

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$10$x1dHBD7XRph8Us7aH/fB1e8thBSeJyYkaL2sGAPTfaPN4Kqo38YH2

DEVCLIENT_LOGIN=dev_client
USERADMIN_PASS=$2b$10$x1dHBD7XRph8Us7aH/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.

```
E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend>node index.js
[dotenv@17.2.3] injecting env (9) from .env -- tip: 🐞 suppress all logs with { quiet: true }
[dotenv@17.2.3] injecting env (0) from .env -- tip: 🐞 override existing env vars with { override: true }
[dotenv@17.2.3] injecting env (0) from .env -- tip: 🐞 backup and recover secrets: https://dotenvx.com/ops
Base de datos brm_base verificada o creada.
tablas sincronizadas correctamente
Conexion a base de datos exitosa.
Servidor corriendo en http://localhost:3000
```

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.

```
E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend>node index.js
[dotenv@17.2.3] injecting env (9) from .env -- tip: add secrets lifecycle management: https://dotenvx.com/ops
[dotenv@17.2.3] injecting env (0) from .env -- tip: run anywhere with `dotenvx run -- yourcommand`
[dotenv@17.2.3] injecting env (0) from .env -- tip: add secrets lifecycle management: https://dotenvx.com/ops
Error creando/verificando la base de datos: Error: getaddrinfo ENOTFOUND localho
    at Object.createConnectionPromise [as createConnection] (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\node_modules\mysql2\promise.js:19:31)
    at sincronizarDatos (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\config\initDB.js:12:36)
    at iniciarServidor (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\index.js:35:11)
    at Object.<anonymous> (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\index.js:56:1)
    at Module._compile (node:internal/modules/cjs/loader:1706:14)
    at Object.<.> (node:internal/modules/cjs/loader:1839:10)
    at Module.load (node:internal/modules/cjs/loader:1441:32)
    at Function._load (node:internal/modules/cjs/loader:1263:12)
    at TracingChannel.traceSync (node:diagnostics_channel:322:14)
    at wrapModuleLoad (node:internal/modules/cjs/loader:237:24) {
  code: 'ENOTFOUND',
  errno: -3008,
  sqlState: undefined
}
Error al iniciar Servidor: Error: getaddrinfo ENOTFOUND localho
    at Object.createConnectionPromise [as createConnection] (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\node_modules\mysql2\promise.js:19:31)
    at sincronizarDatos (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\config\initDB.js:12:36)
    at iniciarServidor (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\index.js:35:11)
    at Object.<anonymous> (E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend\index.js:56:1)
    at Module._compile (node:internal/modules/cjs/loader:1706:14)
    at Object.<.> (node:internal/modules/cjs/loader:1839:10)
    at Module.load (node:internal/modules/cjs/loader:1441:32)
    at Function._load (node:internal/modules/cjs/loader:1263:12)
    at TracingChannel.traceSync (node:diagnostics_channel:322:14)
    at wrapModuleLoad (node:internal/modules/cjs/loader:237:24) {
  code: 'ENOTFOUND',
  errno: -3008,
  sqlState: undefined
}
E:\Documentos\DesarrolloGeneral\pruebasTrabajo\BRMPuebaDesarrolloBackend>
```

