

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

Taller 2 Unidad Backend

Autor

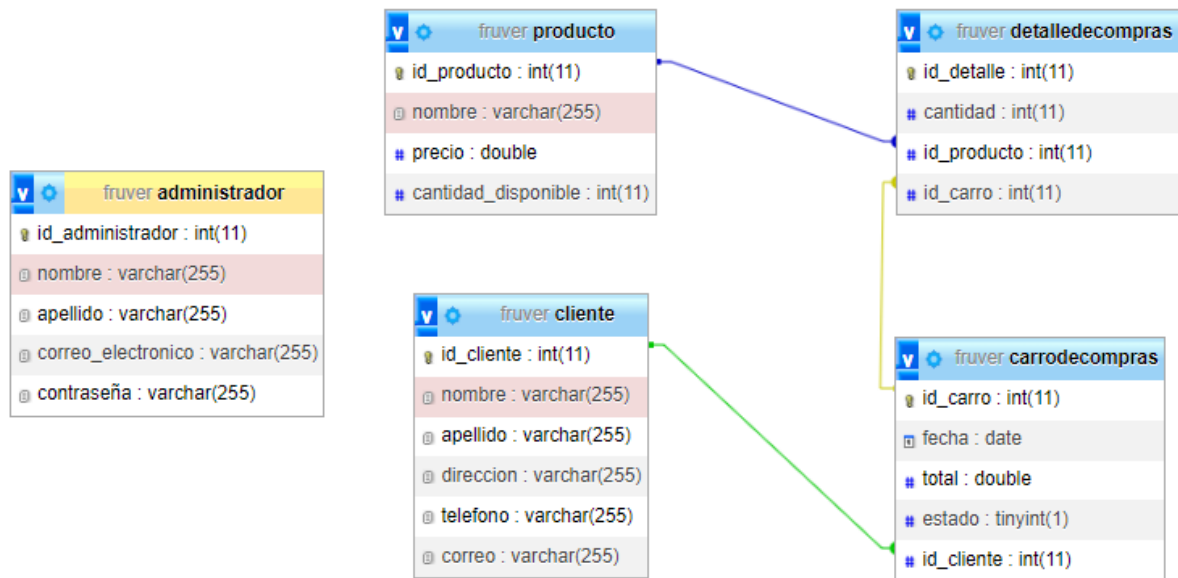
MAFLA ANDRES

Profesor

VICENTE AUX REVELO

**DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE NARIÑO
JUNIO, 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.



Para la construcción de la base de datos se contó con la creación de 5 tablas las cuales son: producto, cliente, administrador, carro de compras y detalle de compra.

La tabla "Producto" contiene información detallada sobre los productos, como su nombre, descripción, precio y cantidad disponible. En la tabla "Cliente" se almacena información relacionada con los clientes, como su nombre, apellido, dirección, número de teléfono, correo electrónico y contraseña. Por otro lado, la tabla "Administrador" guarda los datos de los administradores, como su nombre, apellido, correo electrónico y contraseña.

La tabla "carrodecompras" se utiliza para registrar los carros de compras de los clientes. En esta tabla se incluye una llave foránea con el id del cliente correspondiente y la fecha en que se creó el carrito. Finalmente, la tabla "detalledecompra" registra los productos agregados a un carrito de compras específico, incluyendo la cantidad de cada producto. Esta tabla tiene una llave foránea a las tablas "carrodecompras" y "Producto".

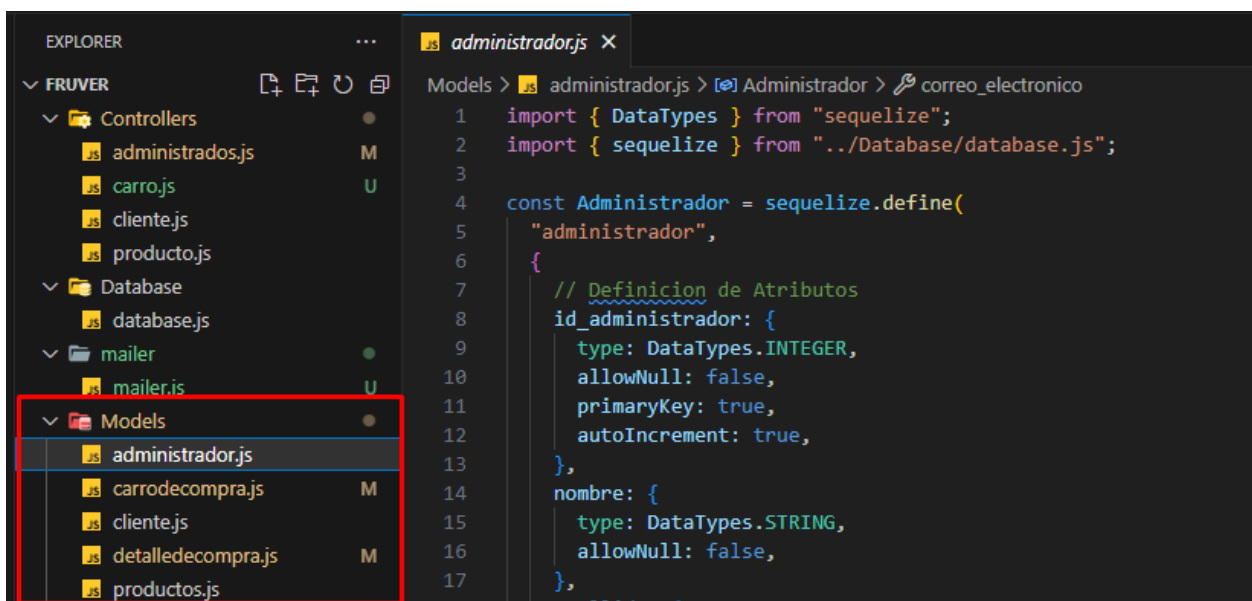
2.Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS que haga uso de la base de datos del primer punto y que permita el desarrollo de todas las tareas asociadas al registro y administración de las frutas y verduras (La empresa debe contar con un nombre). Se debe hacer uso correcto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

