Saludo, presento documentación del servicio de backend y su funcionamiento.

Instalación y Arranque

Ejecutar instalación de dependencias node.js.

npm install

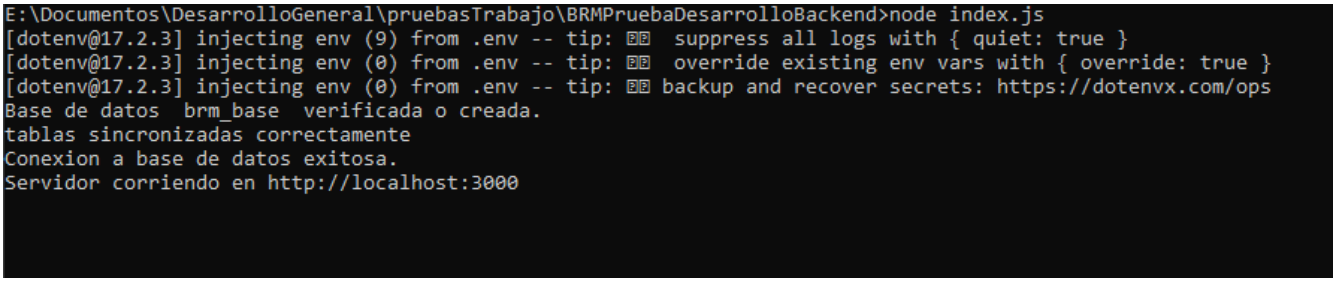
Este proyecto tienes sus configuraciones en un archivo env. Esta programado para conectarse a una base de datos mysql, en mis pruebas locales lo tengo configurado de la siguiente manera.

```
PORT=3000
DB_NAME=brm_base
DB_USER=root
DB_PASS=
DB_HOST=localhost

USERADMIN_LOGIN=root
USERADMIN_PASS=$2b$10$xldHBD7XRph8Us7aH/fB1e8thBSeJyYkaL2sGAPTfaPN4Kqo38YH2

DEVCLIENT_LOGIN=dev_client
USERADMIN_PASS=$2b$10$xldHBD7XRph8Us7aH/fB1e8thBSeJyYkaL2sGAPTfaPN4Kqo38YH2

JWT_SECRET=un_token_mas_potente_que_el_del_codigo_quemado_034jrfwerjodfjvas09dfj
```

Las contraseñas encriptadas aquí son para usuarios que vienen por defecto, estas contraseñas hacen referencia a “test1234”, para un usuario administrador y un usuario cliente, posteriormente con los servicios montados se pueden crear mas usuarios, editarlos o eliminarlos, ya que tienen un crud completo.

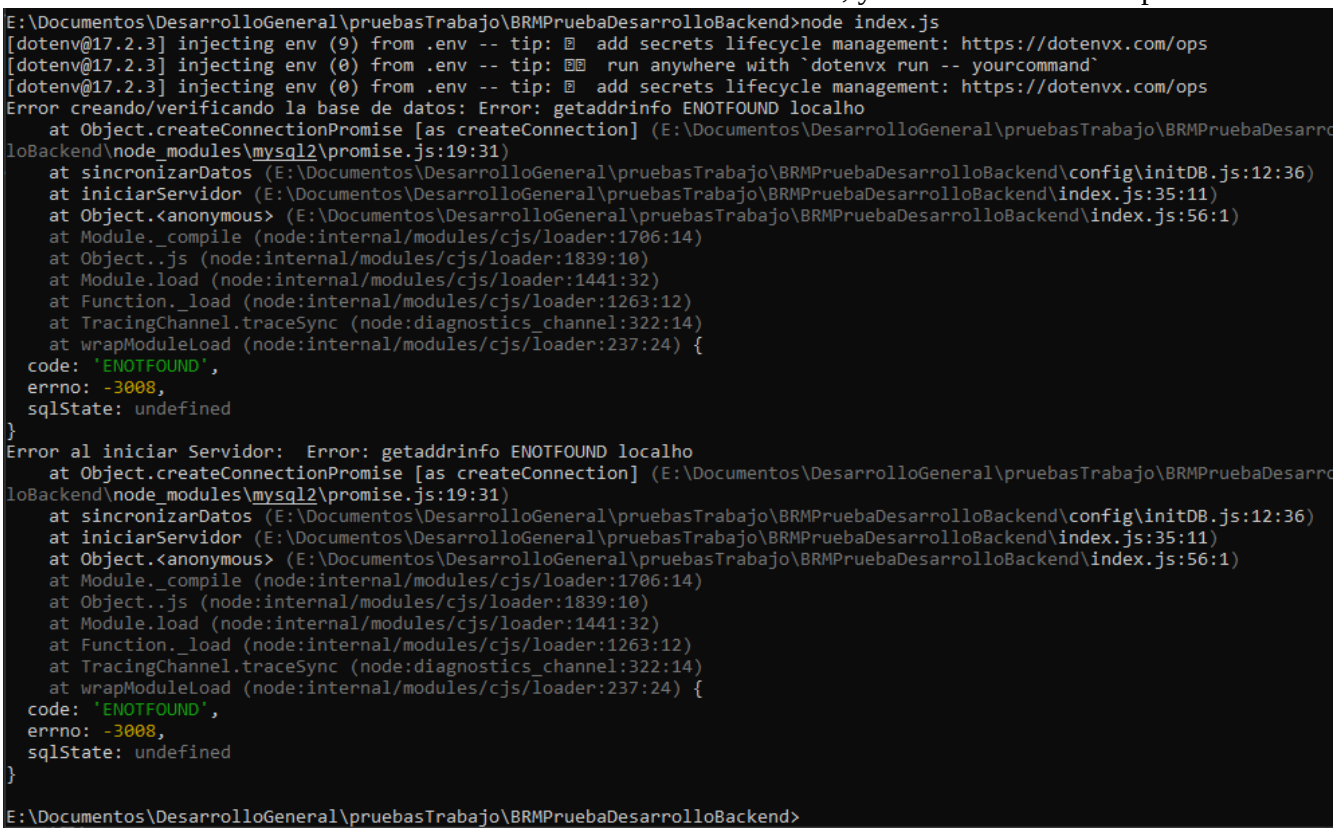
Para comenzar el proyecto se puede ejecutar desde una consola con node.js con los comandos node index.js o como desarrollo nodemon index.js, a mi gusto prefiero usar node index.js ya que me permite tener el control de lo que actualizo en el servidor.

Este proyecto no requiere configurar o ejecutar código sql directo en base de datos si no que al iniciar el servidor, y tener una conexión correcta con una base de datos crea una nueva base de datos y le asigna las tablas de los modelos sequelize configurados.



Así queda activo el servidor, y muestra como fue posible la conexión con la base de datos.

En caso de no tener una conexión correcta el servidor no iniciará, y mostrará el error en pantalla así.



En Cuanto a los servicios/endpoints se tiene varios y se dividen en secciones, igualmente también se monto un front pequeño de pruebas para que se puedan realizar de una manera grafica e intuitiva, los servicios end points están divididos de la siguiente manera.

Login

Este servicio es fundamental para la utilización de toda la api-rest ya que este permite iniciar sesión mediante un token, este permitirá validar al usuario y si es de tipo administrador o si es de tipo cliente, esto se define en la tabla rol_usuarios.

Result Grid | Fi

id	nombre
1	admin
2	cliente
NULL	NULL

Aquí se pueden ver los tipos de usuario que puede tener el servicio y que incluso es escalable ya que se pueden agregar mas tipos de roles.

Consumo del servicio Login: Post: http://localhost:3000/login

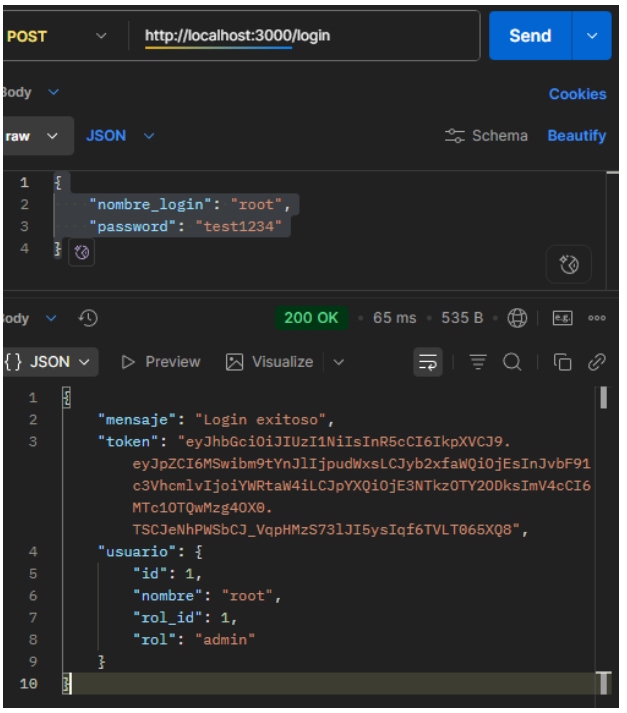
respuesta exitosa admin

json:

```
{
    "nombre_login": "root",
    "password": "test1234"
}
```

respuesta:

```
{
   "mensaje": "Login exitoso",
   "token": "eyJhbGciOiJIUzI1….XQ8",
   "usuario": {
      "id": 1,
      "nombre": "root",
      "rol_id": 1,
      "rol": "admin"
   }
}
```



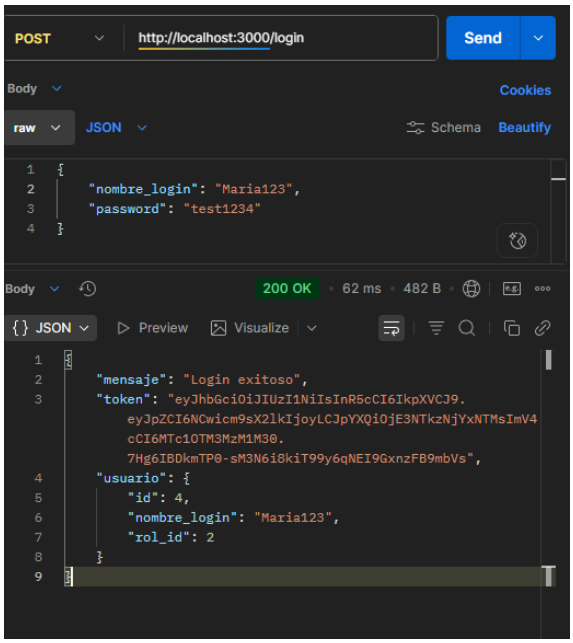
respuesta exitosa cliente http://localhost:3000/login

json:

```
{
   "nombre_login": "Maria123",
   "password": "test1234"
}
```

respuesta:

```
{
   "mensaje": "Login exitoso",
   "token": "eyJh...Q8",
   "usuario": {
      "id": 1,
      "nombre": "root",
      "rol_id": 1,
      "rol": "admin"
   }
}
```