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

	id	nombre
1	1	admin
2	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": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibm9tYnJlIjpudWxsLCJyb2kiOiJlbnVvF91c3VhcmlyIjoiYWRTaW4iLCJpYXQiOiJlbnVvF91c3VhcmlyMTc1OTQwMzg4OXB0.eyJpZCI6MSwibm9tYnJlIjpudWxsLCJyb2kiOiJlbnVvF91c3VhcmlyMTc1OTQwMzg4OXB0.eyJpZCI6MSwibm9tYnJlIjpudWxsLCJyb2kiOiJlbnVvF91c3VhcmlyMTc1OTQwMzg4OXB0",
  "usuario": {
    "id": 1,
    "nombre": "root",
    "rol_id": 1,
    "rol": "admin"
  }
}
```

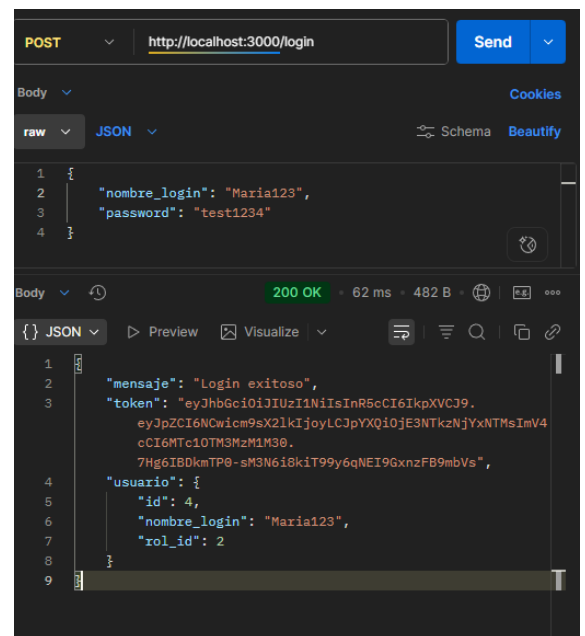
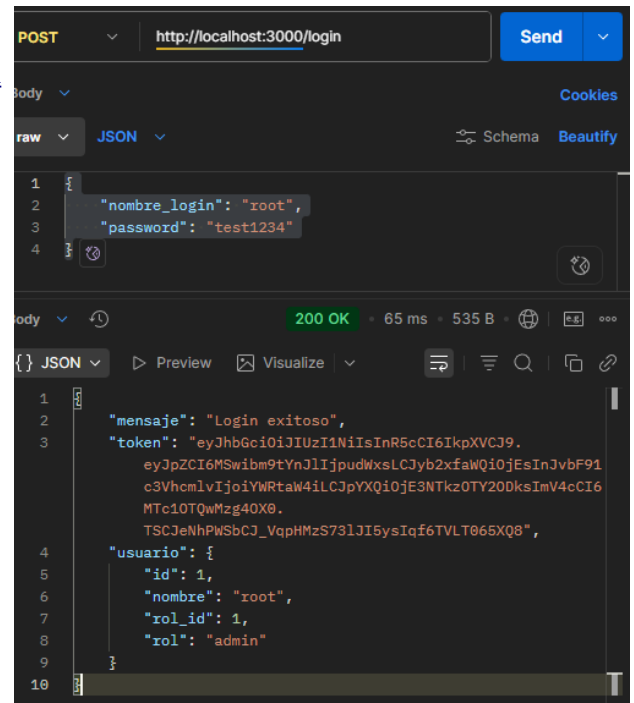
respuesta exitosa cliente <http://localhost:3000/login>

json:

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

respuesta:

```
{
  "mensaje": "Login exitoso",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NCwibm9tYnJlIjpudWxsLCJyb2kiOiJlbnVvF91c3VhcmlyMTc1OTQwMzg4OXB0.eyJpZCI6NCwibm9tYnJlIjpudWxsLCJyb2kiOiJlbnVvF91c3VhcmlyMTc1OTQwMzg4OXB0",
  "usuario": {
    "id": 4,
    "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.