2.1 Se crea la carpeta “Database” la cual contiene el archivo “database.js” dentro de este se encuentra la conexión a la base de datos.



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

2.2 Se crea la carpeta “Models” dentro de esta se encuentran diferentes archivos los cuales van a formar nuestra base de datos.



```
1 import { DataTypes } from "sequelize";
2 import { sequelize } from "../Database/database.js";
3
4 const Administrador = sequelize.define(
5   "administrador",
6   {
7     // Definicion de Atributos
8     id_administrador: {
9       type: DataTypes.INTEGER,
10      allowNull: false,
11      primaryKey: true,
12      autoIncrement: true,
13    },
14     nombre: {
15       type: DataTypes.STRING,
16       allowNull: false,
17     },
18     apellido: {
```

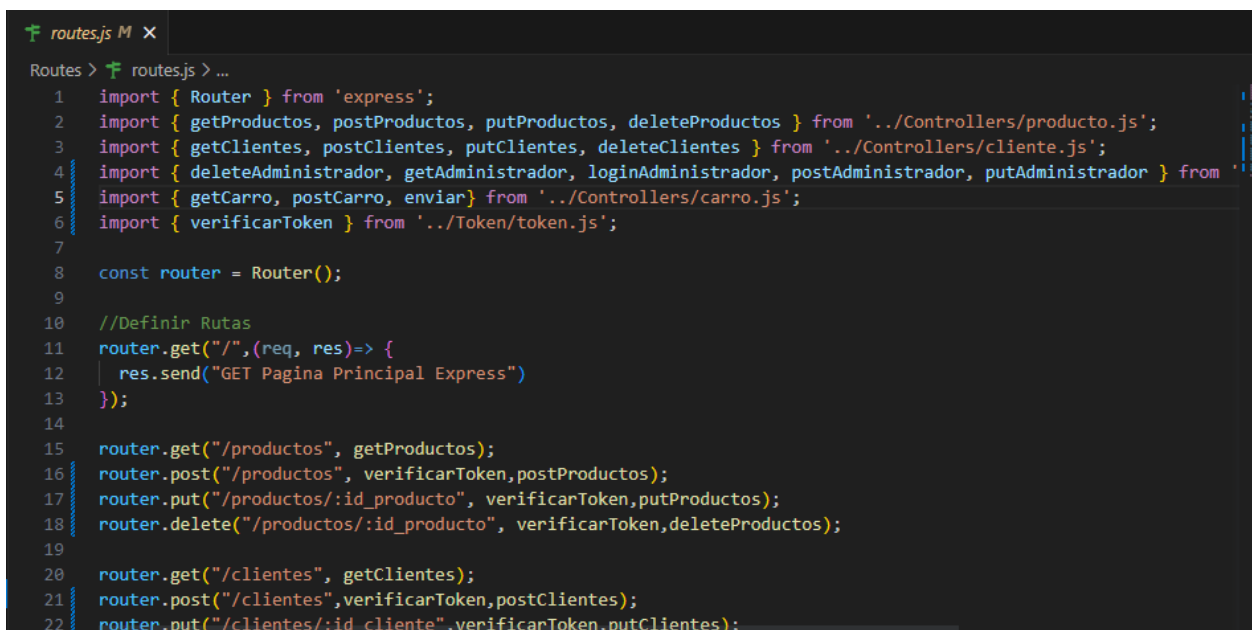
2.3 Se creo la carpeta “Controllers” dentro de esta se almacenan archivos que contienen la lógica del manejo de las solicitudes HTTP.



```
EXPLORER
FRUVER
  Controllers
    administrados.js M
    carro.js U
    cliente.js
    producto.js
  Database
    database.js
  mailer
    mailer.js U
  Models
    administrador.js
    carrodecompra.js M
    cliente.js
    detalladecompra.js M
    productos.js
  node_modules
  Routes
    routes.js M
  Token

administrados.js M
1 import { Administrador } from "../Models/administrador.js";
2 import jwt from 'jsonwebtoken';
3
4 //manejo de clientes
5
6 const loginAdministrador = async (req, res) => {
7   const admin = req.body;
8   try{
9     const admin2 = await Administrador.findOne({
10       where: {
11         correo_electronico: admin.correo_electronico
12       }
13     });
14     if (admin2){
15       if (admin2.contraseña != admin.contraseña ){
16         return res.status(401).json({message: "Datos incorrectos"});
17       }else if(admin2.contraseña == admin.contraseña){
18         const datos = { correo_electronico: admin2.correo_electronico, nombre: admin2.nombre};
19         const token = jwt.sign(datos, "SECRET", {expiresIn: '1h'})
20         return res.status(200).json({token: token});
21       }else{
22         return res.status(400).json({message: "Error"});
```

