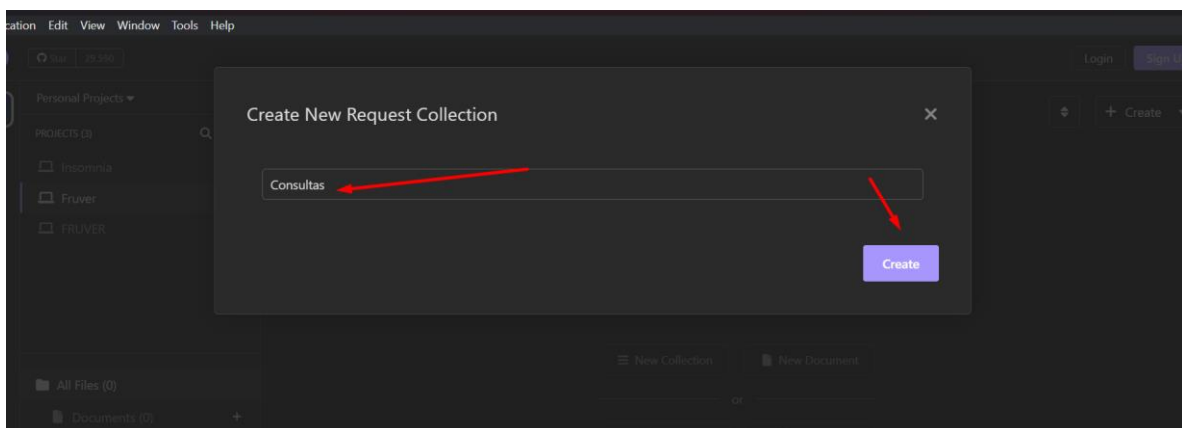
16- En este archivo de rutas (routes.js) vamos a tener las direcciones para gestionar las rutas y solicitudes asociadas a las URL de la aplicación con las funciones y controladores que le corresponden.



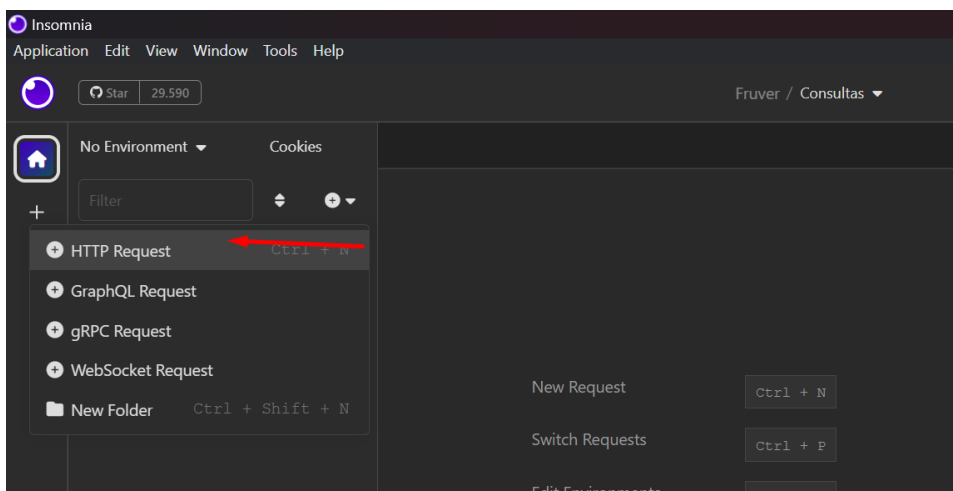
17-En insomnia crearemos un nuevo proyecto para poder hacer nuestras consultas de base de datos.

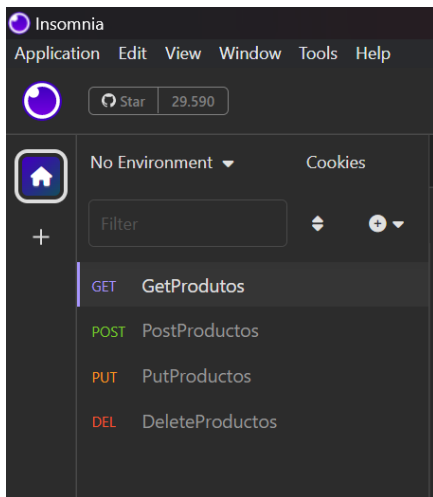


18-Creamos una nueva colección de consultas

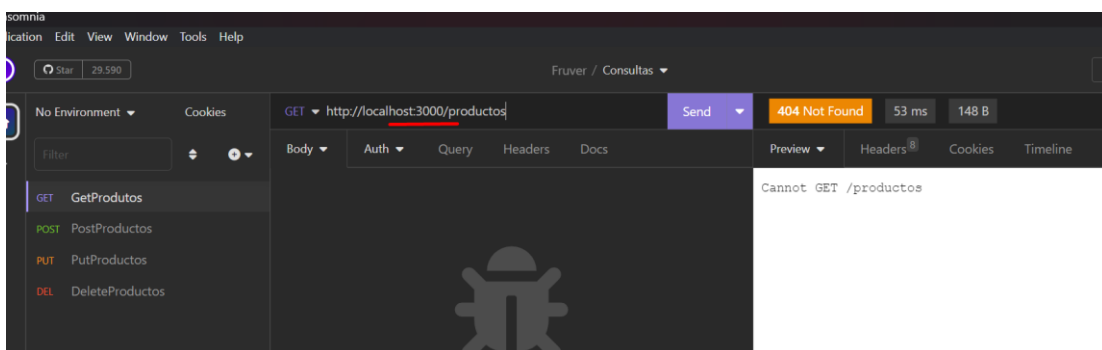


19-Creamos las consultas http de para nuestro backend

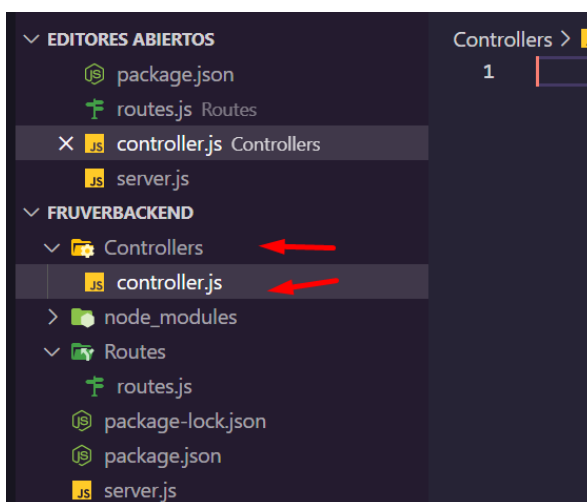




20- Para poder usar estas consultas, debemos dar la dirección de nuestro proyecto, de esta manera. Damos la ruta desde el puerto en que se esta ejecutando. Como siguiente tenemos que modificar las consultas



21-Vamos a crear la carpeta y archivo para los controladores, es decir donde estarán las consultas a la base de datos.



22-En este archivo controlers se conecta la base de datos y se dan las consultas para la interacción de la base de datos.