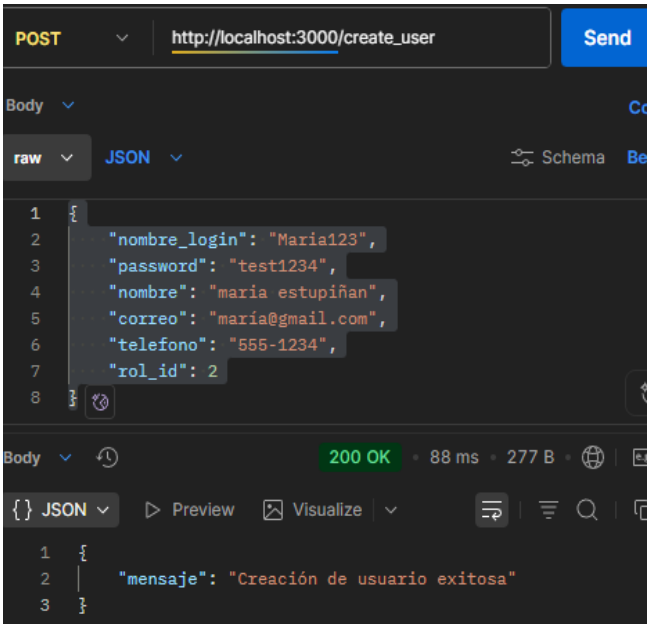


Crud usuarios

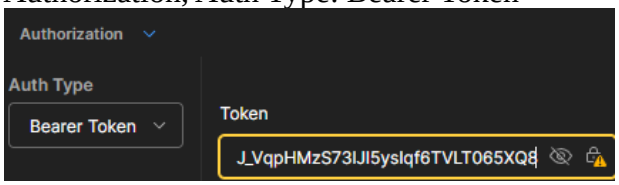
Servicios que permiten la creación, actualización, eliminación y consulta de usuarios, solo se le permite el ingreso a usuarios con rol administrador.

Creación: POST: http://localhost:3000/create_user
json:

```
{
   "nombre_login": "Maria123",
   "password": "test1234",
   "nombre": "maria estupiñan",
   "correo": "maría@gmail.com",
   "telefono": "555-1234",
   "rol_id": 2
},
```



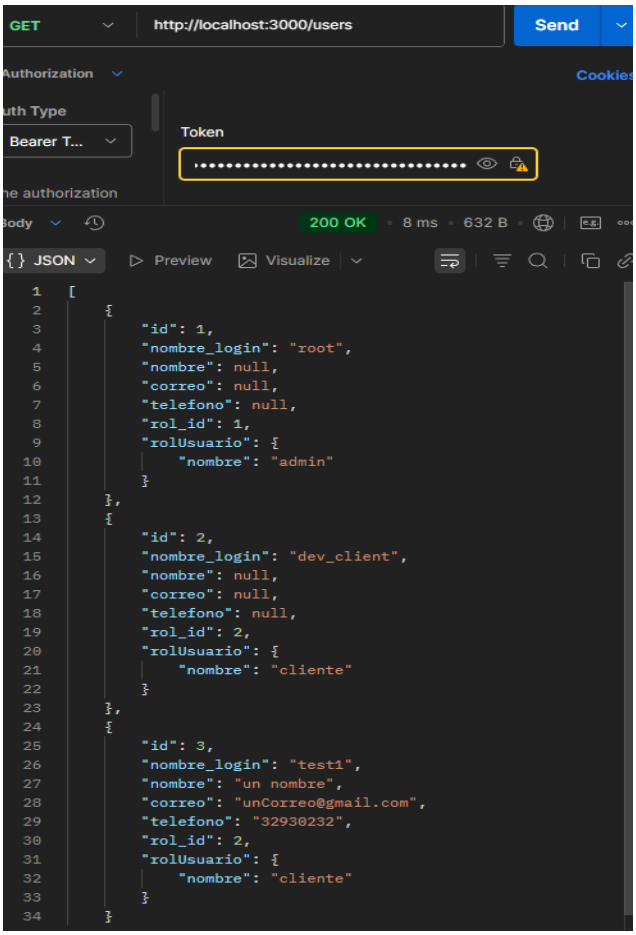
Debe ingresar el token el el parámetro Authorization, Auth Type: Bearer Token



respuesta exitosa:

```
{
   "mensaje": "Creación de usuario exitosa"
}
```

GetUsers: GET: http://localhost:3000/users



respuesta exitosa:

```
[
   {
      "id": 1,
      "nombre_login": "root",
      "nombre": null,
      "correo": null,
      "telefono": null,
      "rol_id": 1,
      "rolUsuario": {
         "nombre": "admin"
      }
   },
   {
      "id": 2,
      "nombre_login": "dev_client",
      "nombre": null,
      "correo": null,
      "telefono": null,
```

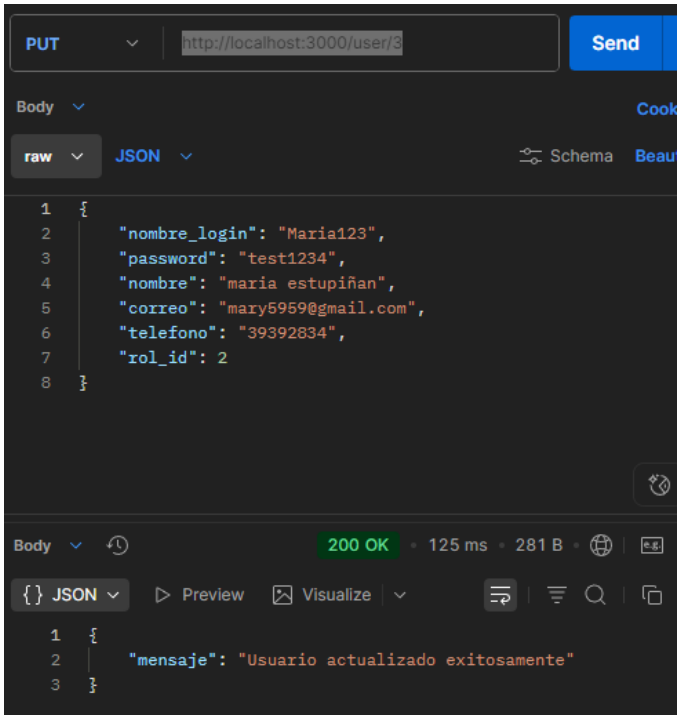
```
    "rol_id": 2,
    "rolUsuario": {
      "nombre": "cliente"
    }
  },
  {
    "id": 3,
    ...
  }
]
```

Actualización PUT: http://localhost:3000/user/3
json:

```
{
  "nombre_login": "Maria123",
  "password": "test1234",
  "nombre": "maria estupiñan",
  "correo": "mary5959@gmail.com",
  "telefono": "39392834",
  "rol_id": 2
}
```

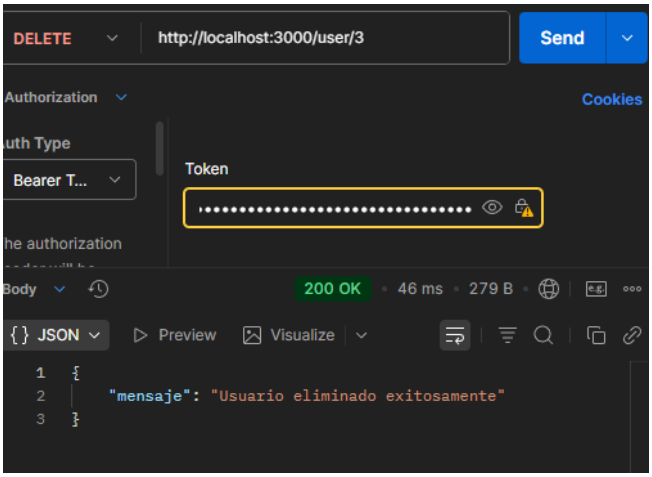
Requiere autentificación por token.
respuesta exitosa:

```
{
  "mensaje": "Usuario actualizado exitosamente"
}
```



Eliminación DELTE: http://localhost:3000/user/3
Requiere autentificación por token.
respuesta exitosa:

```
{
  "mensaje": "Usuario eliminado exitosamente"
}
```

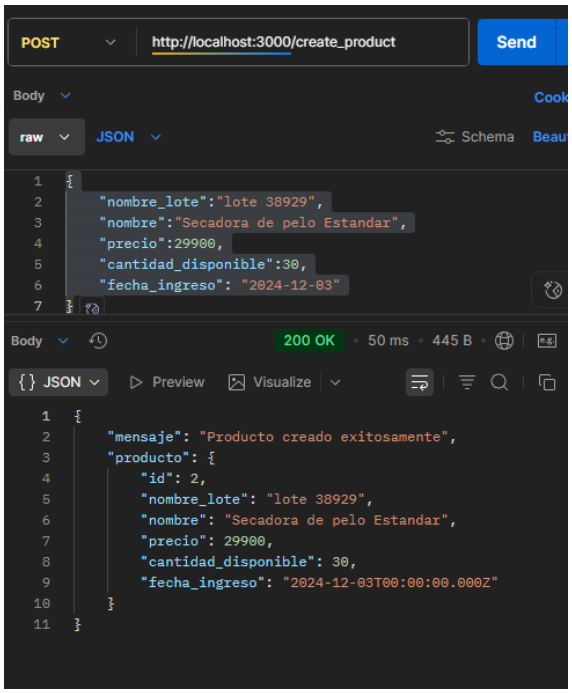


Crud Productos

Creación Producto POST: http://localhost:3000/create_product

Json:

```
{
   "nombre_lote":"lote 38929",
   "nombre":"Secadora de pelo Estandar",
   "precio":29900,
   "cantidad_disponible":30,
   "fecha_ingreso": "2024-12-03"
}
```



Requiere autentificación por token.