2.4 Se creo la carpeta “Routes” en la cual esta el archivo “routes.js” en el cual se encuentran las distintas rutas que permiten manejar los endpoints para recibir solicitudes HTTP y devolver respuestas.



```
routes.js M
Routes > routes.js > ...
1 import { Router } from 'express';
2 import { getProductos, postProductos, putProductos, deleteProductos } from '../Controllers/producto.js';
3 import { getCientes, postCientes, putCientes, deleteCientes } from '../Controllers/cliente.js';
4 import { deleteAdministrador, getAdministrador, loginAdministrador, postAdministrador, putAdministrador } from '../Controllers/administrador.js';
5 import { getCarro, postCarro, enviar } from '../Controllers/carro.js';
6 import { verificarToken } from '../Token/token.js';
7
8 const router = Router();
9
10 //Definir Rutas
11 router.get("/",(req, res)=> {
12   res.send("GET Pagina Principal Express")
13 });
14
15 router.get("/productos", getProductos);
16 router.post("/productos", verificarToken,postProductos);
17 router.put("/productos/:id_producto", verificarToken,putProductos);
18 router.delete("/productos/:id_producto", verificarToken,deleteProductos);
19
20 router.get("/clientes", getCientes);
21 router.post("/clientes",verificarToken,postCientes);
22 router.put("/clientes/:id_cliente",verificarToken,putCientes);
```

2.5 Se creo la carpeta “Token” la cual cuenta con la función “VerificarToken” con esto podremos autenticar la solicitud permitiendo pasar a la siguiente función.

```
token.js U X
Token > token.js > verificarToken > jwt.verify("SECRET") callback
1 import jwt from 'jsonwebtoken';
2
3 function verificarToken(req, res, next){
4   const bearerHeader = req.headers['authorization'];
5   if (!bearerHeader){
6     return res.status(401).json("No autorizado");
7   }
8
9   const bearerToken = bearerHeader.split(" ")[1];
10  jwt.verify(bearerToken, "SECRET", (err, response) => {
11    if (err){
12      return res.sendStatus(403);
13    }else{
14      res.locals = response;
15      next();
16    }
17  });
18 }
19
20 export {verificarToken};
```

2.6 Se creo la carpeta “Mailer” aquí se encuentra en archivo “mailer.js” el cual permite enviar correos electrónicos.

```
token.js U mailer.js U X
mailer > mailer.js > transporter > auth > pass
1 import nodemailer from 'nodemailer';
2
3 const transporter = nodemailer.createTransport({
4   host: 'smtp.gmail.com',
5   port: 587,
6   secure: true,
7   auth: {
8     user: 'surtifruver977@gmail.com',
9     pass: 'ncefuoppwdgmntwv',
10  },
11 });
12
13 transporter.verify().then(() => {
14   console.log('Listo para enviar emails');
15 });
16
17 export {transporter};
18
```

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

Solicitudes realizadas de administradores.

1. Registrar administradores.

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/administrador`. The request body is a JSON object with the following fields:

```
{  "nombre": "carlos",  "apellido": "leon",  "correo_electronico": "amafila572@gmail.com",  "contraseña": "admin"}
```

The response is a JSON object with the following fields:

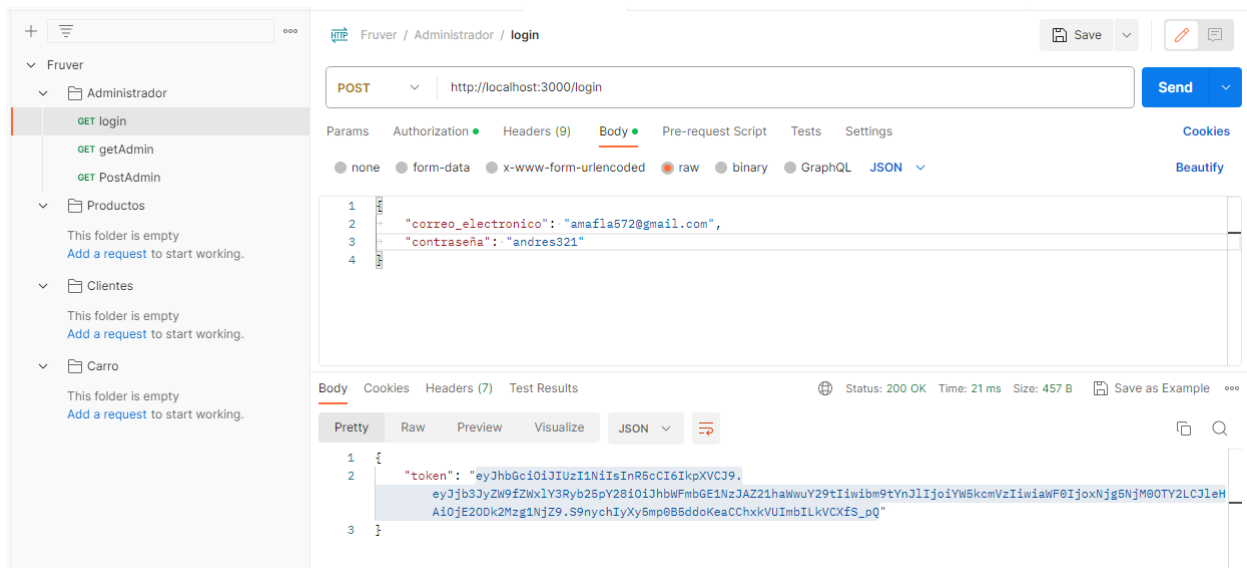
```
{  "id_administrador": 3,  "nombre": "carlos",  "apellido": "leon",  "correo_electronico": "amafila572@gmail.com",  "contraseña": "admin"}
```

Below the Postman interface, there is a table with the following data:

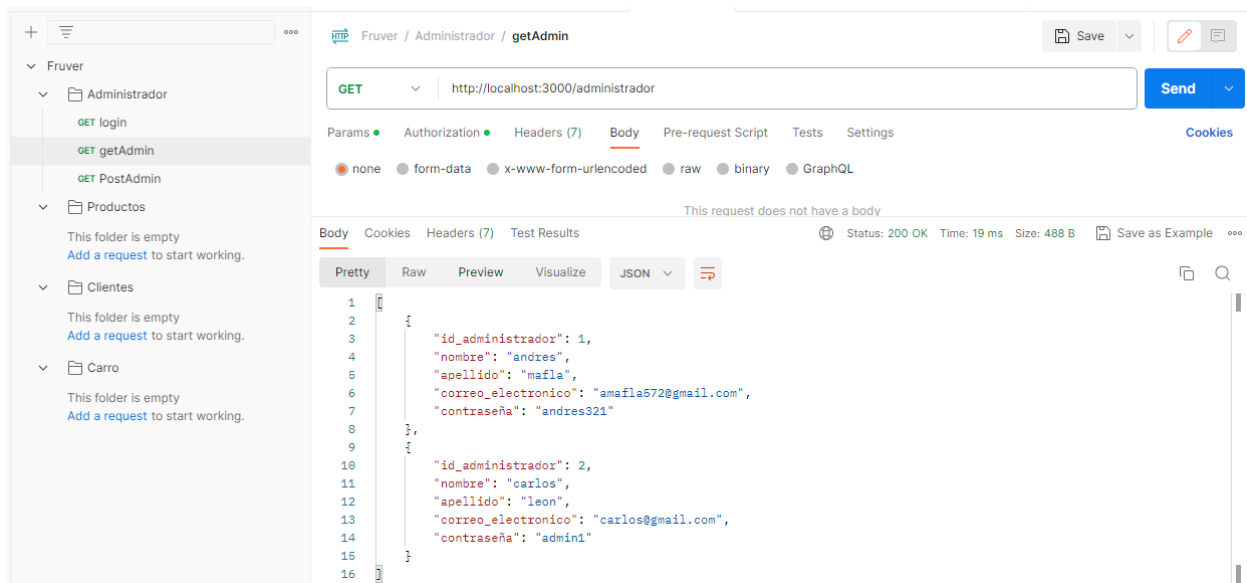
	id_administrador	nombre	apellido	correo_electronico	contraseña
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	andres	mafla	amafila572@gmail.com	andres321
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	carlos	leon	carlos@gmail.com	admin1

At the bottom, there is a section for "Opciones extra" with a "Seleccionar todo" checkbox and a "Para los elementos que están marcados:" section with "Editar", "Copiar", "Borrar", and "Exportar" options.

2. Login.



3. Visualizar administradores.



4. Editar administradores.

The screenshot shows the Fruver API client interface. On the left, a sidebar lists the API structure: Fruver > Administrador > PUT Admin. The main panel shows a PUT request to `http://localhost:3000/administrador/1`. The request body is a JSON object: `{ "nombre": "andres", "apellido": "admin", "correo_electronico": "amafla572@gmail.com", "contraseña": "admin" }`. The response status is 200 OK, and the response body is a JSON object: `{ "id_administrador": 1, "nombre": "andres", "apellido": "admin", "correo_electronico": "amafla572@gmail.com", "contraseña": "admin" }`.