Cliente

```
const clientes_controller.js - FruverBackend - Visual Studio Code

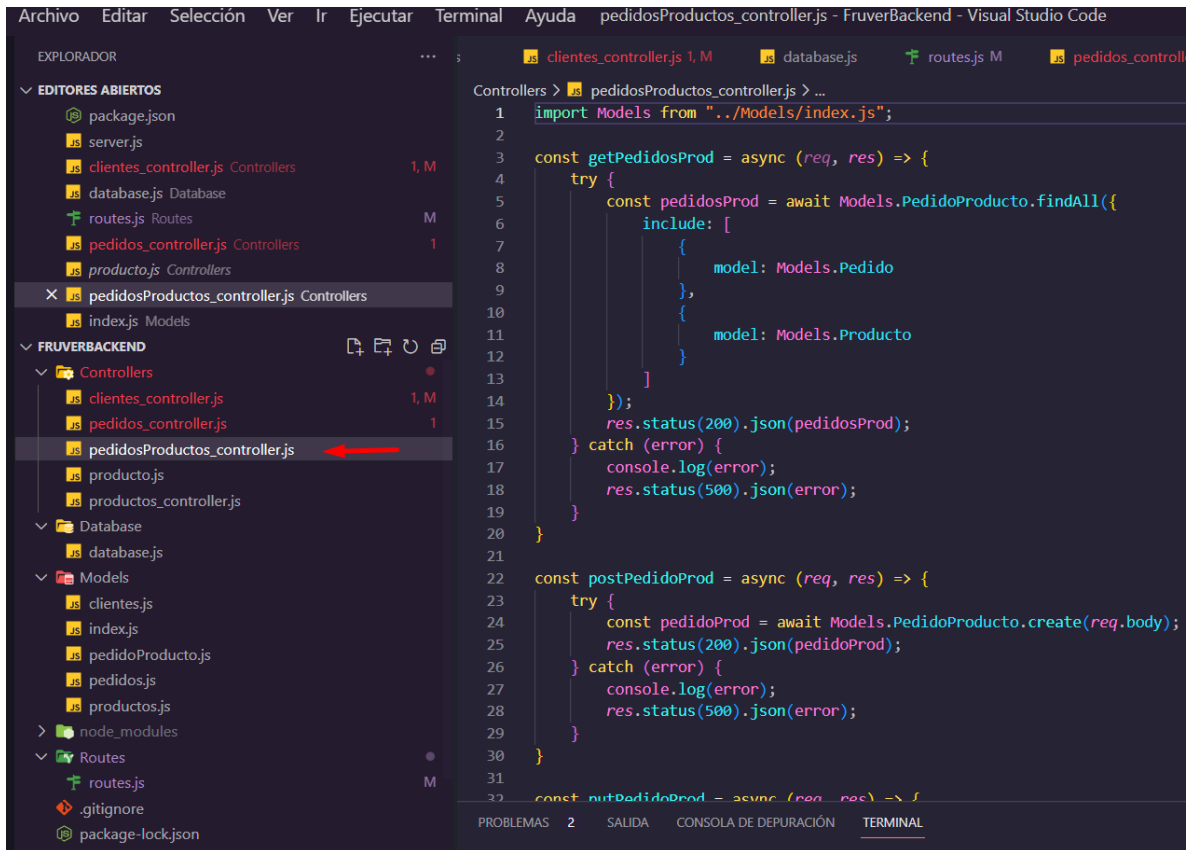
1  import Models from '../models/index.js';
2
3  const getClientes = async (req, res) => {
4    try {
5      const productos = await Models.Cliente.findAll();
6      res.status(200).json(productos);
7    } catch (error) {
8      res.status(400).json({mensaje: error});
9    }
10  }
11
12  const postCliente = async (req, res) => {
13    try {
14      const cliente = await Models.Cliente.create(req.body);
15      res.status(200).json(cliente);
16    } catch (error) {
17      console.log(error);
18      res.status(500).json(error);
19    }
20  }
21
22  const putCliente = async (req, res) => {
23    try {
24      let idCliente = req.params.id;
25      let cliente = await Models.Cliente.findByPk(idCliente);
26      cliente = await cliente.update(req.body);
27      res.status(200).json(cliente);
28    } catch (error) {
29      console.log(error);
30      res.status(500).json(error);
31    }
32  }
33
```

Pedido

```
const pedidos_controller.js - FruverBackend - Visual Studio Code

1  import Models from '../models/index.js';
2
3  const getPedidos = async (req, res) => {
4    try {
5      let idPedido = req.params.id;
6      let pedido = await Models.Pedido.findByPk(idPedido);
7      res.status(200).json(pedido);
8    } catch (error) {
9      console.log(error);
10     res.status(500).json(error);
11   }
12 }
13
14 const postPedido = async (req, res) => {
15   let { idCliente } = req.body;
16   try {
17     const pedido = await Models.Pedido.create({
18       idCliente: idCliente
19     });
20
21     const pedidoProducto = await Models.PedidoProducto.create({
22       idPedido: pedido.idPedido,
23       idProducto: req.body.idProducto,
24       cantidad: req.body.cantidad
25     });
26
27     res.status(200).json(pedido);
28   } catch (error) {
29     console.log(error);
30     res.status(500).json(error);
31   }
32 }
33
```

Pedido producto

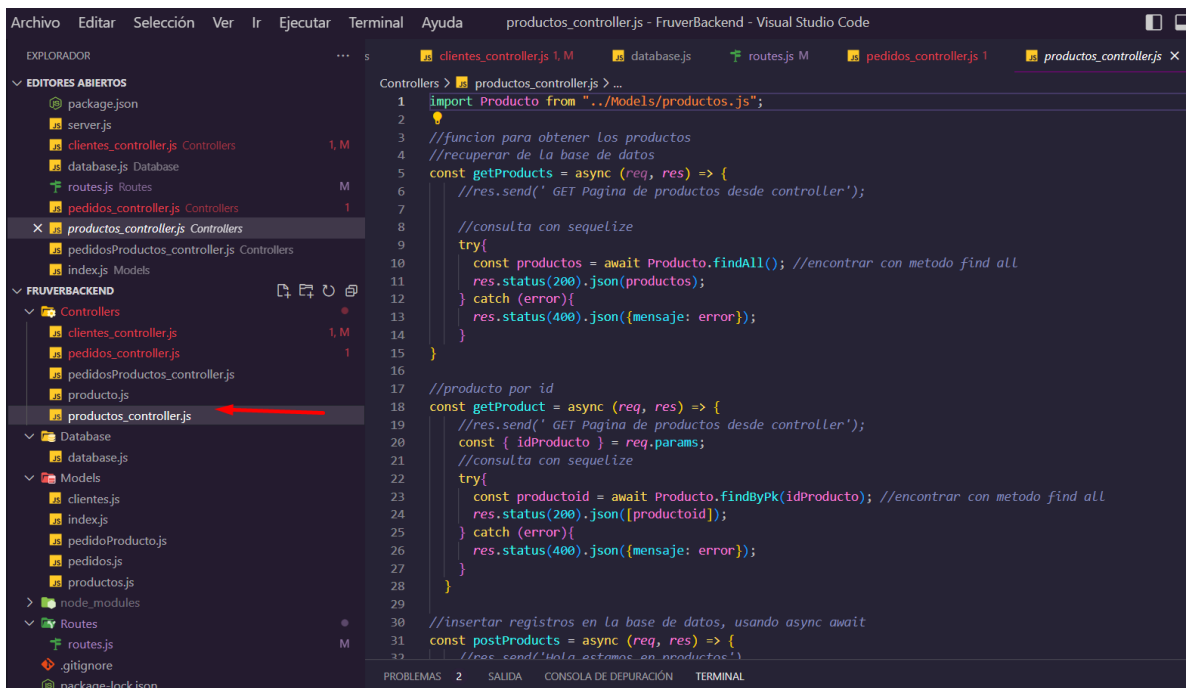


```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  pedidosProductos_controller.js - FruverBackend - Visual Studio Code

EXPLORADOR
  EDITORES ABIERTOS
    package.json
    server.js
    clientes_controller.js Controllers 1, M
    database.js Database
    routes.js Routes M
    pedidos_controller.js Controllers 1
    producto.js Controllers
    pedidosProductos_controller.js Controllers
    index.js Models
  FRUVERBACKEND
    Controllers
      clientes_controller.js 1, M
      pedidos_controller.js 1
      pedidosProductos_controller.js
      producto.js
      productos_controller.js
    Database
      database.js
    Models
      clientes.js
      index.js
      pedidoProducto.js
      pedidos.js
      productos.js
    node_modules
    Routes
      routes.js M
    .gitignore
    package-lock.json

Controllers > pedidosProductos_controller.js > ...
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
32 const putPedidoProd = async (req, res) => {
33   try {
34     const pedidoProd = await Models.PedidoProducto.update(req.body, {
35       where: { id: req.params.id }
36     });
37     res.status(200).json(pedidoProd);
38   } catch (error) {
39     console.log(error);
40     res.status(500).json(error);
41   }
42 }
43
44 const deletePedidoProd = async (req, res) => {
45   try {
46     const pedidoProd = await Models.PedidoProducto.destroy({
47       where: { id: req.params.id }
48     });
49     res.status(200).json(pedidoProd);
50   } catch (error) {
51     console.log(error);
52     res.status(500).json(error);
53   }
54 }
55
PROBLEMAS 2  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
```