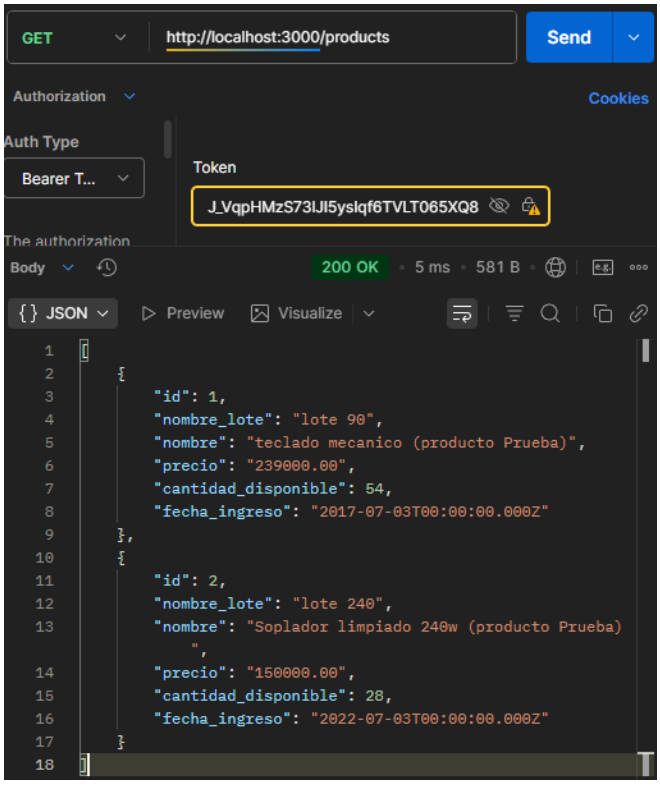
respuesta exitosa:

```
{
   "mensaje": "Producto creado exitosamente",
   "producto": {
      "id": 2,
      "nombre_lote": "lote 38929",
      "nombre": "Secadora de pelo Estandar",
      "precio": 29900,
      "cantidad_disponible": 30,
      "fecha_ingreso": "2024-12-03T00:00:00.000Z"
   }
}
```

Consultar Productos: GET: http://localhost:3000/products

Requiere autentificación por token.

Respuesta exitosa:

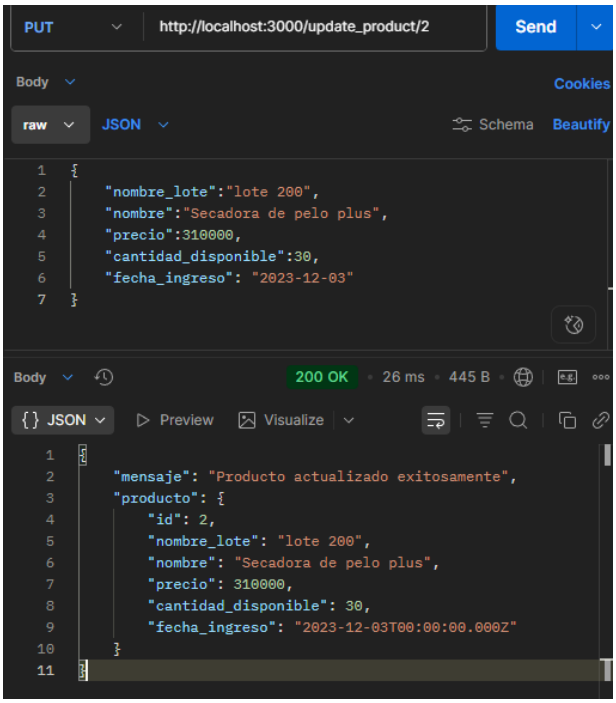



```
[
   {
      "id": 1,
      "nombre_lote": "lote 90",
      "nombre": "teclado mecanico (producto Prueba)",
      "precio": "239000.00",
      "cantidad_disponible": 54,
      "fecha_ingreso": "2017-07-03T00:00:00.000Z"
   },
   {
      "id": 2,
      "nombre_lote": "lote 240",
      "nombre": "Soplador limpiado 240w (producto Prueba)",
      "precio": "150000.00",
      "cantidad_disponible": 28,
      "fecha_ingreso": "2022-07-03T00:00:00.000Z"
   }
]
```

Actualización Producto PUT: http://localhost:3000/update_product/90

json: Requiere autentificación por token.

```
{
   "nombre_lote":"lote 200",
   "nombre":"Secadora de pelo plus",
   "precio":310000,
   "cantidad_disponible":30,
   "fecha_ingreso": "2023-12-03"
}
```



respuesta exitosa.

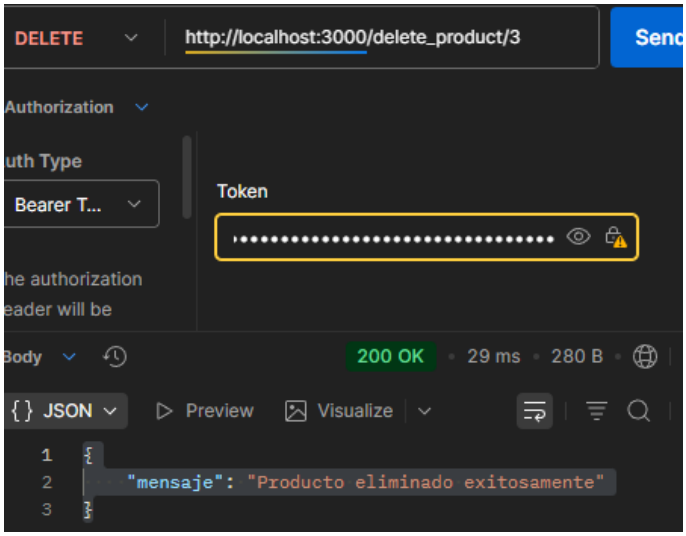
```
{
   "mensaje": "Producto actualizado exitosamente",
   "producto": {
      "id": 2,
      "nombre_lote": "lote 200",
      "nombre": "Secadora de pelo plus",
      "precio": 310000,
      "cantidad_disponible": 30,
      "fecha_ingreso": "2023-12-03T00:00:00.000Z"
   }
}
```

Eliminación Producto: DELETE: http://localhost:3000/delete_product/3

Requiere autentificación por token.

respuesta exitosa.

```
{
   "mensaje": "Producto eliminado exitosamente"
}
```



Compras

Agregar Producto Carrito: POST: http://localhost:3000/cart

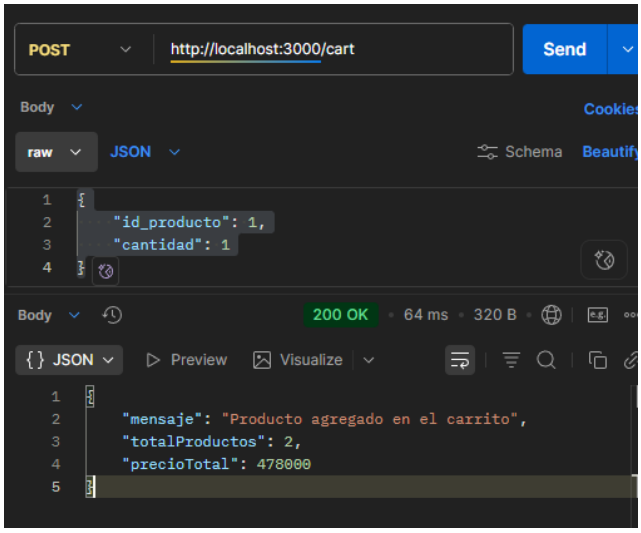
Requiere autentificación por token.

Json:

```
{
   "id_producto": 1,
   "cantidad": 1
}
```

Respuesta exitosa.

```
{
   "mensaje": "Producto agregado en el carrito",
   "totalProductos": 2,
   "precioTotal": 478000
}
```

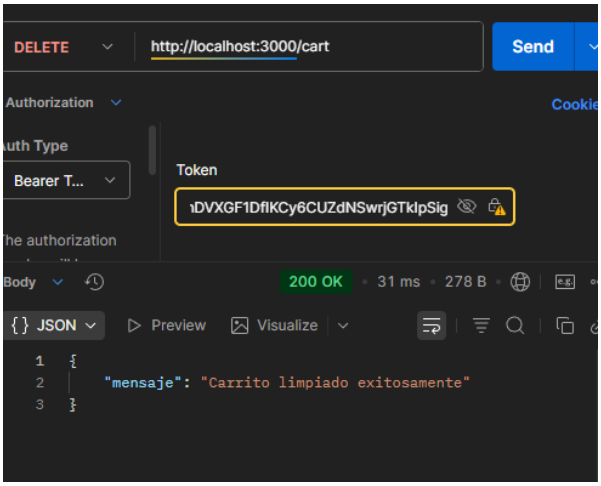


Limpiar Productos del Carrito: DELETE:
http://localhost:3000/cart

Requiere autentificación por token.

Respuesta exitosa.

```
{
   "mensaje": "Carrito limpiado exitosamente"
}
```

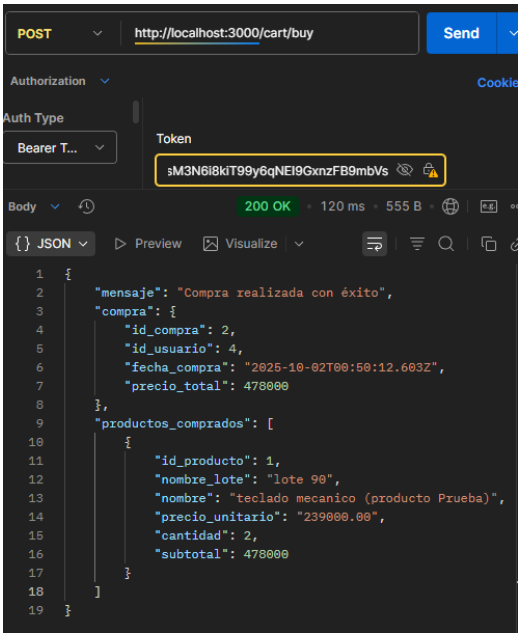


Comprar productos del carrito: POST:http://localhost:3000/cart/buy

Requiere autentificación por token.