5. Eliminar administradores.

The screenshot shows the Fruver API client interface. On the left, a sidebar lists the API structure: Fruver > Administrador > DELETE Admin. The main panel shows a DELETE request to `http://localhost:3000/administrador/1`. The request is authenticated with a Bearer token: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1IjoiMSIsImN1b250IjoiYWRhcm91IiwiaWF0IjoiMTYxMjM0MjM0In0.`. The response status is 200 OK, and the response body is a JSON object: `{ "mensaje": "Registro Eliminado" }`.

	id_administrador	nombre	apellido	correo_electronico	contraseña
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	carlos	leon	carlos@gmail.com	admin1

Solicitudes realizadas de productos.

1. Registrar productos.

The screenshot shows a REST client interface with a sidebar on the left containing a folder tree: **Fruver** (expanded), **Administrador**, **Productos** (expanded), **GET PostProductos**, **Cientes** (with a note "This folder is empty. Add a request to start working."), and **Carro** (with a note "This folder is empty. Add a request to start working.>"). The main panel displays a **POST** request to `http://localhost:3000/productos. The Body tab is selected, showing a JSON payload:

```
{  "nombre": "kiwi",  "precio": "2000",  "cantidad_disponible": "10"}
```

. The response status is 200 OK with a time of 59 ms and size of 311 B. The Body tab is also selected for the response, showing a JSON payload:

```
{  "id_producto": 1,  "nombre": "kiwi",  "precio": "2000",  "cantidad_disponible": "10"}
````

☐ **Mostrar todo** | **Número de filas:** 25 **Filtrar filas:**

Opciones extra

	id_producto	nombre	precio	cantidad_disponible
<input type="checkbox"/> Editar Copiar Borrar	1	kiwi	2000	10

2. Visualizar productos.

The screenshot shows a REST client interface with a sidebar on the left containing a tree view with folders like 'Fruver', 'Administrador', 'Productos', 'Clientes', and 'Carro'. The main panel displays a 'New Request' for a GET method to 'http://localhost:3000/productos'. The 'Authorization' tab is active, showing a Bearer token. The response body is displayed in 'Pretty' format, showing a JSON array of two products.

```
1 {
2   {
3     "id_producto": 1,
4     "nombre": "kiwi",
5     "precio": 2000,
6     "cantidad_disponible": 10
7   },
8   {
9     "id_producto": 2,
10    "nombre": "pera",
11    "precio": 1000,
12    "cantidad_disponible": 30
13  }
14 }
```

		id_producto	nombre	precio	cantidad_disponible
<input type="checkbox"/>	Editar Copiar Borrar	1	kiwi	2000	10
<input type="checkbox"/>	Editar Copiar Borrar	2	pera	1000	30

3. Editar productos.

The screenshot shows the Fruver API client interface. On the left is a sidebar with a tree view containing folders like 'Administrador', 'Productos', 'Clientes', and 'Carro'. The 'Productos' folder is expanded, showing 'GET PostProductos' and 'GET New Request'. The main panel is titled 'Fruver / Productos / New Request'. It shows a PUT request to 'http://localhost:3000/productos/1'. The 'Body' tab is selected, displaying a JSON payload:

```
{  "nombre": "kiwi modificado",  "precio": "1000",  "cantidad_disponible": "100"}
```

. Below the request, the 'Response' tab shows the status '200 OK' and a JSON response:

```
{  "id_producto": 1,  "nombre": "kiwi modificado",  "precio": "1000",  "cantidad_disponible": "100"}
```

				id_producto	nombre	precio	cantidad_disponible
<input type="checkbox"/>		Editar		Copiar		Borrar	1 kiwi modificado1000100
<input type="checkbox"/>		Editar		Copiar		Borrar	2 pera100030

4. Eliminar productos.

Fruver / Productos / New Request

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

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer T...
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Body Cookies Headers (7) Test Results Status: 200 OK Time: 40 ms Size: 267 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "mensaje": "Registro Eliminado"  
3 }
```

	id_producto	nombre	precio	cantidad_disponible
<input type="checkbox"/> Editar <input type="image"/> Copiar <input type="image"/> Borrar	2	pera	1000	30

Solicitudes realizadas de clientes.

1. Registrar clientes.

The screenshot shows a REST client interface with a sidebar on the left containing a tree view of folders: Fruver, Administrador, Productos, Clientes, and Carro. The 'Clientes' folder is expanded, showing a 'GET PostCliente' endpoint. The main area displays a POST request to 'http://localhost:3000/clientes'. The request body is a JSON object with the following fields: 'nombre' (carlos), 'apellido' (leon), 'direccion' (calle 21), 'telefono' (3146754324), and 'correo' (carlos@gmail.com). The response body is also shown in JSON format, with an additional 'id_cliente' field set to 1. The status bar at the bottom indicates a 200 OK response with a time of 86 ms and a size of 363 B.