Productos

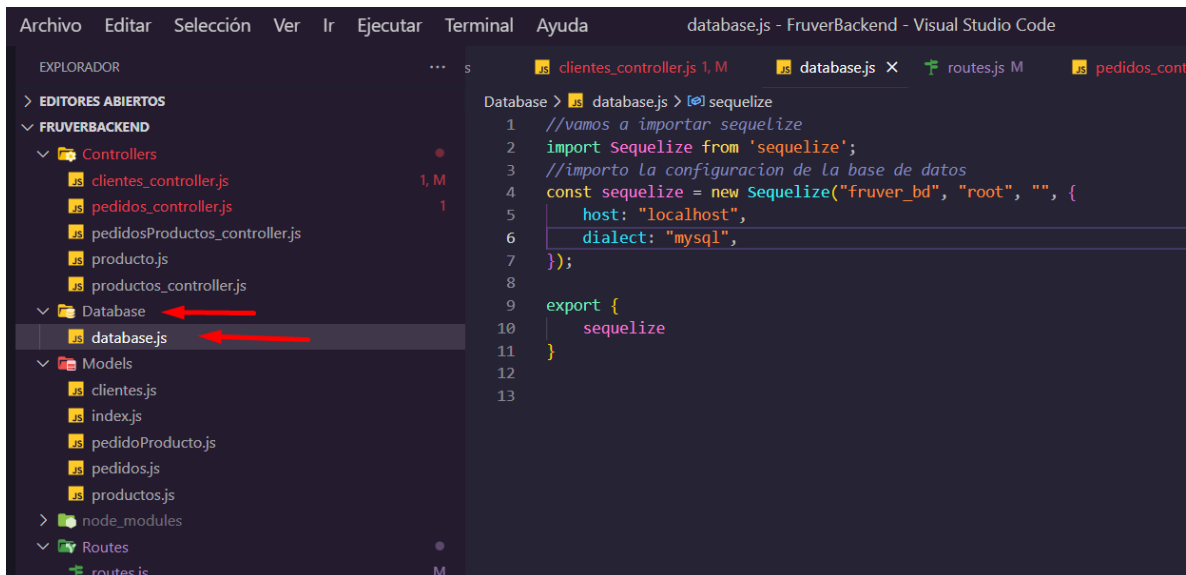


```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  productos_controller.js - FruverBackend - Visual Studio Code

EXPLORADOR
  EDITORES ABIERTOS
    package.json
    server.js
    clientes_controller.js Controllers 1, M
    database.js Database
    routes.js Routes M
    pedidos_controller.js Controllers 1
    productos_controller.js Controllers
    index.js Models
  FRUVERBACKEND
    Controllers
      clientes_controller.js 1, M
      pedidos_controller.js 1
      pedidosProductos_controller.js
      producto.js
      productos_controller.js
    Database
      database.js
    Models
      clientes.js
      index.js
      pedidoProducto.js
      pedidos.js
      productos.js
    node_modules
    Routes
      routes.js M
    .gitignore
    package-lock.json

Controllers > productos_controller.js > ...
1  import Producto from "../Models/productos.js";
2
3  //funcion para obtener Los productos
4  //recuperar de la base de datos
5  const getProducts = async (req, res) => {
6    //res.send(' GET Pagina de productos desde controller');
7
8    //consulta con sequelize
9    try{
10     const productos = await Producto.findAll(); //encontrar con metodo find all
11     res.status(200).json(productos);
12   } catch (error){
13     res.status(400).json({mensaje: error});
14   }
15 }
16
17 //producto por id
18 const getProduct = async (req, res) => {
19   //res.send(' GET Pagina de productos desde controller');
20   const { idProducto } = req.params;
21   //consulta con sequelize
22   try{
23     const productoid = await Producto.findByPk(idProducto); //encontrar con metodo find all
24     res.status(200).json([productoid]);
25   } catch (error){
26     res.status(400).json({mensaje: error});
27   }
28 }
29
30 //insertar registros en la base de datos, usando async await
31 const postProducts = async (req, res) => {
32   //res.send(' Una actamoc en productos ');
33   try{
34     const producto = await Producto.create(req.body);
35     res.status(200).json(producto);
36   } catch (error){
37     res.status(400).json({mensaje: error});
38   }
39 }
40
PROBLEMAS 2  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
```


23-Previo a esto se crea un archivo database.js donde se configura la conexión con la base de datos

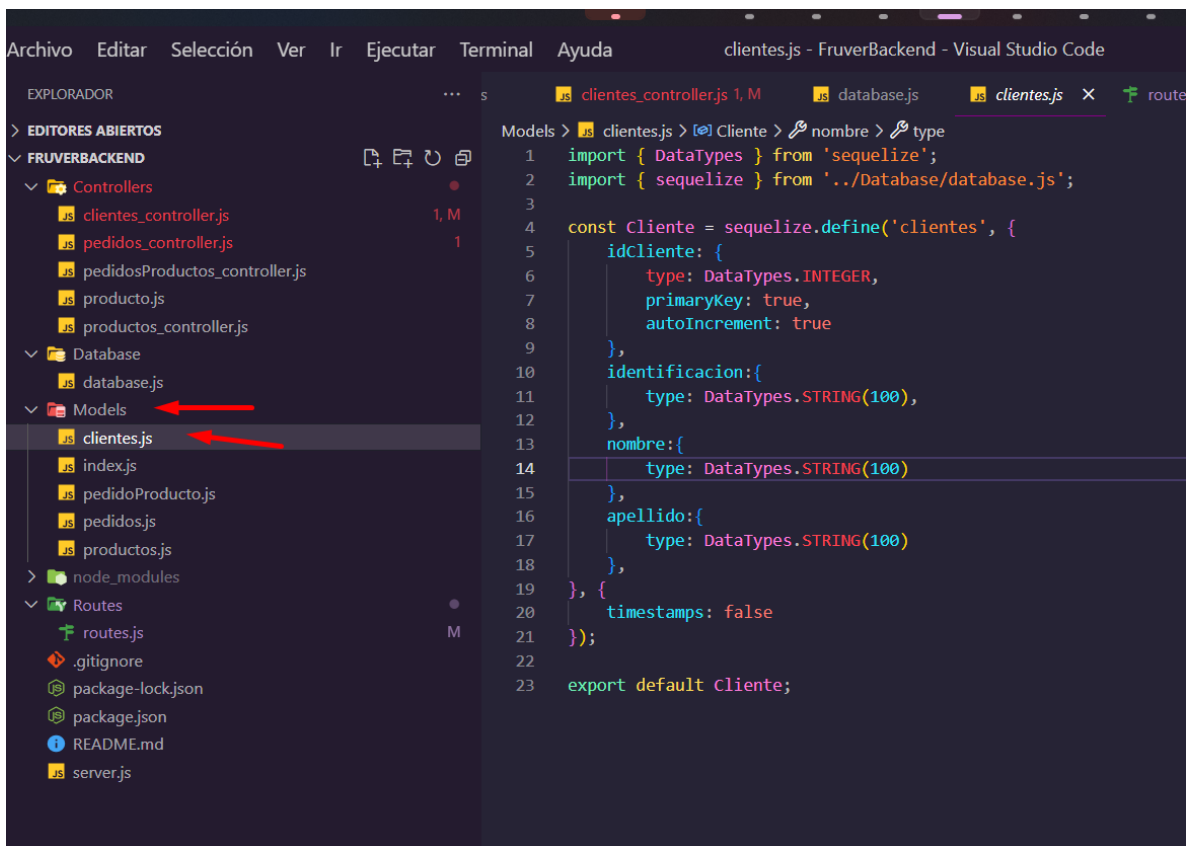


The screenshot shows the Visual Studio Code interface with the 'database.js' file open in the editor. The file is located in the 'Database' folder, which is highlighted in the Explorer sidebar. The code in 'database.js' is as follows:

```
Database > database.js > sequelize
1 //vamos a importar sequelize
2 import Sequelize from 'sequelize';
3 //importo la configuracion de la base de datos
4 const sequelize = new Sequelize("fruver_bd", "root", "", {
5   host: "localhost",
6   dialect: "mysql",
7 });
8
9 export {
10   sequelize
11 }
12
13
```

24- Se crea además la carpeta de models, que contendrá la estructura de las tablas de la base de datos.