Respuesta exitosa.

```
{
   "mensaje": "Compra realizada con éxito",
   "compra": {
      "id_compra": 2,
      "id_usuario": 4,
      "fecha_compra": "2025-10-02T00:50:12.603Z",
      "precio_total": 478000
   },
   "productos_comprados": [
      {
         "id_producto": 1,
         "nombre_lote": "lote 90",
         "nombre": "teclado mecanico (producto Prueba)",
         "precio_unitario": "239000.00",
         "cantidad": 2,
```

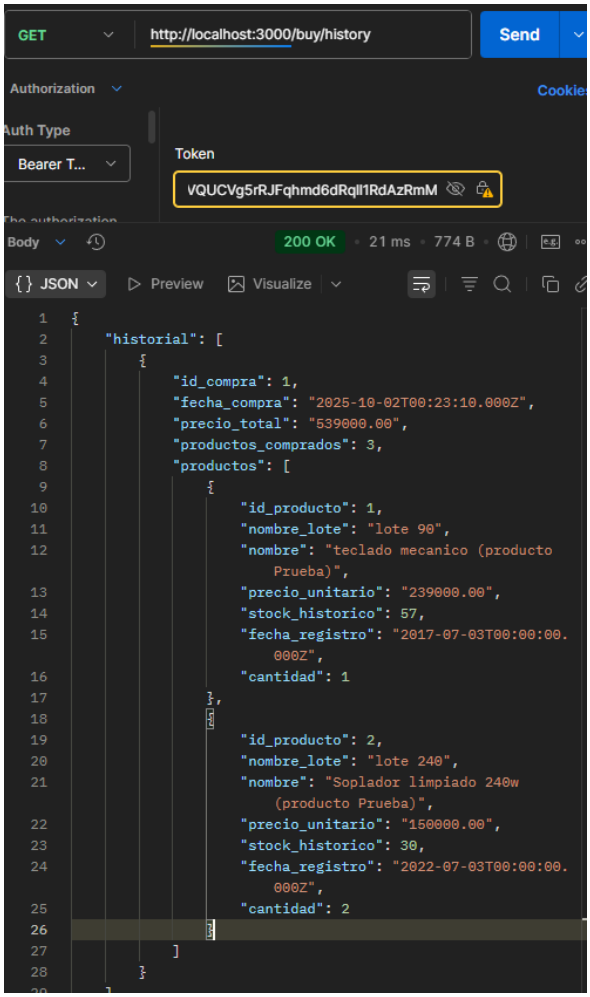


```
        "subtotal": 478000
      }
   ]
}
```


Historial de compras: GET: http://localhost:3000/buy/history

Requiere autentificación por token.

Respuesta exitosa.

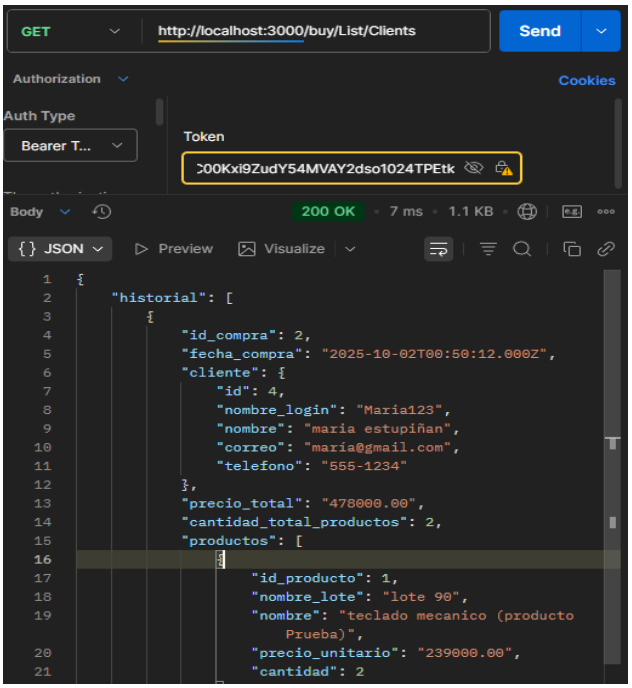



```
{
   "historial": [
      {
         "id_compra": 1,
         "fecha_compra": "2025-10-02T00:23:10.000Z",
         "precio_total": "539000.00",
         "productos_comprados": 3,
         "productos": [
            {
               "id_producto": 1,
               "nombre_lote": "lote 90",
               "nombre": "teclado mecanico (producto
Prueba)",
               "precio_unitario": "239000.00",
               "stock_historico": 57,
               "fecha_registro": "2017-07-
03T00:00:00.000Z",
               "cantidad": 1
            },
            {
               "id_producto": 2,
               "nombre_lote": "lote 240",
               "nombre": "Soplador limpiado 240w (producto
Prueba)",
               "precio_unitario": "150000.00",
               "stock_historico": 30,
               "fecha_registro": "2022-07-03T00:00:00.000Z",
               "cantidad": 2
            }
         ]
      }
   ]
}
```

Listar Compras clientes: GET: http://localhost:3000/buy/List/Clients

Requiere autentificación por token.

Respuesta exitosa.




```
{
   "historial": [
      {
         "id_compra": 2,
         "fecha_compra": "2025-10-02T00:50:12.000Z",
         "cliente": {
            "id": 4,
            "nombre_login": "Maria123",
            "nombre": "maria estupiñan",
            "correo": "maría@gmail.com",
            "telefono": "555-1234"
         },
         "precio_total": "478000.00",
         "cantidad_total_productos": 2,
         "productos": [
            {
               "id_producto": 1,
               "nombre_lote": "lote 90",
               "nombre": "teclado mecanico (producto Prueba)",
               "precio_unitario": "239000.00",
               "cantidad": 2
            }
         ]
      },
      {
         "id_compra": 1,
         "fecha_compra": "2025-10-02T00:23:10.000Z",
         "cliente": {
            "id": 2,
            "nombre_login": "dev_client",
            "nombre": null,
            "correo": null,
            "telefono": null
         },
         "precio_total": "539000.00",
         "cantidad_total_productos": 3,
         "productos": [
            {
               "id_producto": 1,
               "nombre_lote": "lote 90",
               "nombre": "teclado mecanico (producto Prueba)",
               "precio_unitario": "239000.00",
               "cantidad": 1
            },
            {
```

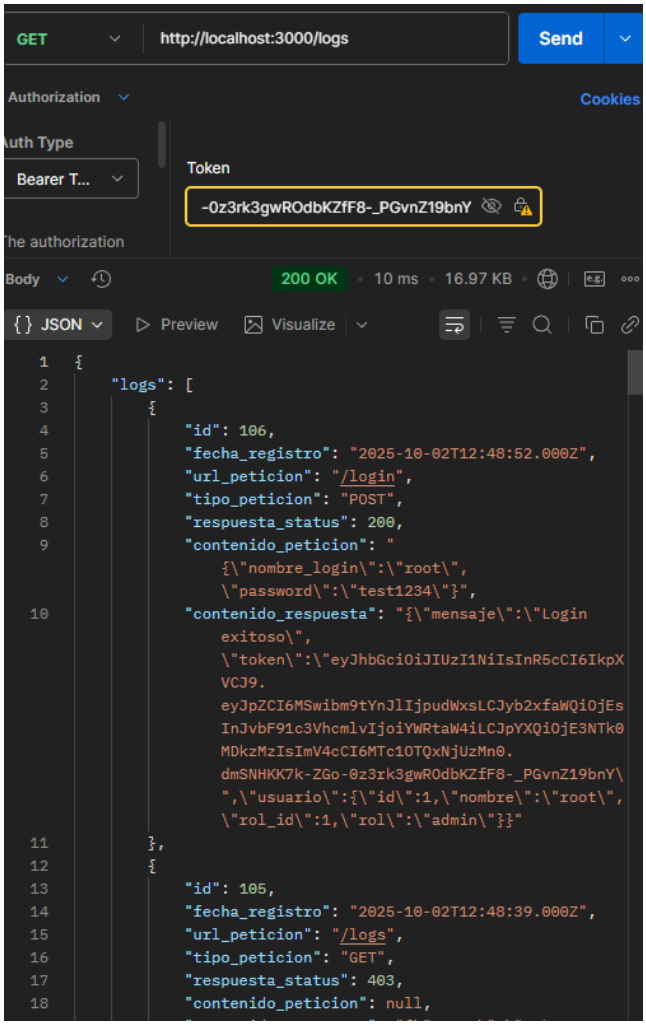
```
            "id_producto": 2,
            "nombre_lote": "lote 240",
            "nombre": "Soplador limpiado 240w (producto Prueba)",
            "precio_unitario": "150000.00",
            "cantidad": 2
        }
    ]
  }
 ]
}
```


Logs

Se tiene una sección dedicada a los logs, están como registros en base de datos.