```
POST http://localhost:3000/clientes

{
  "nombre": "carlos",
  "apellido": "leon",
  "direccion": "calle 21",
  "telefono": "3146754324",
  "correo": "carlos@gmail.com"
}
```

Body Cookies Headers (7) Test Results

```

{
  "id_cliente": 1,
  "nombre": "carlos",
  "apellido": "leon",
  "direccion": "calle 21",
  "telefono": "3146754324",
  "correo": "carlos@gmail.com"
}
```

Status: 200 OK Time: 86 ms Size: 363 B Save as Example

	id_cliente	nombre	apellido	direccion	telefono	correo
	1	carlos	leon	calle 21	3146754324	carlos@gmail.com

2. Visualizar clientes.

The screenshot shows the Fruver API client interface. On the left, a sidebar lists the project structure: Fruver, Administrador, Productos, Clientes, and Carro. The 'Clientes' folder is expanded, showing 'GET PostCliente' and 'GET GetCliente'. The main panel displays a GET request to 'http://localhost:3000/clientes'. The 'Authorization' tab is active, showing a Bearer Token. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The response body is shown in JSON format:

```
1 {
2   "id_cliente": 1,
3   "nombre": "carlos",
4   "apellido": "leon",
5   "direccion": "calle 21",
6   "telefono": "3146754324",
7   "correo": "carlos@gmail.com"
8 }
```

	id_cliente	nombre	apellido	direccion	telefono	correo
	1	carlos	leon	calle 21	3146754324	carlos@gmail.com

3. Editar clientes.

The screenshot displays a REST client interface with a sidebar on the left showing a project structure: Fruver > Clientes > GET PutCliente. The main area shows a PUT request to `http://localhost:3000/clientes/1`. The request body is a JSON object:

```
{  "nombre": "carlos nuevo",  "apellido": "leon fiera",  "direccion": "calle 21",  "telefono": "3146754324",  "correo": "amafla572@gmail.com"}
```

The response status is 200 OK. The response body is also shown in JSON format:

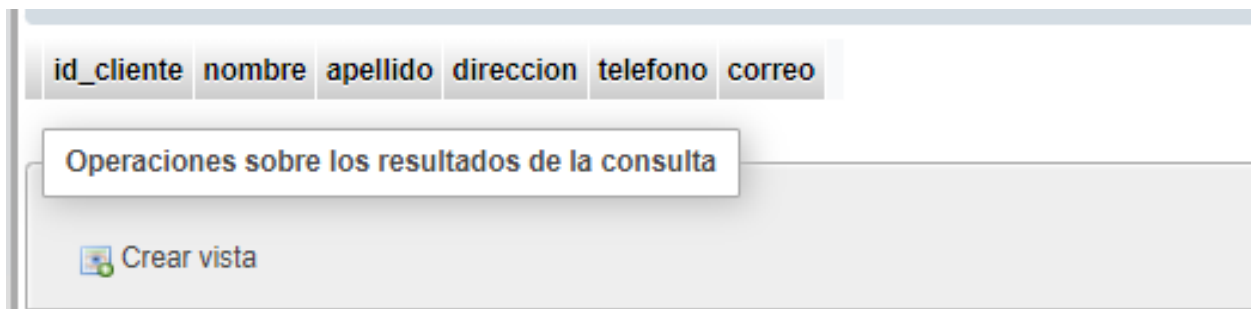
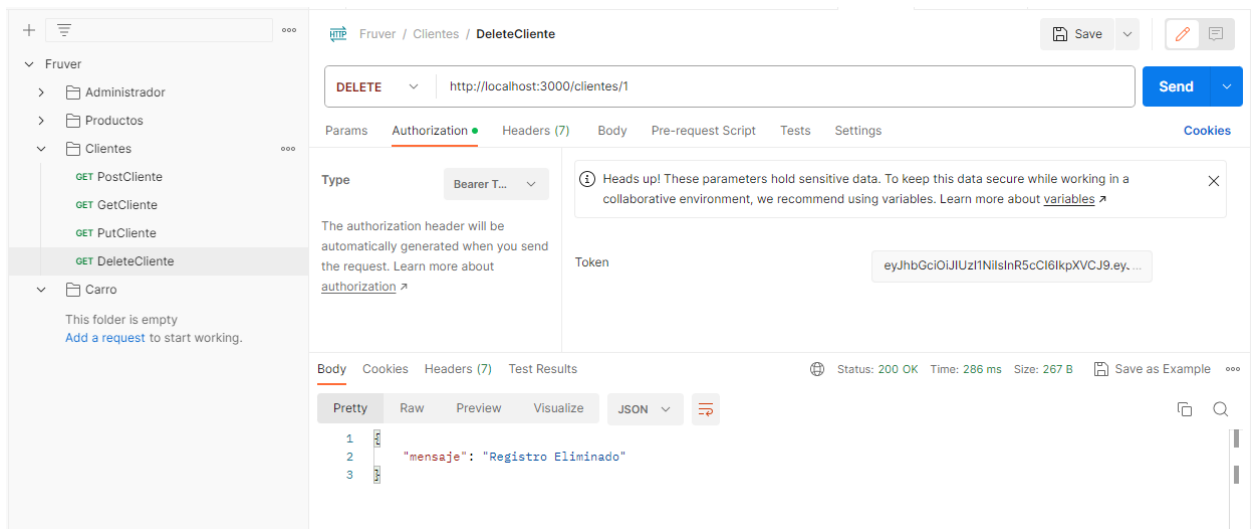
```
{  "id_cliente": 1,  "nombre": "carlos nuevo",  "apellido": "leon fiera",  "direccion": "calle 21",  "telefono": "3146754324",  "correo": "amafla572@gmail.com"}
```

Below the JSON view, a table displays the client data:

	id_cliente	nombre	apellido	direccion	telefono	correo
1	carlos nuevo	leon fiera	calle 21	3146754324	amafla572@gmail.com	

At the bottom, there are action buttons: Editar, Copiar, and Borrar.

4. Eliminar clientes.



Solicitudes realizadas de carro de compras.

1. Registrar compra.

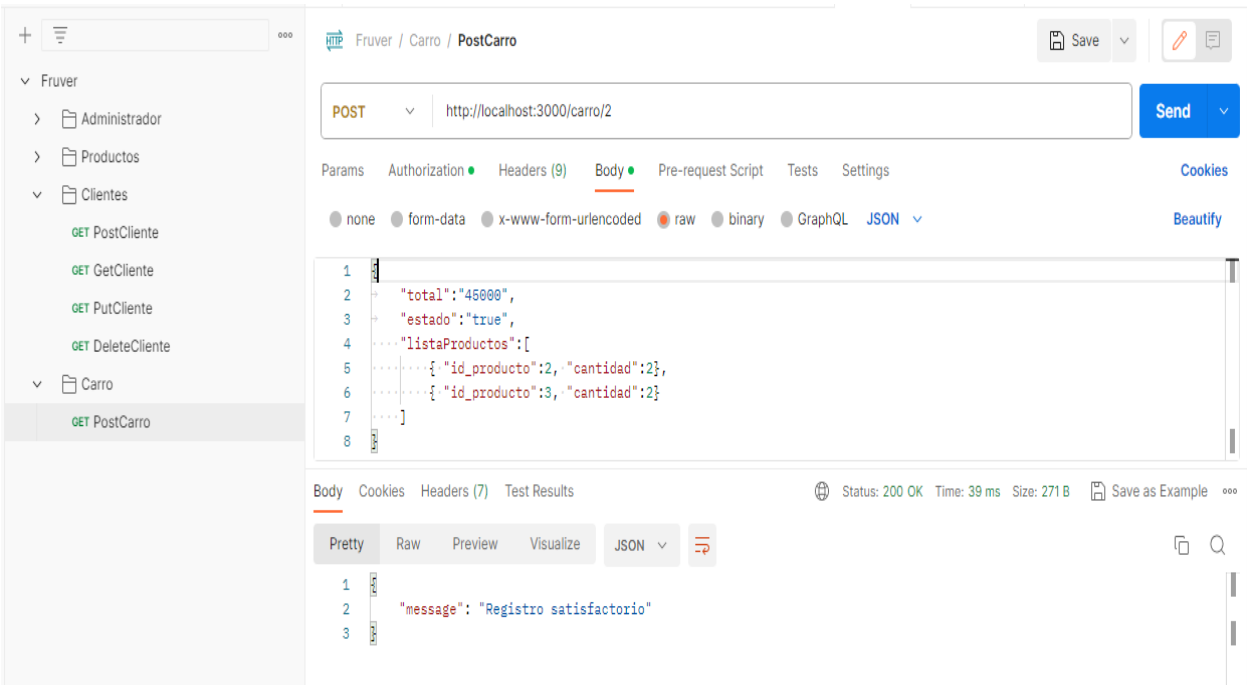








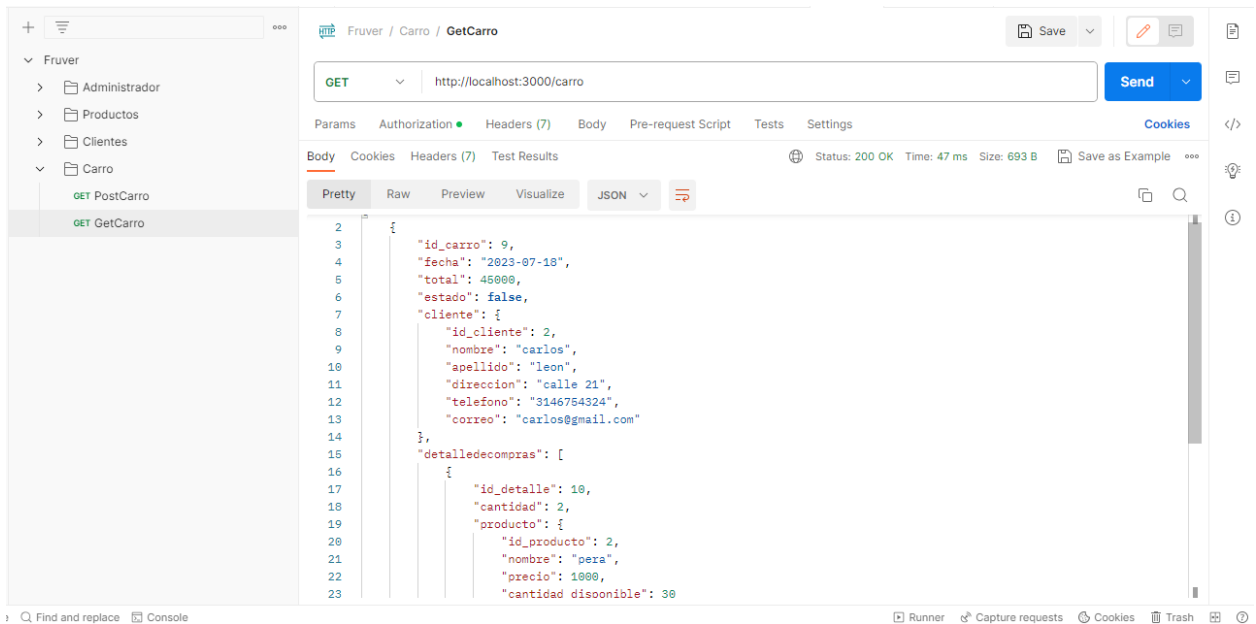
Tabla carro de compras

<div>↩️ ⬆️ ⬇️ ⬆️ ⬅️</div>				id_carro	fecha	total	estado	id_cliente
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	9	2023-07-18	45000	0	2

Tabla detalle de compras

<div>← T →</div>					id_detalle	cantidad	id_producto	id_carro
<input type="checkbox"/>	 Editar	 Copiar	 Borrar		10	2	2	9
<input type="checkbox"/>	 Editar	 Copiar	 Borrar		11	2	3	9

2. Visualizar compras.



GET http://localhost:3000/carro

Status: 200 OK Time: 47 ms Size: 693 B