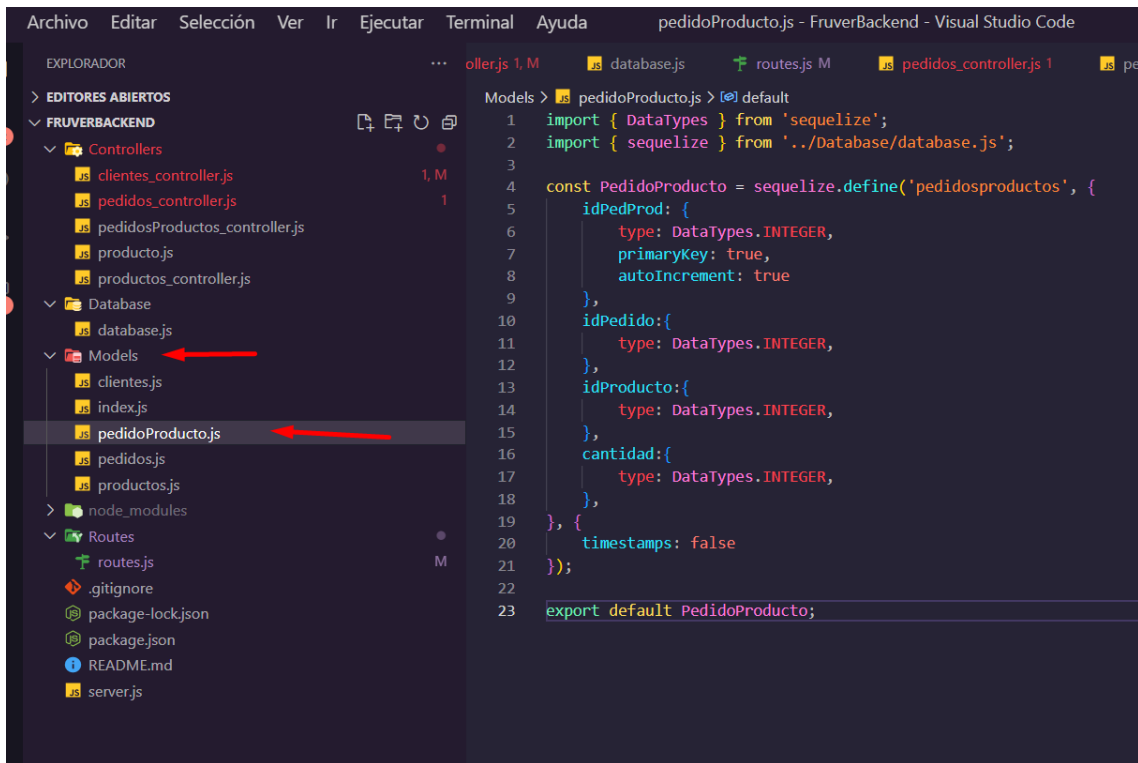
Modelo cliente



The screenshot shows the Visual Studio Code interface with the 'clientes.js' file open in the editor. The file is located in the 'Models' folder, which is highlighted in the Explorer sidebar. The code in 'clientes.js' is as follows:

```
Models > clientes.js > Cliente > nombre > type
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   },
13   nombre:{
14     type: DataTypes.STRING(100)
15   },
16   apellido:{
17     type: DataTypes.STRING(100)
18   },
19 }, {
20   timestamps: false
21 });
22
23 export default cliente;
```

Modelo Pedido productos

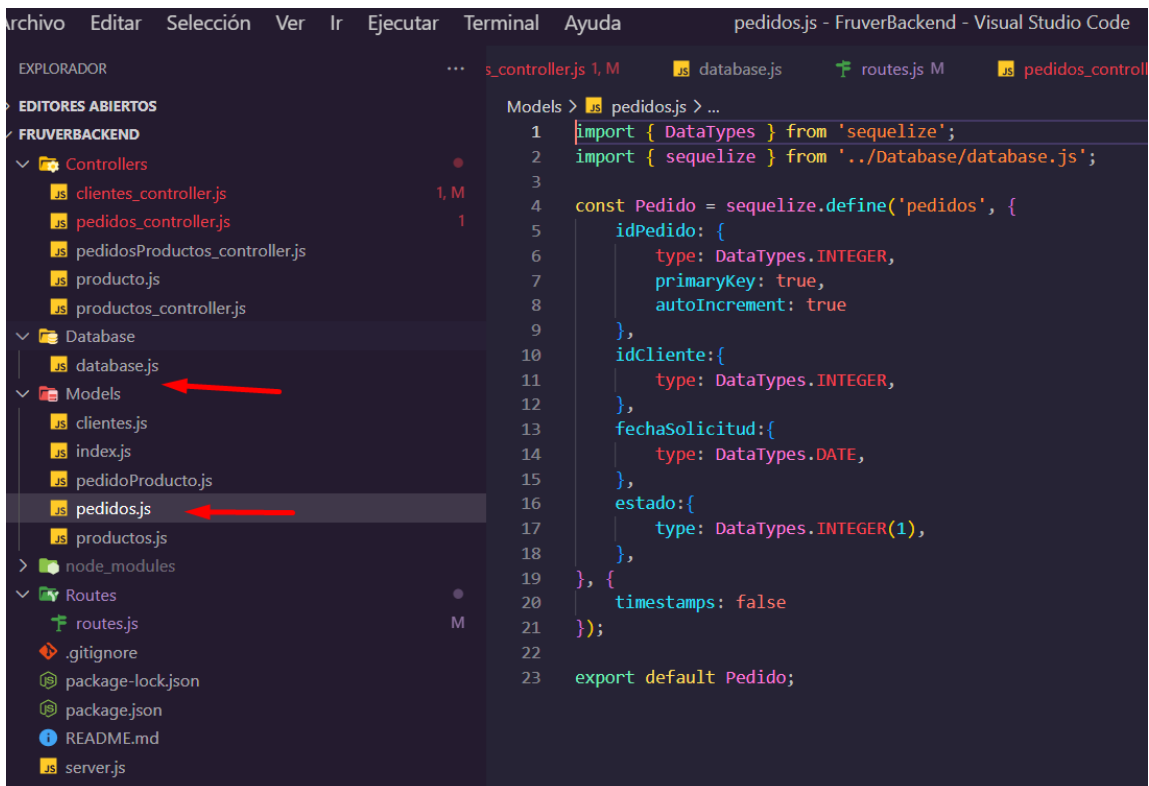


```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  pedidoProducto.js - FruverBackend - Visual Studio Code

EXPLORADOR
EDITORES ABIERTOS
FRUVERBACKEND
  Controllers
    clientes_controller.js
    pedidos_controller.js
    pedidosProductos_controller.js
    producto.js
    productos_controller.js
  Database
    database.js
  Models
    clientes.js
    index.js
    pedidoProducto.js
    pedidos.js
    productos.js
  node_modules
  Routes
    routes.js
    .gitignore
    package-lock.json
    package.json
    README.md
    server.js

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

Modelo Pedidos

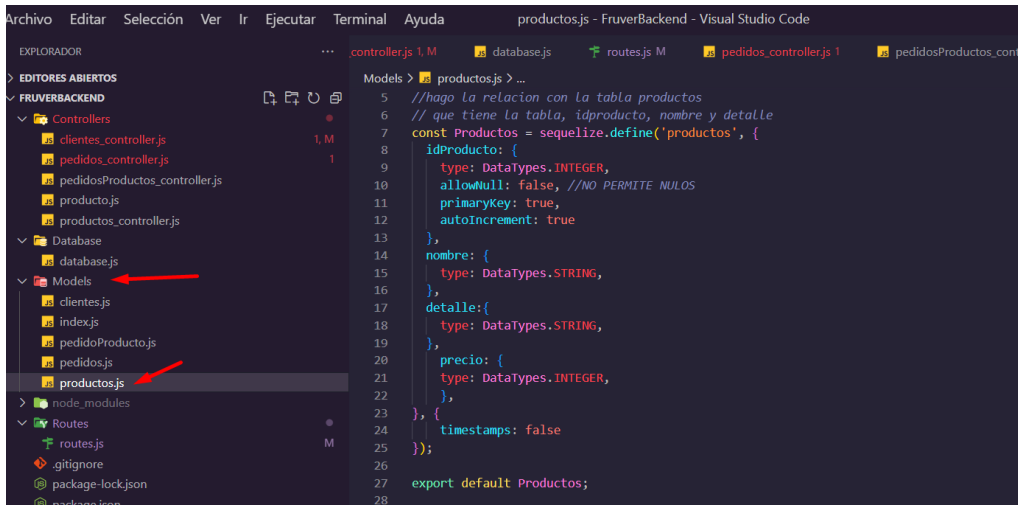


```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  pedidos.js - FruverBackend - Visual Studio Code

EXPLORADOR
EDITORES ABIERTOS
FRUVERBACKEND
  Controllers
    clientes_controller.js
    pedidos_controller.js
    pedidosProductos_controller.js
    producto.js
    productos_controller.js
  Database
    database.js
  Models
    clientes.js
    index.js
    pedidoProducto.js
    pedidos.js
    productos.js
  node_modules
  Routes
    routes.js
    .gitignore
    package-lock.json
    package.json
    README.md
    server.js