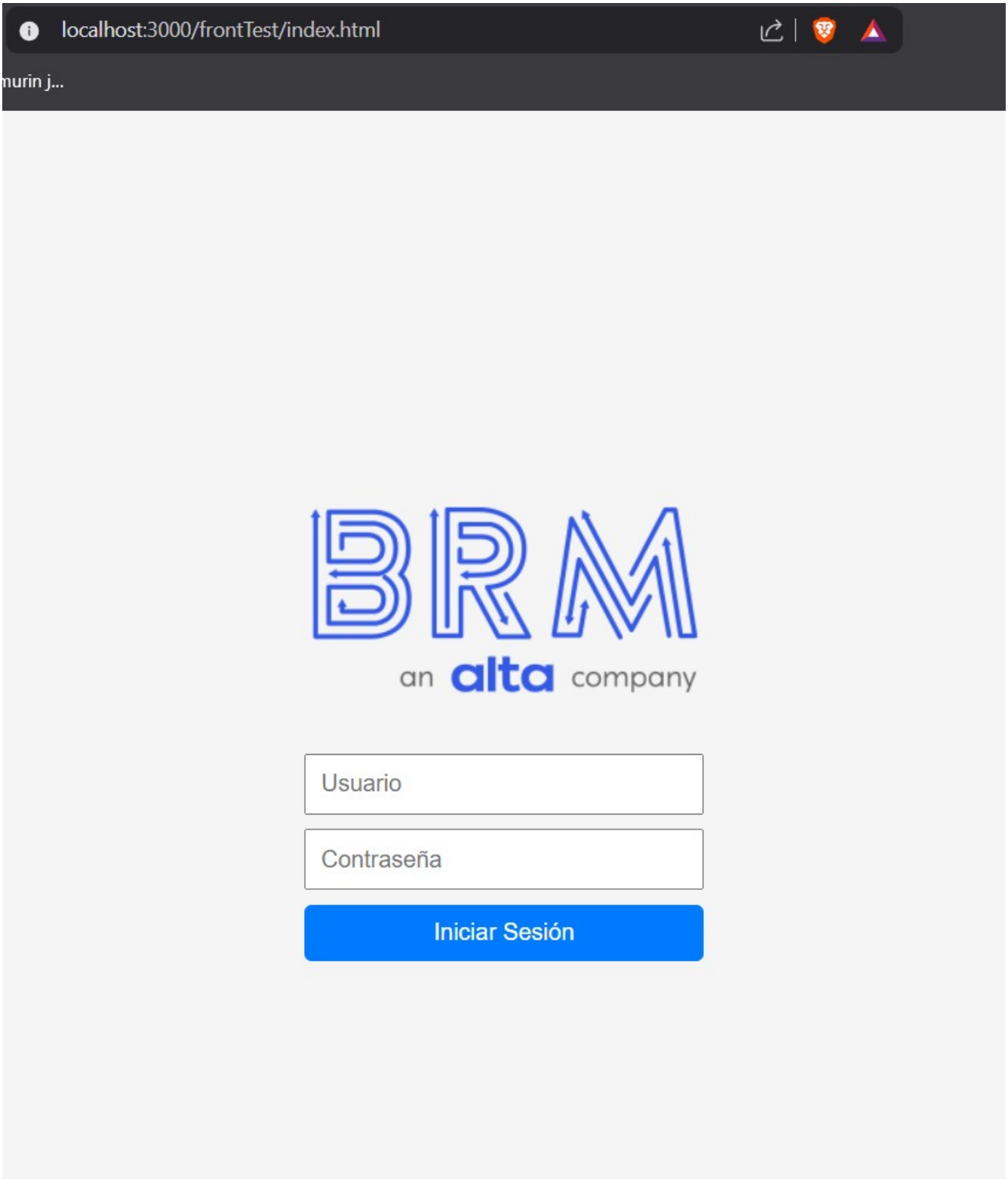
Get Logs: GET: http://localhost:3000/logs
Requiere autentificación por token.
Respuesta exitosa.


```
{
  "logs": [
    {
      "id": 106,
      "fecha_registro": "2025-10-
02T12:48:52.000Z",
      "url_peticion": "/login",
      "tipo_peticion": "POST",
      "respuesta_status": 200,
      "contenido_peticion": "{\"nombre_login\":
\"root\",\"password\":\"test1234\"}",
      "contenido_respuesta": "{\"men…
```

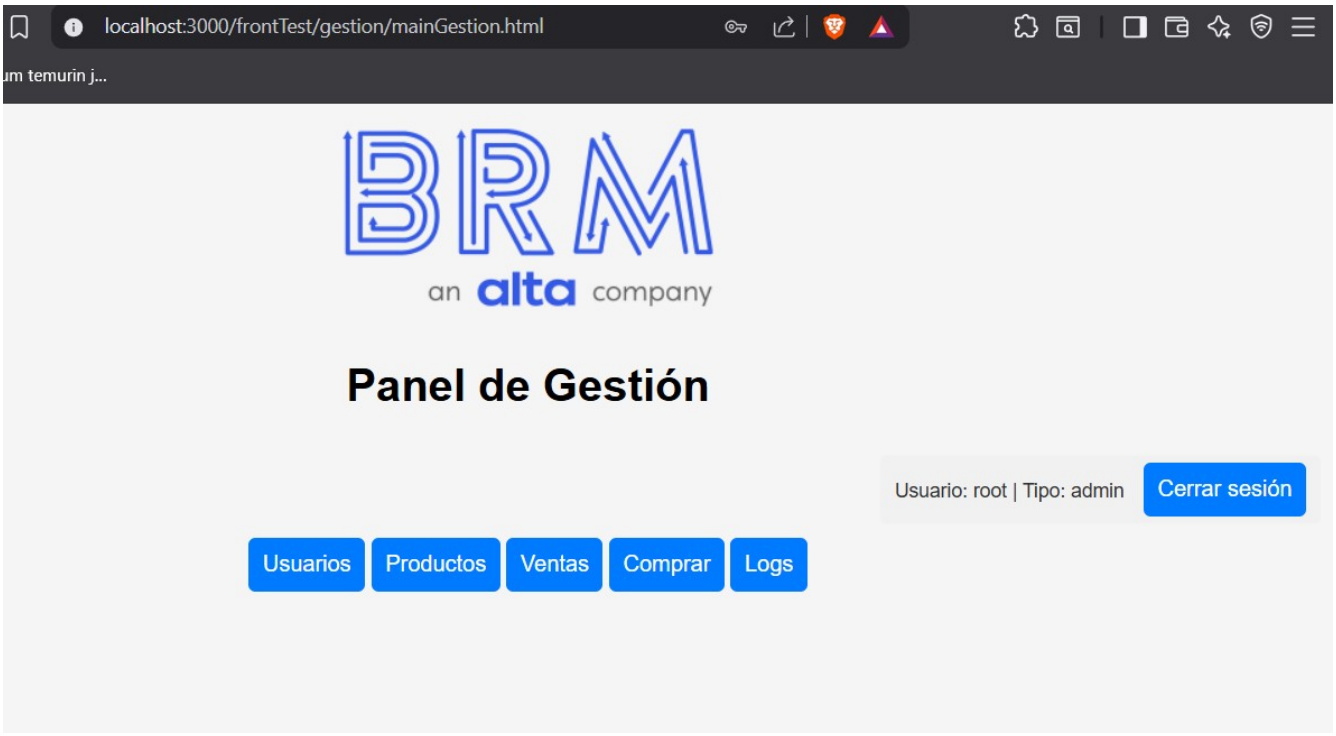



Se deja archivo de collección postman con todos los servicios creados, en el repositorio, para realizar pruebas, o también pueden realizar pruebas graficas mediante el ambiente grafico a continuación.

Pruebas Entorno Grafico

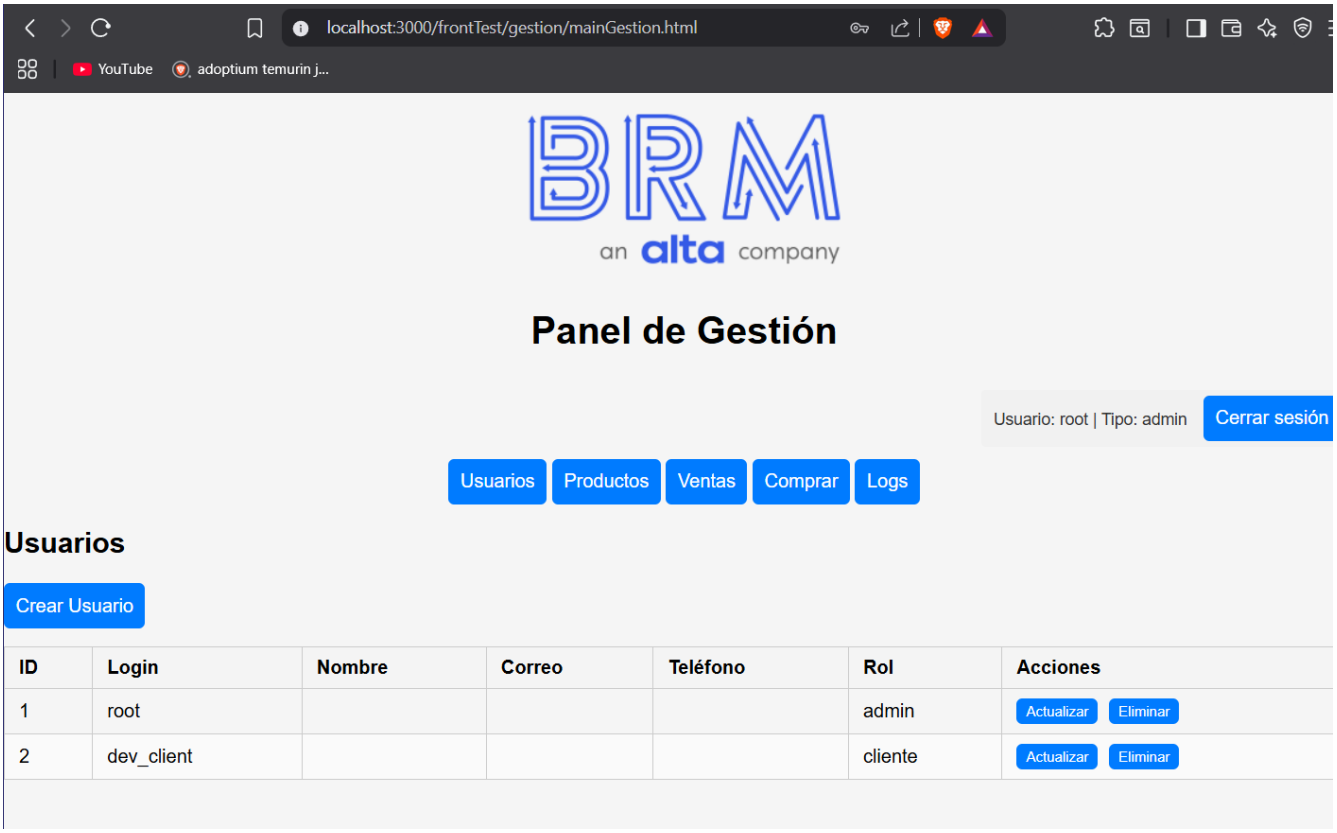


Generalmente se desarrollo un entorno grafico de pruebas para ver ejemplos de ejecución del backend, podemos acceder a este entorno haciendo un GET convencional desde un navegador a la url http://localhost:3000/frontTest, esta nos diseccionará al index.html de este apartado. Aquí podemos usar login y contraseña que hallamos creado o que vengan por defecto en el proyecto