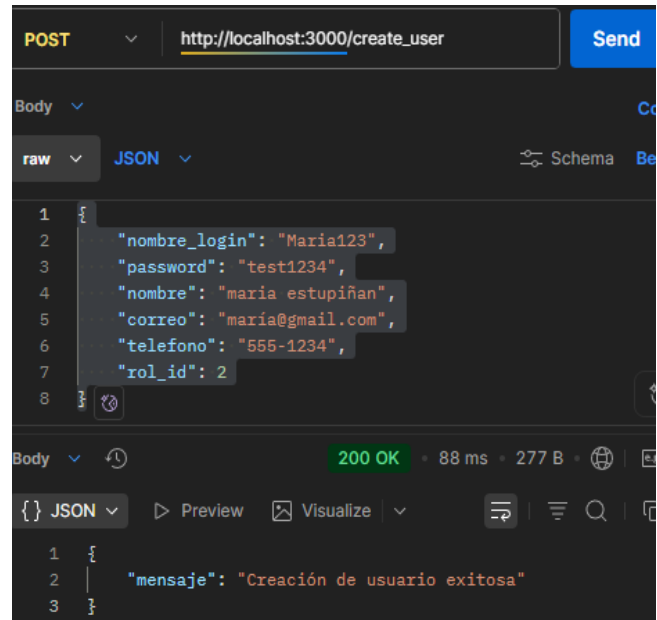
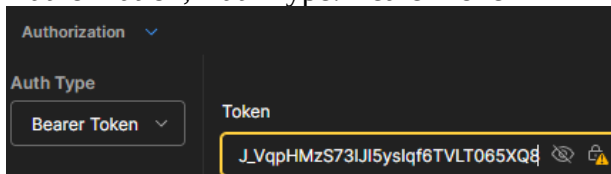
Creación: POST: http://localhost:3000/create_user

json:

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

Debe ingresar el token en el parámetro

Authorization, Auth Type: Bearer Token



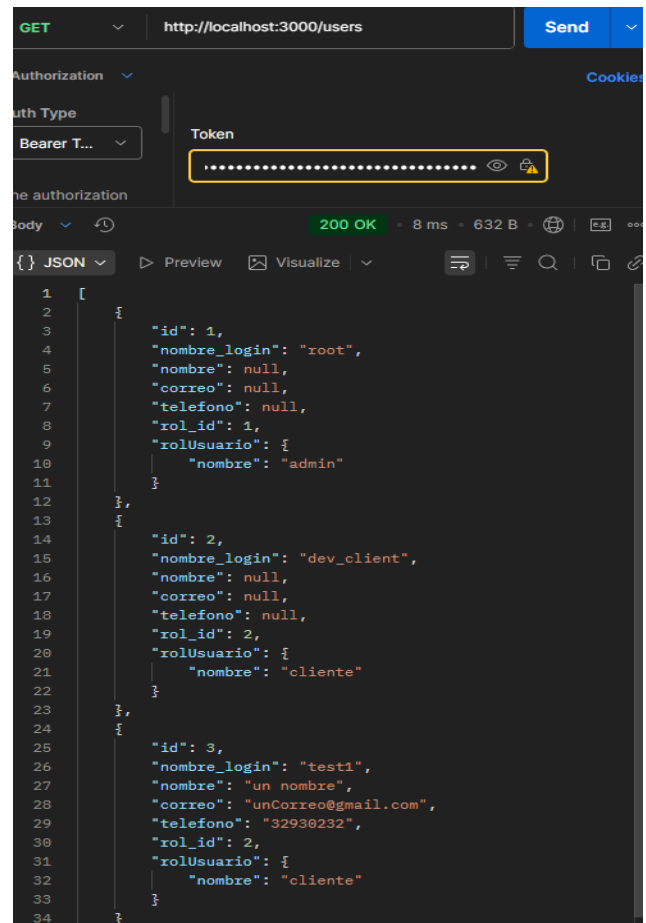
respuesta exitosa:

```
{
  "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,
  },
  {
    "id": 3,
    "nombre_login": "test1",
    "nombre": "un nombre",
    "correo": "unCorreo@gmail.com",
    "telefono": "32930232",
    "rol_id": 2,
    "rolUsuario": {
      "nombre": "cliente"
    }
  }
]
```



```

        "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 autenticación por token.

respuesta exitosa:

```

{
  "mensaje": "Usuario actualizado exitosamente"
}

```

Eliminación DELTE: <http://localhost:3000/user/3>

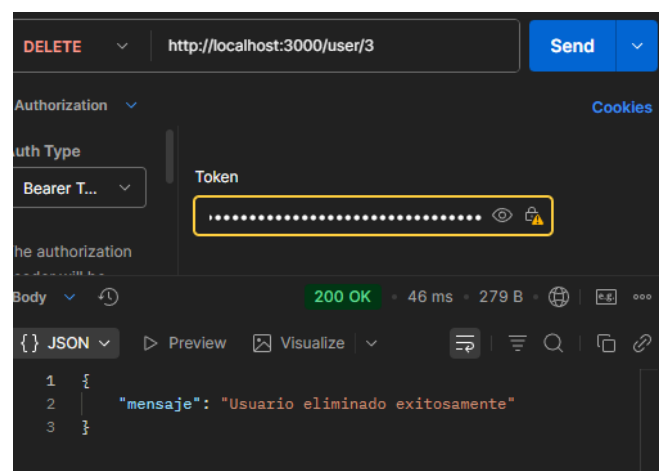
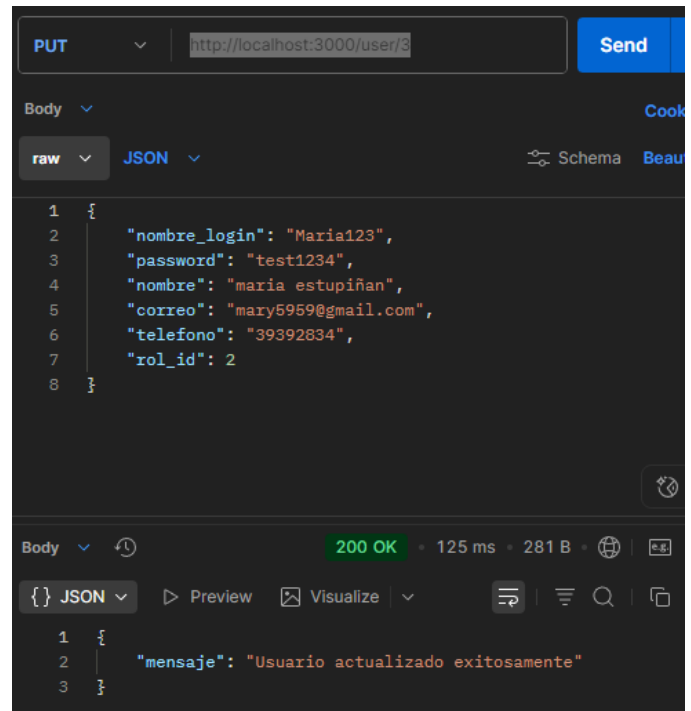
Requiere autenticación por token.

respuesta exitosa:

```

{
  "mensaje": "Usuario eliminado exitosamente"
}

```



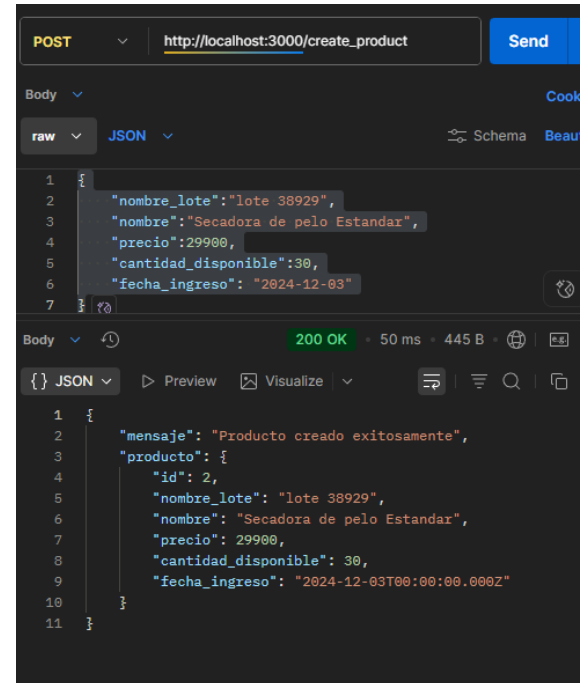
Crud Productos

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 autenticación 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"
  }
}
```