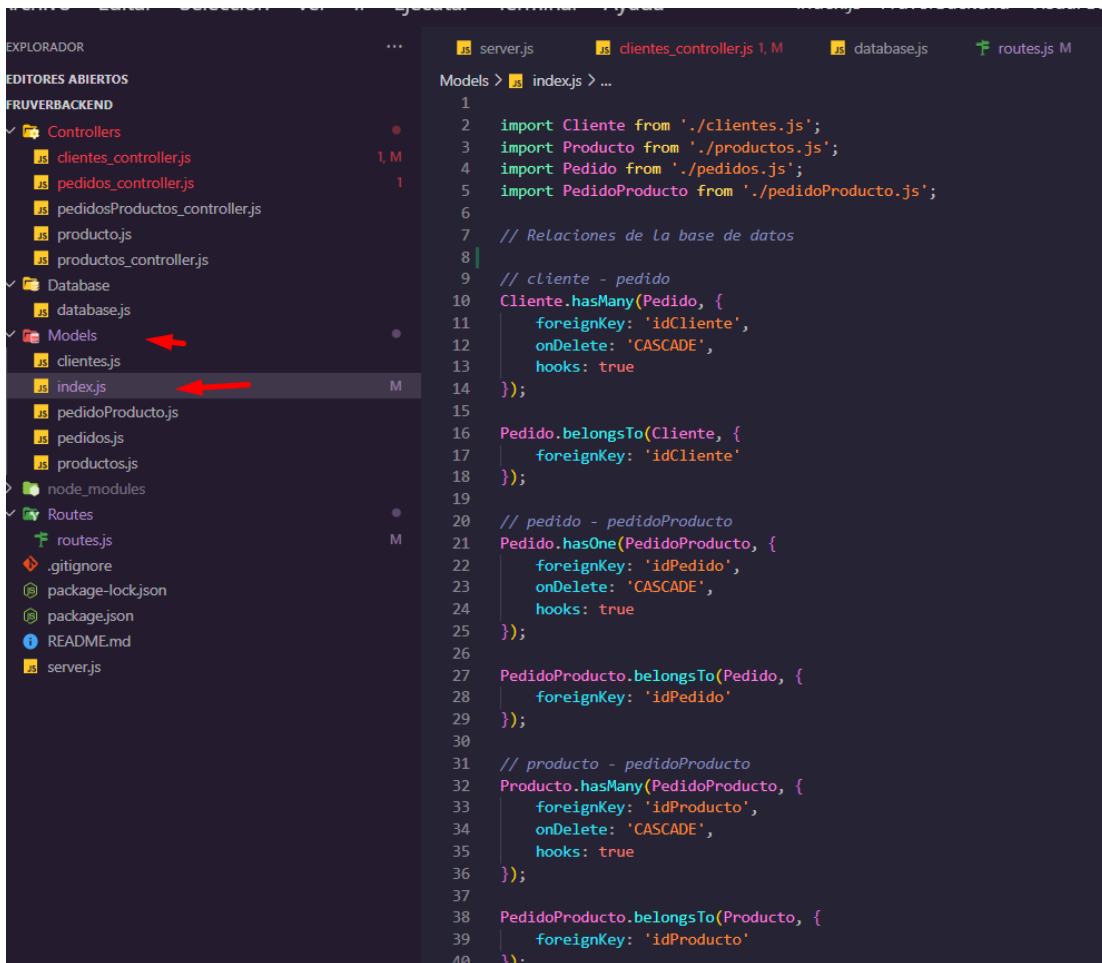
Models > pedidos.js > ...
1  import { DataTypes } from 'sequelize';
2  import { sequelize } from '../Database/database.js';
3
4  const Pedido = sequelize.define('pedidos', {
5    idPedido: {
6      type: DataTypes.INTEGER,
7      primaryKey: true,
8      autoIncrement: true
9    },
10   idCliente:{
11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

Modelo productos



```
5 //hago la relacion con la tabla productos
6 // que tiene la tabla, idproducto, nombre y detalle
7 const Productos = sequelize.define('productos', {
8   idProducto: {
9     type: DataTypes.INTEGER,
10    allowNull: false, //NO PERMITE NULOS
11    primaryKey: true,
12    autoIncrement: true
13  },
14  nombre: {
15    type: DataTypes.STRING,
16  },
17  detalle: {
18    type: DataTypes.STRING,
19  },
20  precio: {
21    type: DataTypes.INTEGER,
22  },
23 }, {
24   timestamps: false
25 });
26
27 export default Productos;
28
```

Modelo index, para las asociaciones entre tablas. Aquí 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.