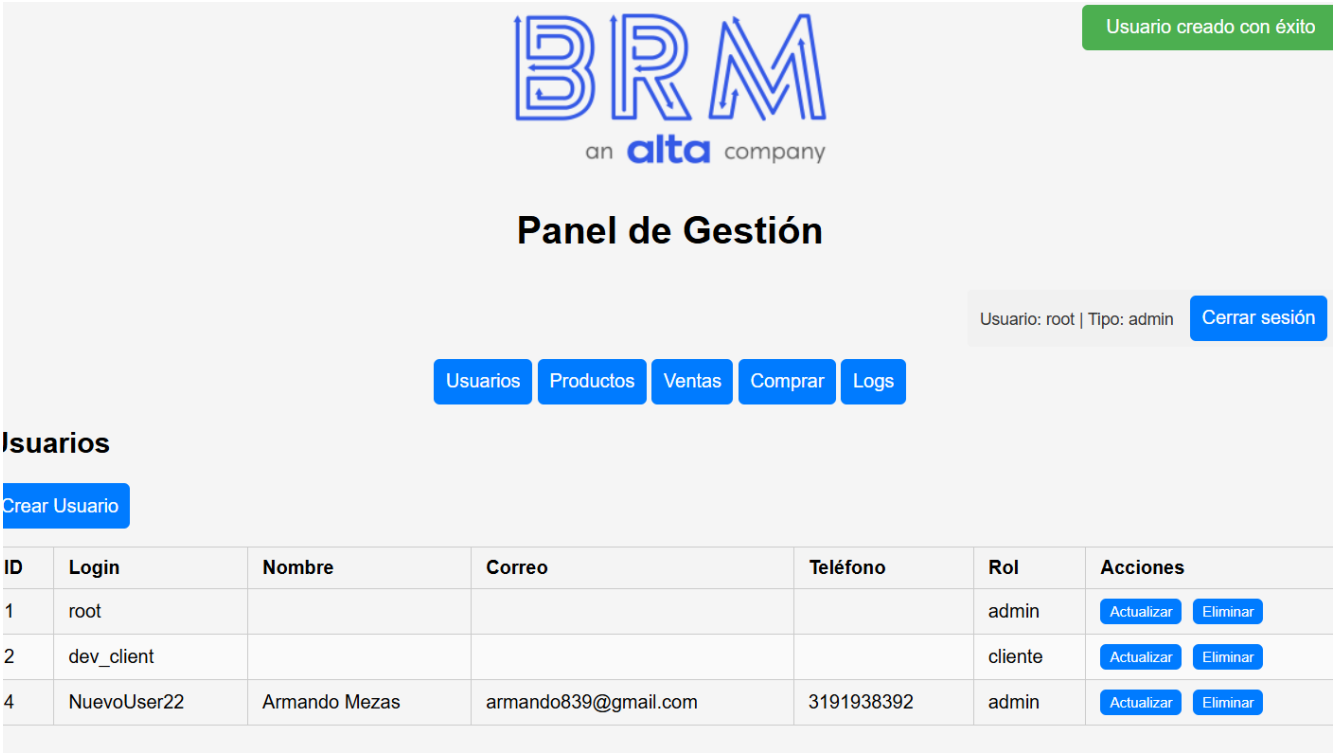
Aquí tendremos todos los apartados que se consumen desde el backend, tenemos crud de usuarios, crud de productos, procesos de ventas, historial de compras y visualización de logs.

Aquí se pueden visualizar todos los usuarios creados y sus roles.

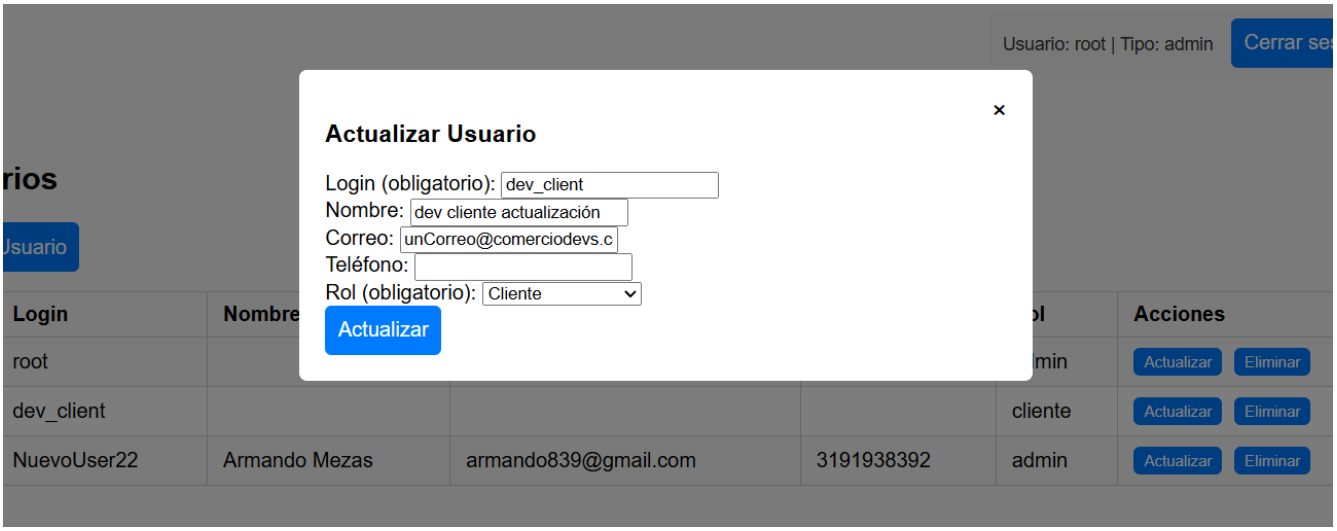


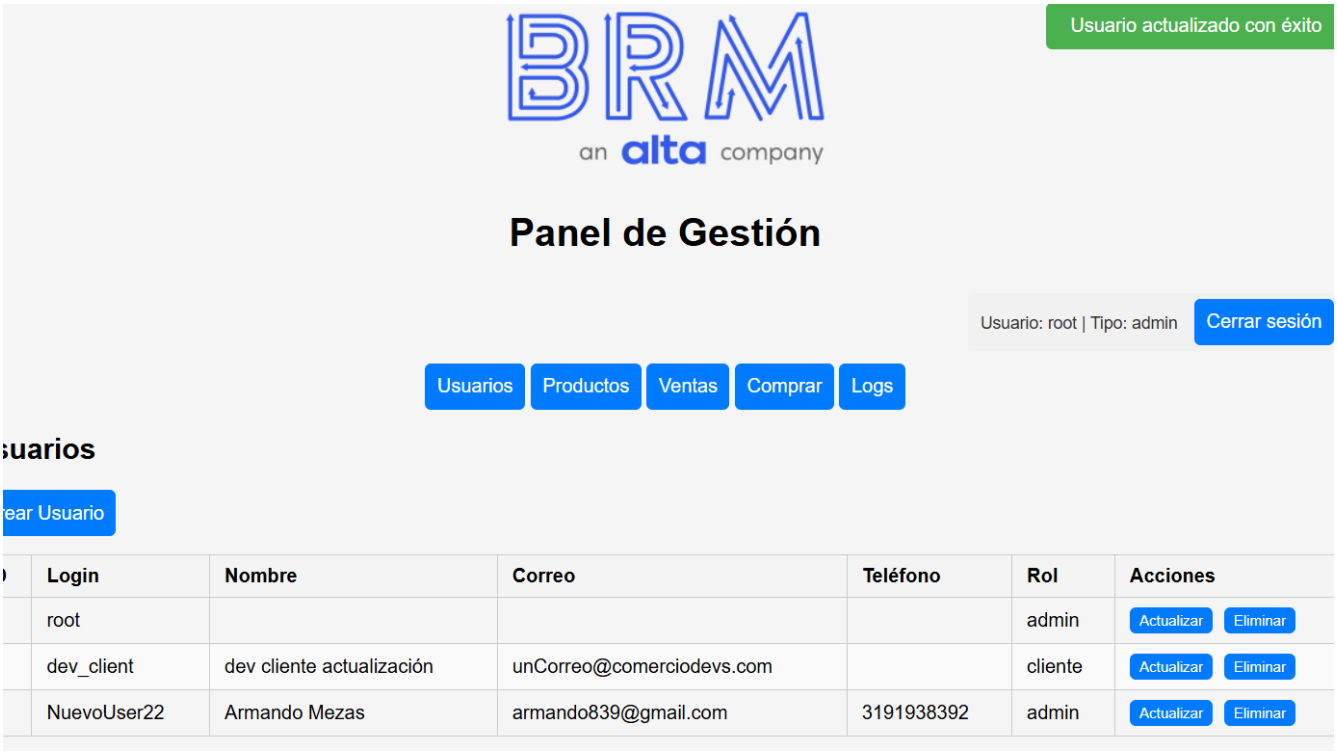
Al darle crear Usuario podemos llenar un formulario en un popup con los datos que nos solicitan y guardar la información.

La pantalla se actualizará y nos mostrará el proceso exitoso.