```
{
  "id_carro": 9,
  "fecha": "2023-07-18",
  "total": 45000,
  "estado": false,
  "cliente": {
    "id_cliente": 2,
    "nombre": "carlos",
    "apellido": "leon",
    "direccion": "calle 21",
    "telefono": "3146754324",
    "correo": "carlos@gmail.com"
  },
  "detalledecompras": [
    {
      "id_detalle": 10,
      "cantidad": 2,
      "producto": {
        "id_producto": 2,
        "nombre": "pera",
        "precio": 1000,
        "cantidad_disponible": 30
      }
    }
  ]
}
```

	id_cliente	nombre	apellido	direccion	telefono	correo
<input type="checkbox"/> Editar Copiar Borrar	2	carlos	leon	calle 21	3146754324	carlos@gmail.com

	id_carro	fecha	total	estado	id_cliente
<input type="checkbox"/> Editar Copiar Borrar	9	2023-07-18	45000	0	2

	id_detalle	cantidad	id_producto	id_carro
<input type="checkbox"/> Editar Copiar Borrar	10	2	2	9
<input type="checkbox"/> Editar Copiar Borrar	11	2	3	9

3. Enviar compra.

API client interface showing a PUT request to `http://localhost:3000/carro`. The request body is a JSON object:


```
{  "id_carro": 15}
```

The response status is 200 OK. The response body is a JSON object:

```
{  "id_carro": 15,  "fecha": "2023-07-18",  "total": 45000,  "estado": true,  "id_cliente": 2}
```

				id_carro	fecha	total	estado	id_cliente
<input type="checkbox"/>	Editar	Copiar	Borrar	9	2023-07-18	45000	1	2
<input type="checkbox"/>	Editar	Copiar	Borrar	10	2023-07-18	45000	1	2
<input type="checkbox"/>	Editar	Copiar	Borrar	11	2023-07-18	45000	1	2
<input type="checkbox"/>	Editar	Copiar	Borrar	12	2023-07-18	45000	1	2
<input type="checkbox"/>	Editar	Copiar	Borrar	13	2023-07-18	45000	1	2
<input type="checkbox"/>	Editar	Copiar	Borrar	14	2023-07-18	45000	1	2
<input type="checkbox"/>	Editar	Copiar	Borrar	15	2023-07-18	45000	1	2

Estado de compra actualizado

 **surtifruver977@gmail.com**
para mí ▾

para mí ▼

21:00 (hace 1 minuto)

El estado de tu compra con ID 15 ha sido actualizado a activo.

← Responder

→ Reenviar