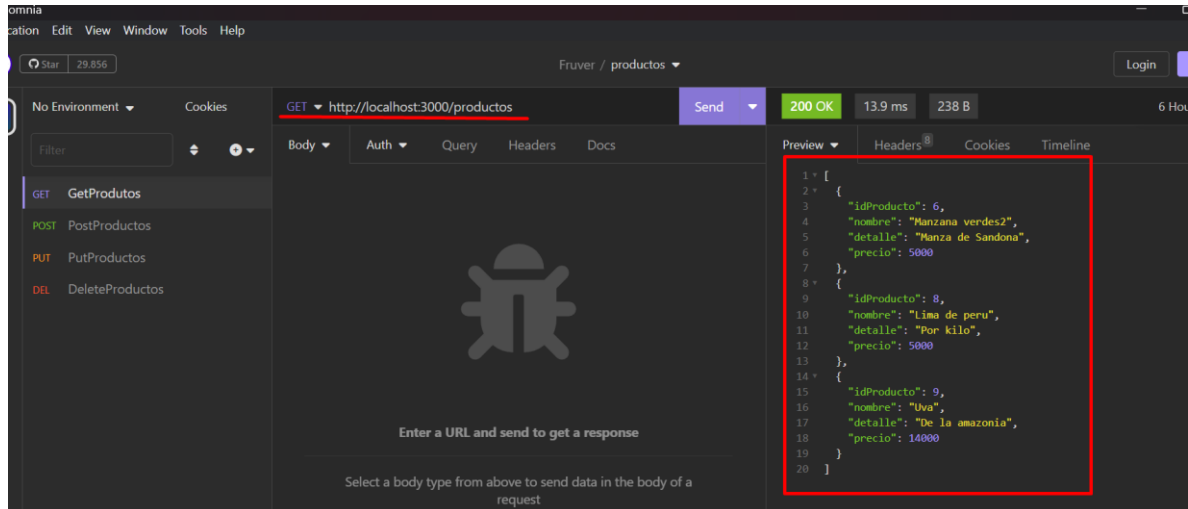


```
1
2 import Cliente from './clientes.js';
3 import Producto from './productos.js';
4 import Pedido from './pedidos.js';
5 import PedidoProducto from './pedidoProducto.js';
6
7 // Relaciones de la base de datos
8
9 // cliente - pedido
10 Cliente.hasMany(Pedido, {
11   foreignKey: 'idCliente',
12  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

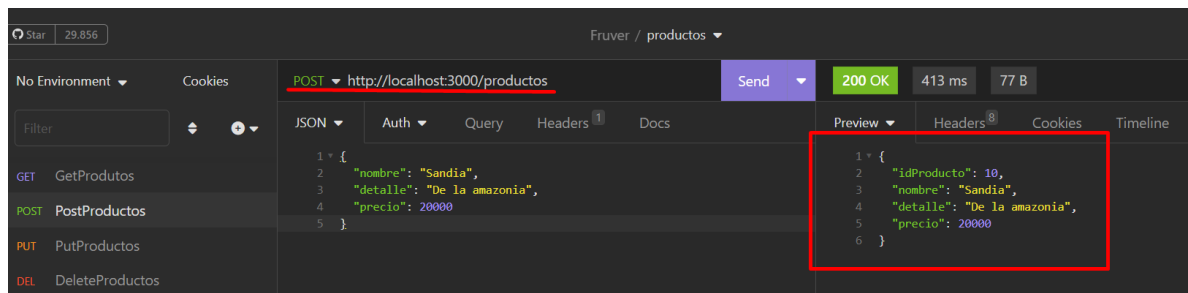
25-Para cada consulta de las rutas utilizamos insomnia, de esta manera.

Consultas de Producto

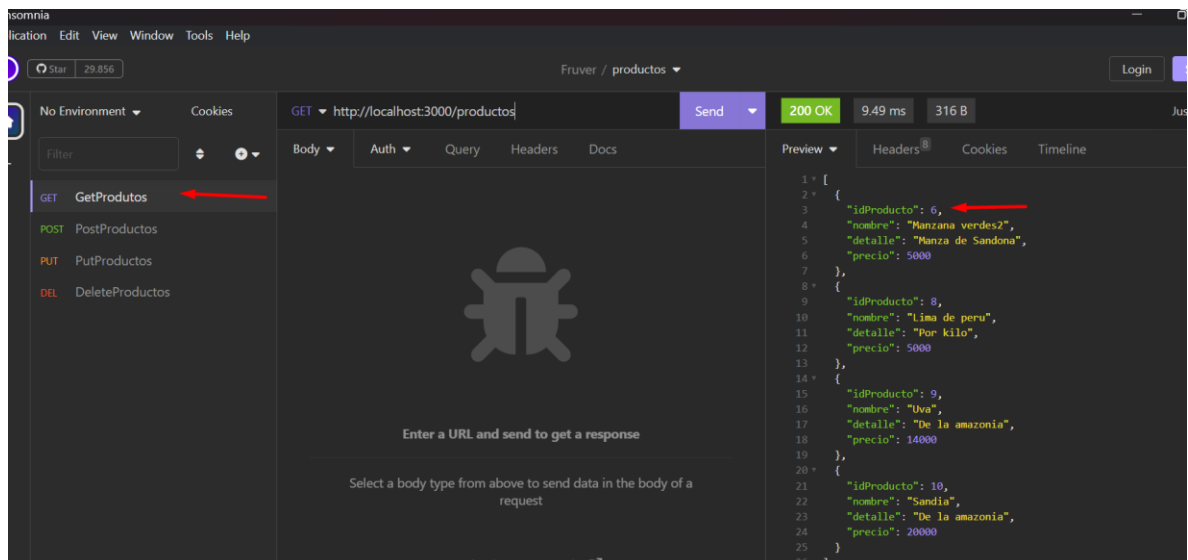
Get

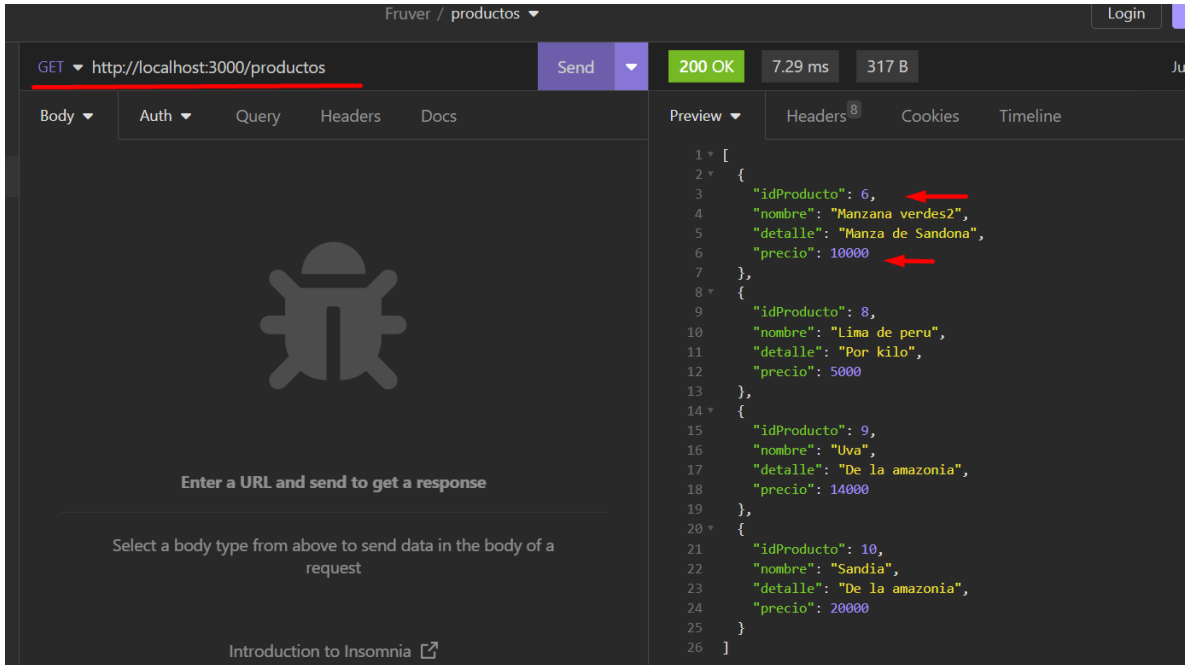