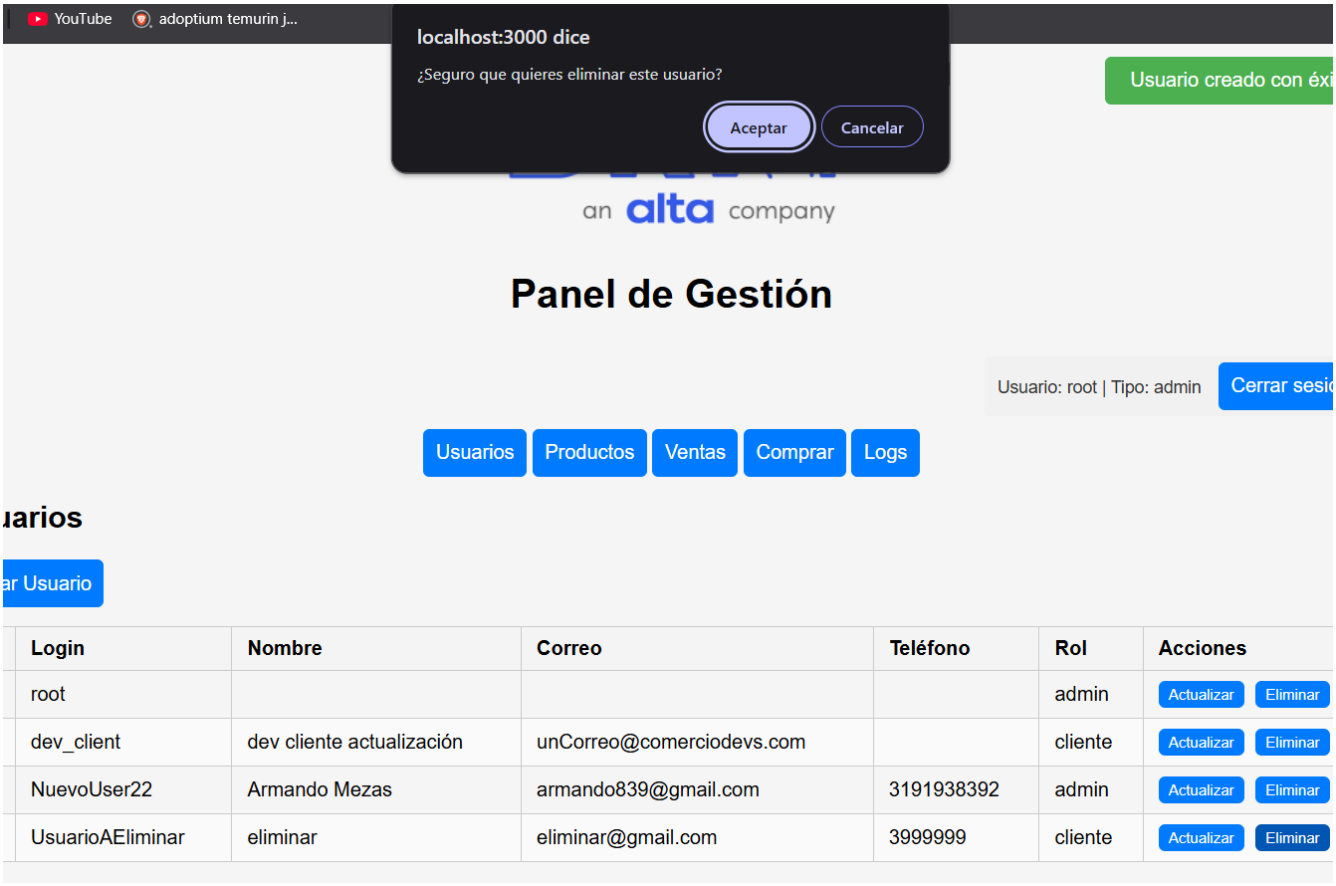


También podremos actualizar los usuarios, cabe recalcar que estas opciones solo son posibles acceder por un usuario con rol de administrador.

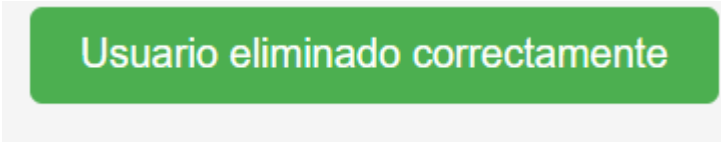




Se añadió una validación cuando se elimina un dato de la base de datos.

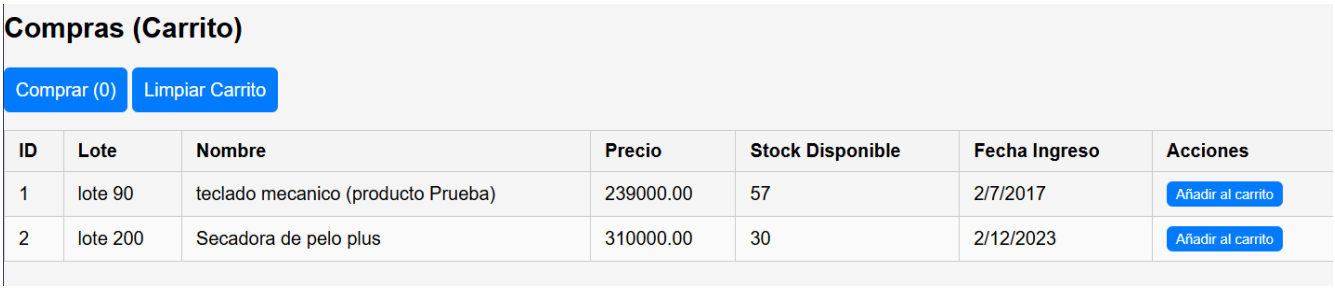


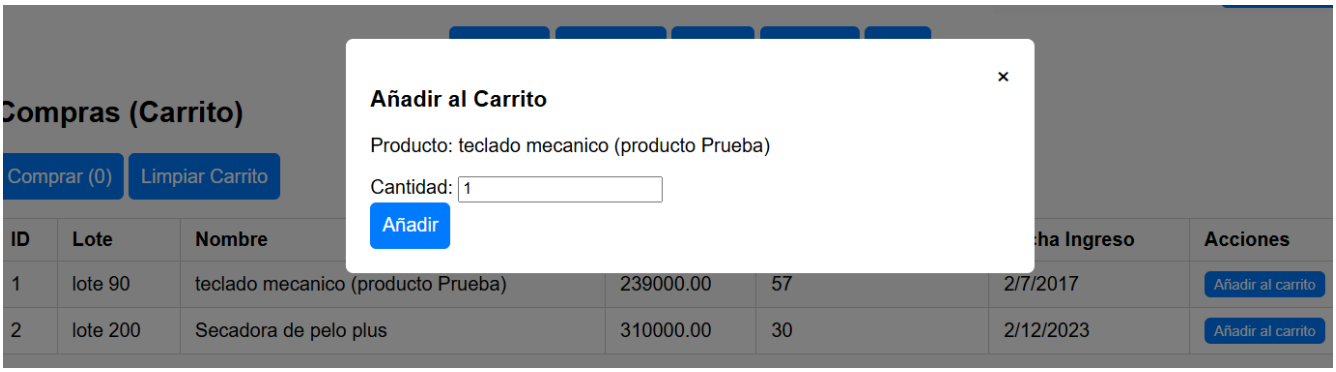
Los mensajes son dinámicos, y son el mensaje que responde el backend, al consumir el servicio.



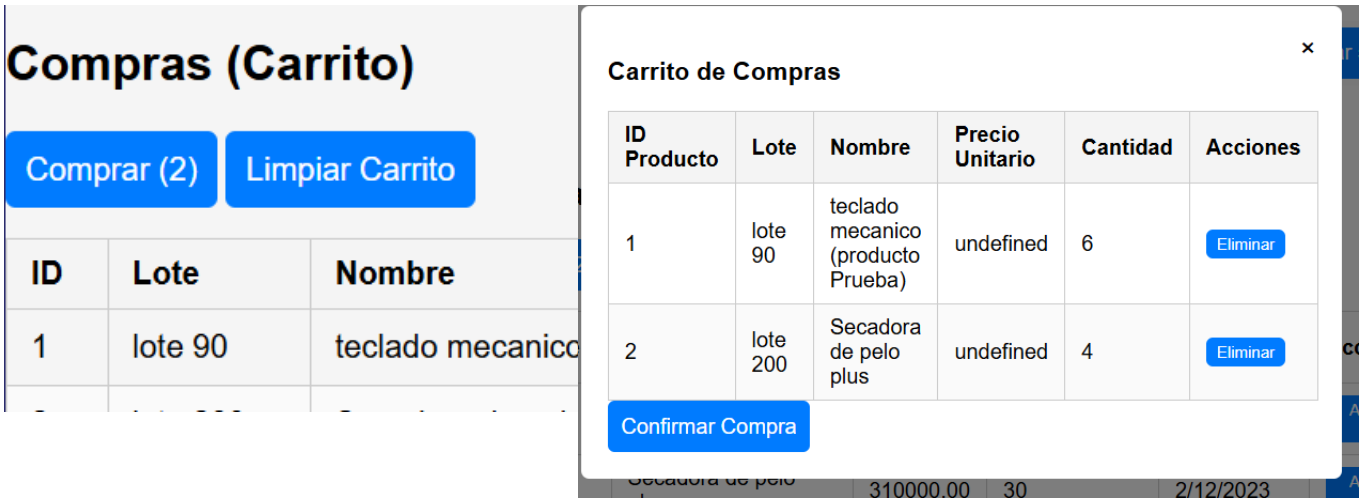
Se tiene el mismo comportamiento con productos.

Se puede realizar el proceso de compra para testear todos los llamados al backend.