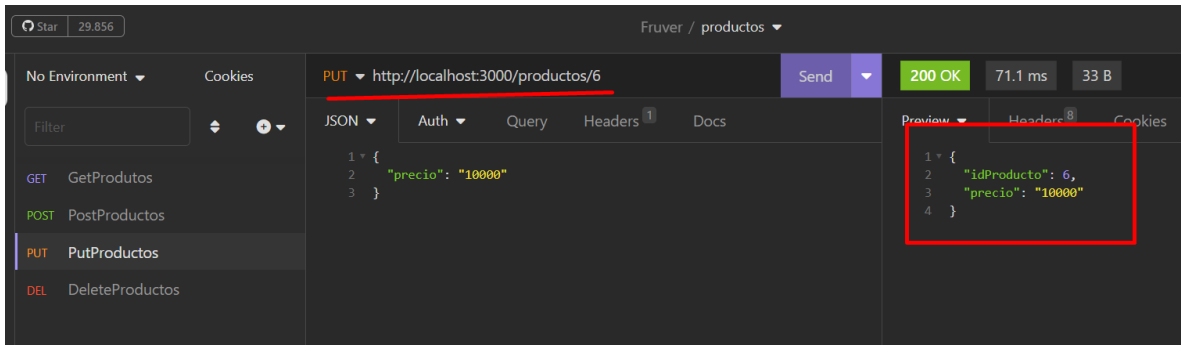


Post

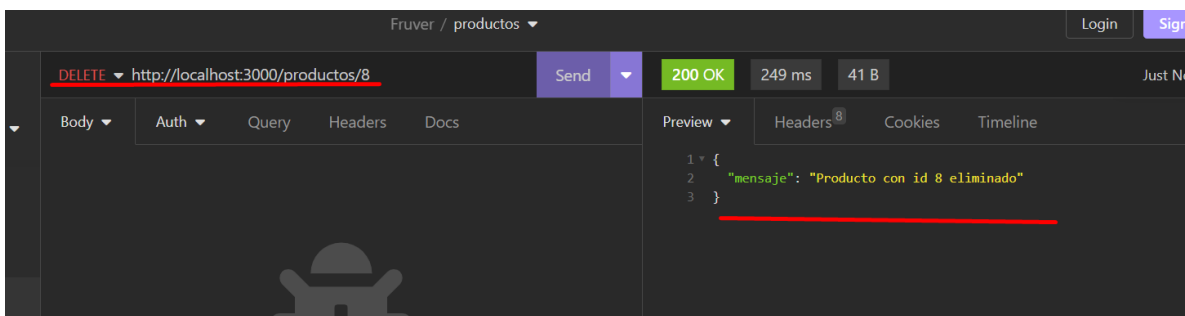


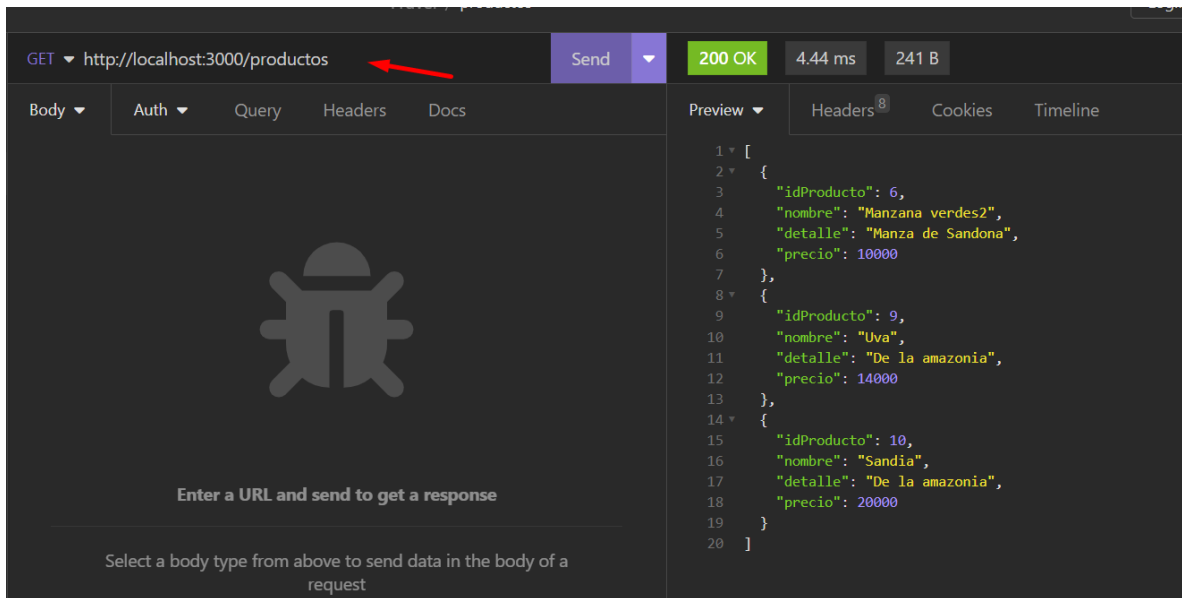
Put





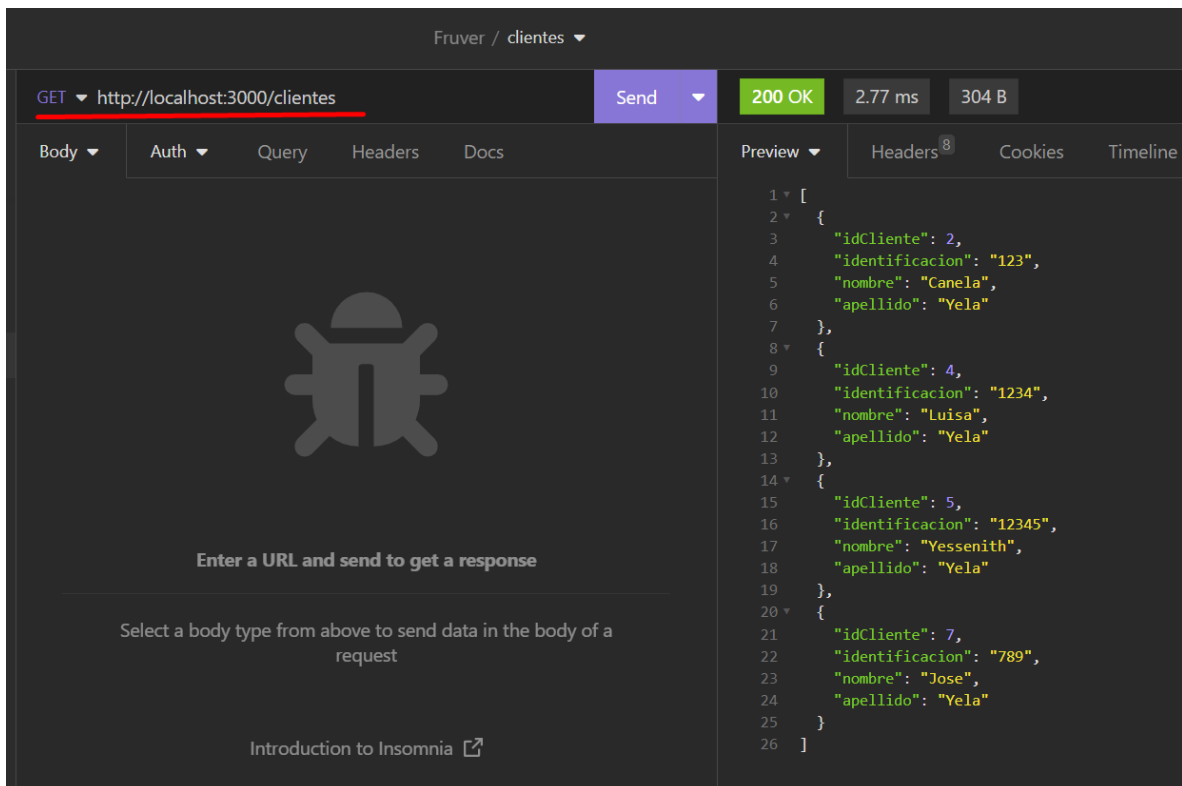
Delete



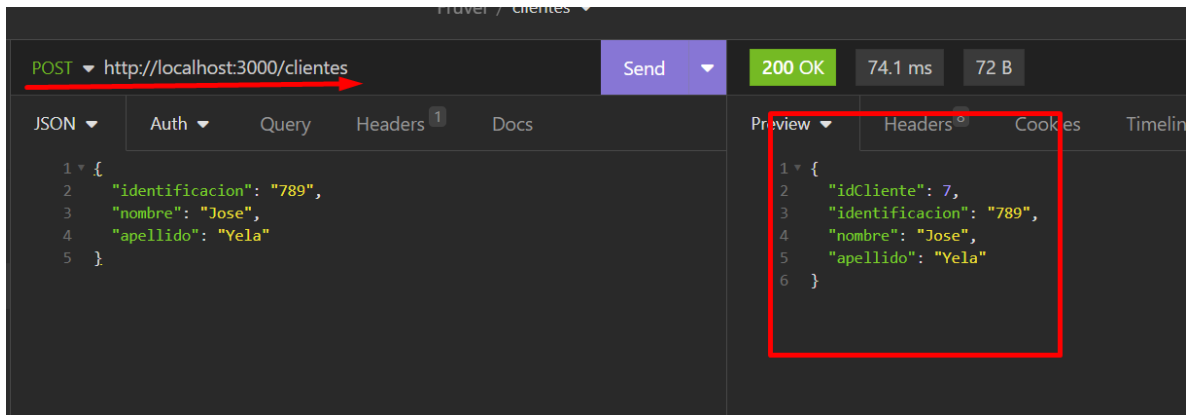


Cliente

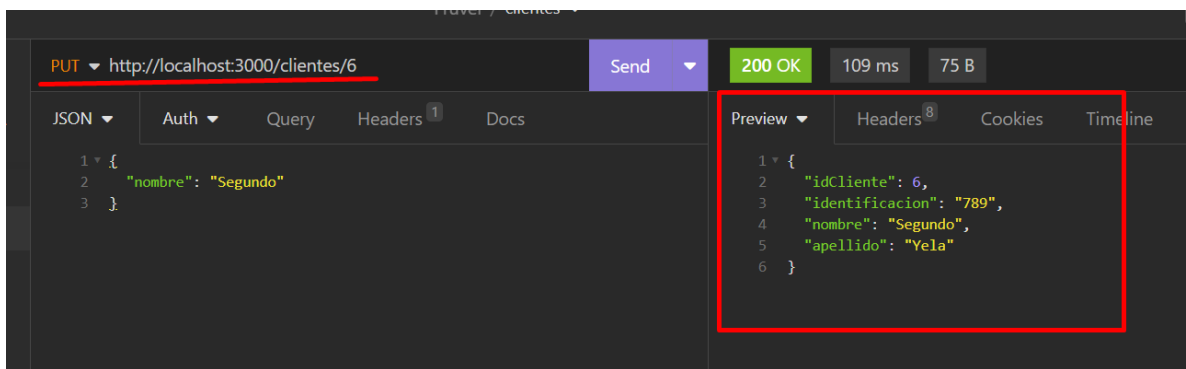
Get



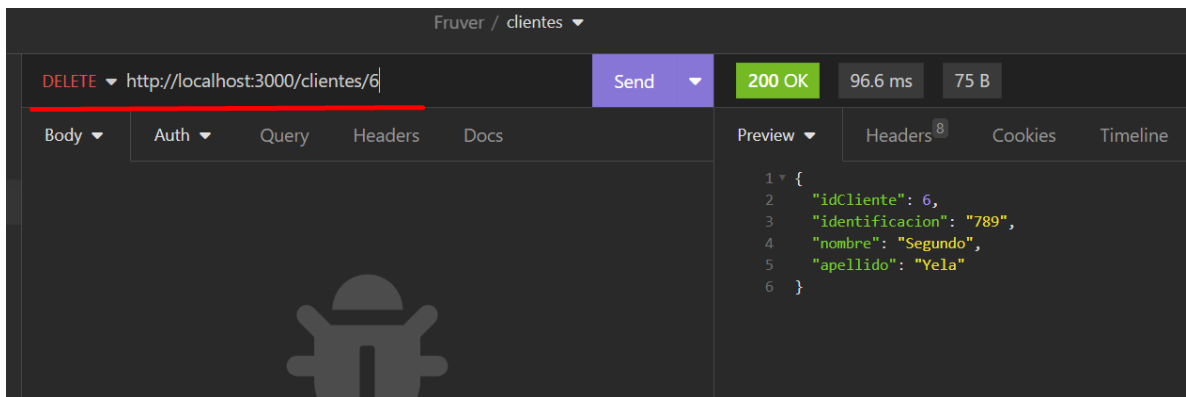
Post



Put



Delete



Consulta Pedidos

Get

GET http://localhost:3000/pedidos Send 200 OK 8.32 ms 330 B

Body Auth Query Headers Docs Preview Headers Cookies Timeline

Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

Introduction to Insomnia

```
1 [
2   {
3     "idPedido": 7,
4     "idCliente": 5,
5     "fechaSolicitud": null,
6     "estado": null,
7     "cliente": {
8       "idCliente": 5,
9       "identificacion": "12345",
10      "nombre": "Yessenith",
11      "apellido": "Vela"
12    },
13    "pedidosproducto": {
14      "idPedProd": 9,
15      "idPedido": 7,
16      "idProducto": 6,
17      "cantidad": 20,
18      "producto": {
19        "idProducto": 6,
20        "nombre": "Manzana verdes2",
21        "detalle": "Manza de Sandona",
22        "precio": 5000
23      }
24    }
25  }
26 ]
```

Get con ID

Fruver / Pedidos

GET http://localhost:3000/pedidos/7 Send 200 OK 5.33 ms 64 B

Body Auth Query Headers Docs Preview Headers Cookies Timeline

```
1 {
2   "idPedido": 7,
3   "idCliente": 5,
4   "fechaSolicitud": null,
5   "estado": null
6 }
```

Post

Fruver / Pedidos

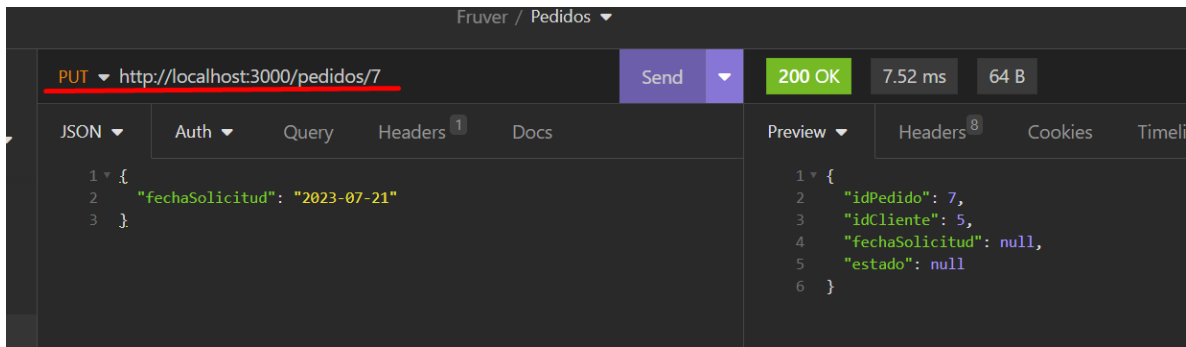
POST http://localhost:3000/pedidos Send 200 OK 114 ms 30 B

JSON Auth Query Headers Docs Preview Headers Cookies Timeline

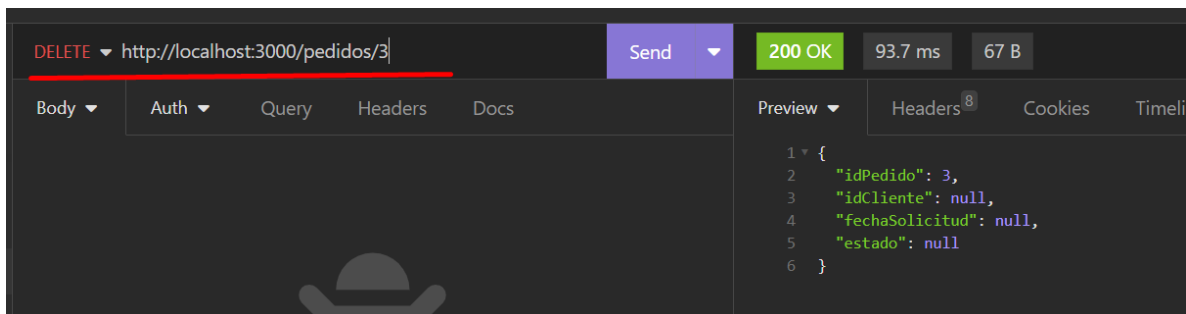
```
1 {
2   "idCliente": "5",
3   "fechaSolicitud": "2023-07-21",
4   "estado": 0
5 }
```

```
1 {
2   "idPedido": 7,
3   "idCliente": "5"
4 }
```

Put

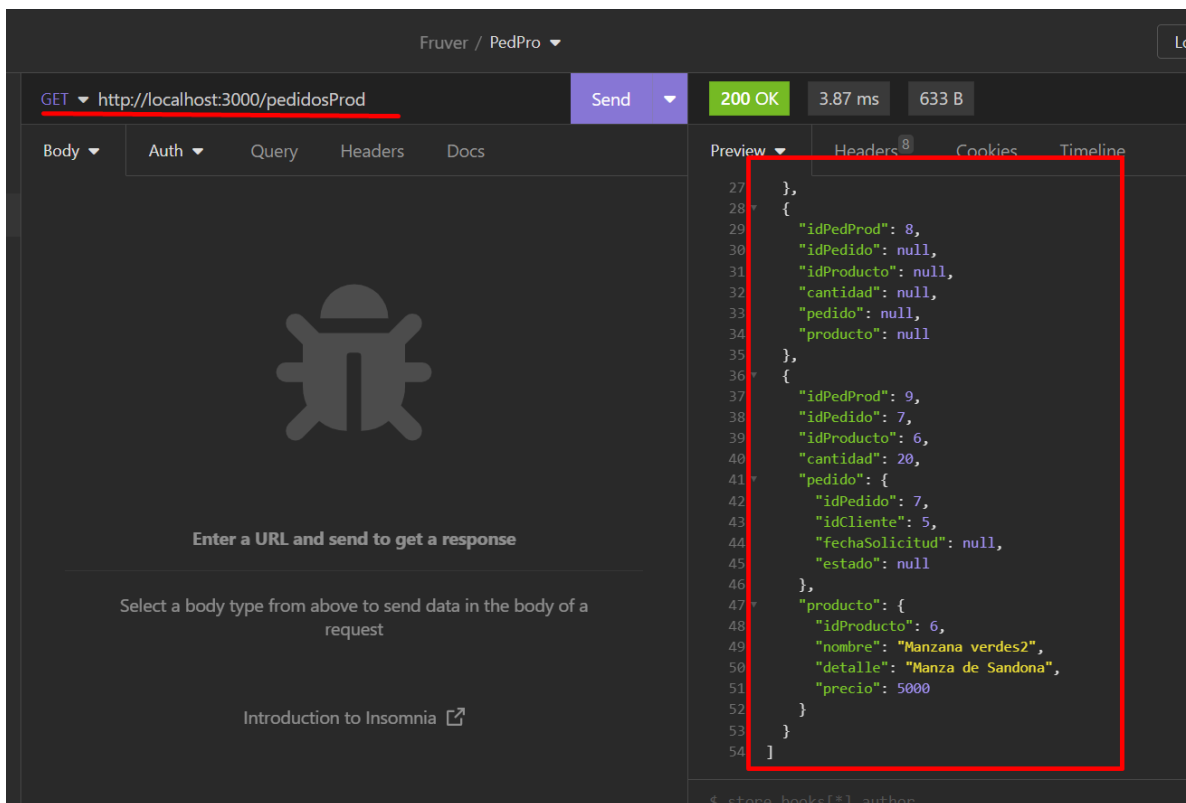


Delete



Consultas Pedido productos

Get



Post

Fruver / PedPro ▼

POST ▼ http://localhost:3000/pedidosProd Send ▼ 200 OK 112 ms 57 B

JSON ▼ Auth ▼ Query Headers 1 Docs Preview ▼ Headers 8 Cookies

```
1 {
2   "idPedido": 7,
3   "idProducto": 6,
4   "cantidad": 20
5 }
6
```

```
1 {
2   "idPedProd": 9,
3   "idPedido": 7,
4   "idProducto": 6,
5   "cantidad": 20
6 }
```

Put

Fruver / PedPro ▼

PUT ▼ http://localhost:3000/pedidosProd/1 Send ▼ 200 OK 6.38 ms 60 B

JSON ▼ Auth ▼ Query Headers 1 Docs Preview ▼ Headers 8 Cookies

```
1 {
2   "idProducto": 8
3 }
```

```
1 {
2   "idPedProd": 1,
3   "idPedido": 1,
4   "idProducto": null,
5   "cantidad": 10
6 }
```

Delete

Fruver / PedPro ▼

DELETE ▼ http://localhost:3000/pedidosProd/5 Send ▼ 200 OK 96.6 ms 62 B

Body ▼ Auth ▼ Query Headers Docs Preview ▼ Headers 8 Cookies

```
1 {
2   "idPedProd": 5,
3   "idPedido": 5,
4   "idProducto": null,
5   "cantidad": null
6 }
```