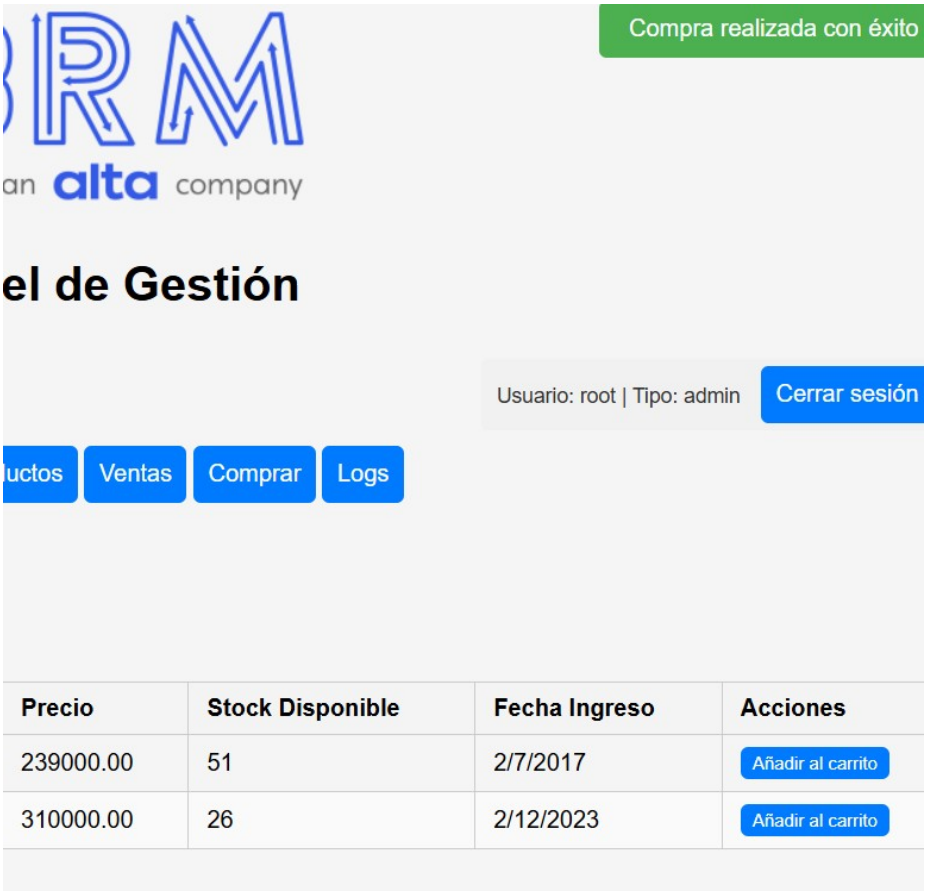




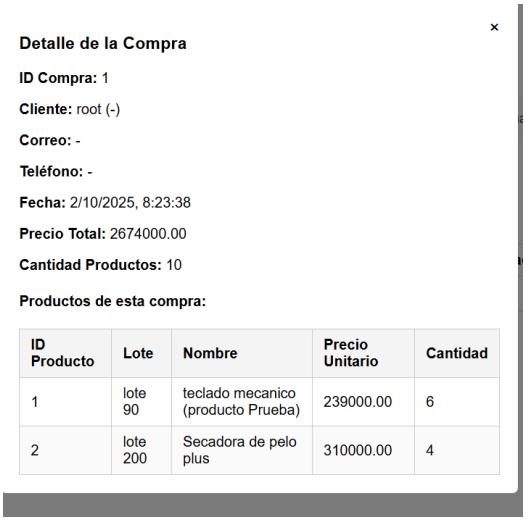
Se usa una tabla templar para almacenar los productos que desea comprar el cliente.



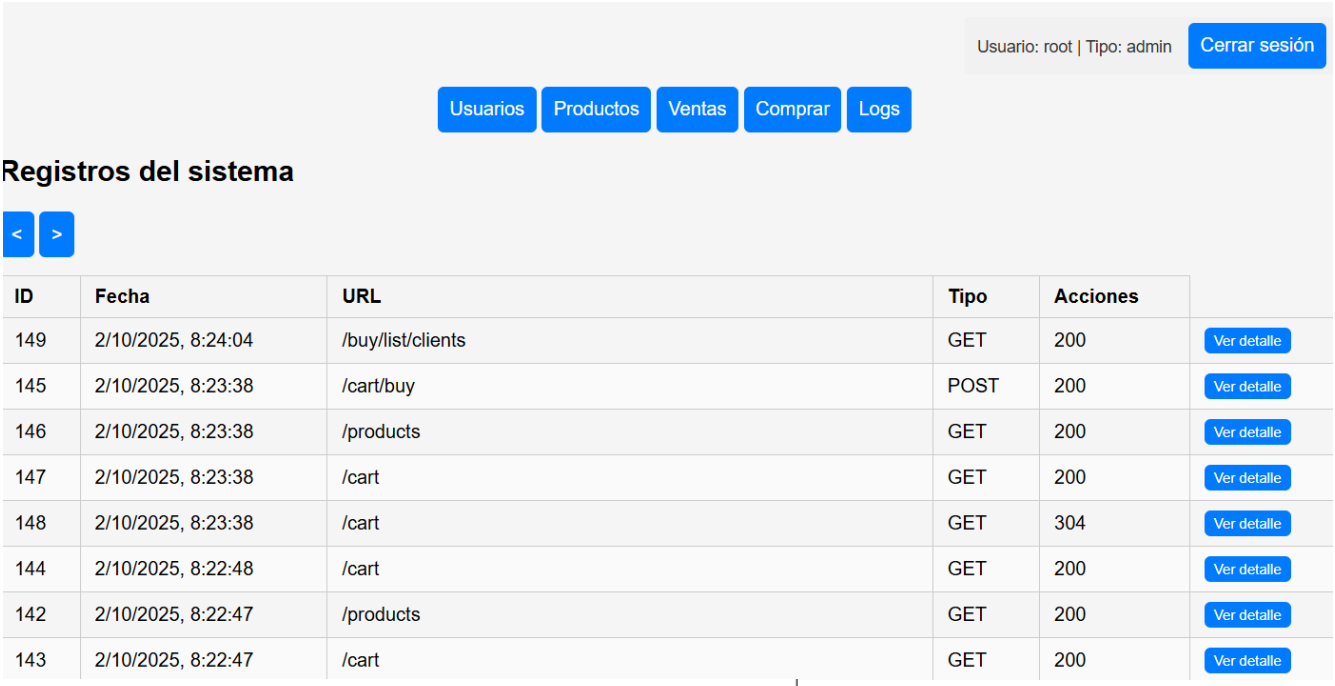
Así se pueden realizar las pruebas de las compras y comprobar que todos los datos estén correctos, como que se disminuyan los valores correctamente, de una forma mas rápida he intuitiva.



Y la venta ya esta disponible en la sección de ventas, en donde podemos ver a detalle quien, cuando y cuando compro el usuario.



Y por ultimo tenemos los registros de los logs que registran todos los procesos que tiene el backend.



El contenido de la petición son los datos que enviaron en la petición y el contenido de respuesta es lo que respondio el backend en ese momento.

Todo debidamente paginado.

Documentación técnica.

Se realiza el desarrollo de 7 modelos base, los cuales están como

Usuario {
  id: Integer, primaryKey, autoincrement
  nombre_login: String(20), not null
  password: String(60), not null
  nombre: String(60), optional
  correo: String(90), optional, must be email
  telefono: String(15), optional
  rol_id: Integer, not null, foreign key -> RolUsuario.id
}

nombre_login:
	String(20),
	único para login

password:
	String(60),
	obligatorio, hash de contraseña

nombre:
	String(60),
	opcional, nombre completo del usuario

correo:
	String(90),
	opcional, validación de formato email

telefono:
	String(15),
	opcional, número de contacto

rol_id:
	Integer,
	obligatorio, referencia al rol del usuario

RolUsuario {
  id: Integer, primaryKey, autoincrement
  nombre: String(20), not null
}

id:
	Integer,
	clave primaria, autoincremental
nombre
	String(20),
	obligatorio, representa el nombre del rol

Registro {
  id: Integer, primaryKey, autoincrement
  fecha_registro: Date, not null, default now
  url_peticion: String, not null
  tipo_peticion: String(10), not null
  respuesta_status: Integer, not null
  contenido_peticion: Text, optional
  contenido_respuesta: Text, optional
}

id
	Integer,
	clave primaria, autoincremental

fecha_registro
	Date,
	obligatorio, fecha y hora en que se realizó el registro, por defecto NOW

url_peticion
	String,
	obligatorio, URL del endpoint o recurso solicitado

tipo_peticion
	String(10),
	obligatorio, tipo de petición HTTP (GET, POST, etc.)

respuesta_status
	Integer,
	obligatorio, código de estado HTTP de la respuesta

contenido_peticion
	Text,
	opcional, contenido enviado en la petición (body)

contenido_respuesta
	Text,
	opcional, contenido devuelto en la respuesta del servidor

Producto {
  id: Integer, primaryKey, autoincrement
  nombre_lote: String(40), not null
  nombre: String(100), not null
  precio: Decimal(10,2), not null
  cantidad_disponible: Integer, not null
  fecha_ingreso: Date, not null
}

id
	Integer,

clave primaria, autoincremental

nombre_lote
String(40),
obligatorio, identifica el lote del producto

nombre
String(100),
obligatorio, nombre del producto

precio
Decimal(10,2),
obligatorio, precio unitario del producto

cantidad_disponible
Integer,
obligatorio, cantidad de unidades disponibles en inventario

fecha_ingreso
Date,
obligatorio, fecha en que el producto ingresó al sistema

ProcesoCompra {
id: Integer, primaryKey, autoincrement
id_usuario: Integer, not null, foreign key -> Usuario.id
id_producto: Integer, not null, foreign key -> Producto.id
cantidad: Integer, not null, default 1
}

id
Integer,
clave primaria, autoincremental

id_usuario
Integer,
obligatorio, referencia al usuario que realiza la compra (Usuario.id)

id_producto
Integer,
obligatorio, referencia al producto agregado al proceso de compra (Producto.id)

cantidad
Integer,
obligatorio, cantidad del producto en el proceso de compra, valor por defecto 1

HistorialProducto {
id: Integer, primaryKey, autoincrement

id_compra: Integer, not null, foreign key -> Compra.id_compra
id_producto_hist: Integer, not null
nombre_lote: String(40), not null
nombre: String(100), not null
precio_hist: Decimal(10,2), not null
stock_hist: Integer, not null
fecha_hist: Date, not null
cantidad_comprada: Integer, not null
}

id
Integer,
clave primaria, autoincremental

id_compra
Integer,
obligatorio, referencia a la compra asociada (Compra.id_compra)

id_producto_hist
Integer,
obligatorio, identifica el producto histórico

nombre_lote
String(40),
obligatorio, nombre del lote del producto

nombre
String(100),
obligatorio, nombre del producto

precio_hist
Decimal(10,2),
obligatorio, precio del producto en el historial de la compra

stock_hist
Integer,
obligatorio, cantidad disponible registrada en el historial

fecha_hist
Date,
obligatorio, fecha del registro en el historial

cantidad_comprada
Integer,
obligatorio, cantidad de ese producto que se compró

Compra {

id_compra: Integer, primaryKey, autoincrement

id_usuario: Integer, not null, foreign key -> Usuario.id

fecha_compra: Date, not null

precio_total: Decimal(10,2), not null

}

id_compra
Integer,
clave primaria, autoincremental

id_usuario
Integer,
obligatorio, referencia al usuario que realiza la compra (Usuario.id)

fecha_compra
Date,
obligatorio, fecha en que se realiza la compra, por defecto la fecha actual

precio_total
Decimal(10,2),
obligatorio, total de la compra