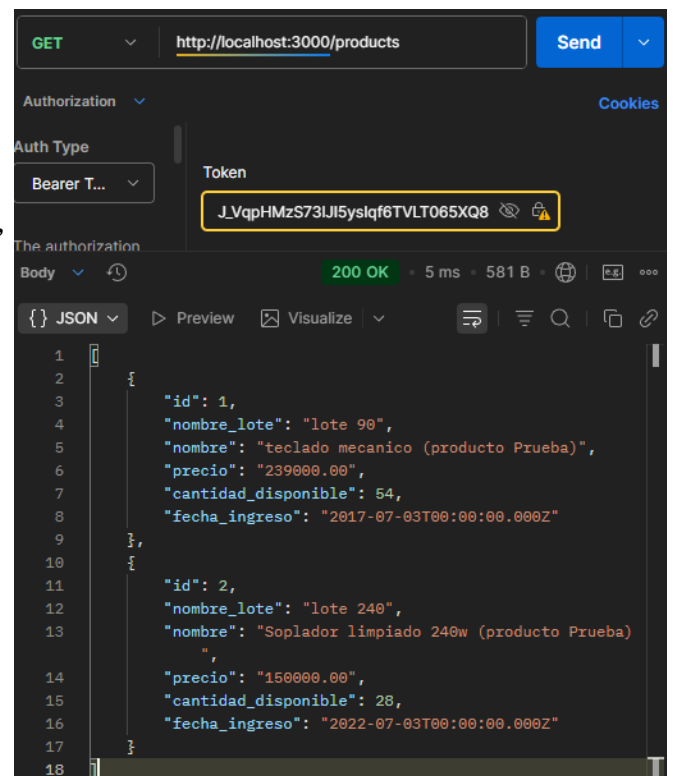


Consultar Productos: GET: <http://localhost:3000/products>

Requiere autenticación 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"
  }
]
```



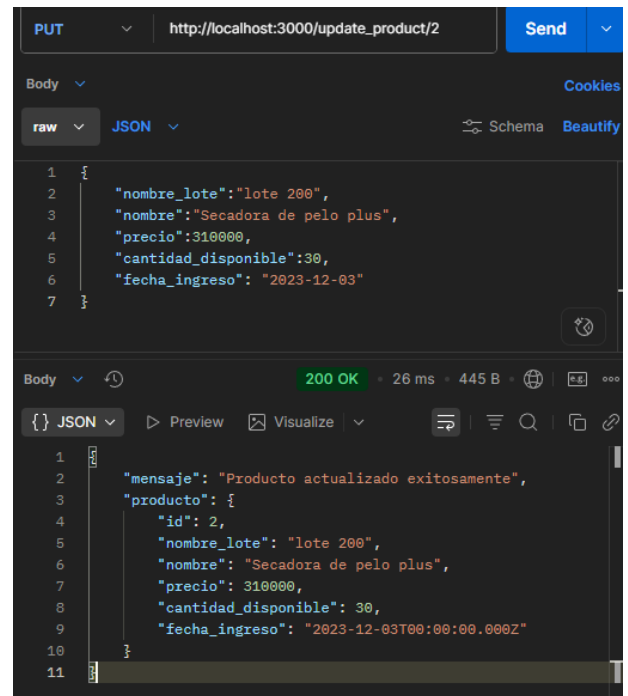
Actualización Producto PUT: http://localhost:3000/update_product/90

json: Requiere autenticación 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"
  }
}
```

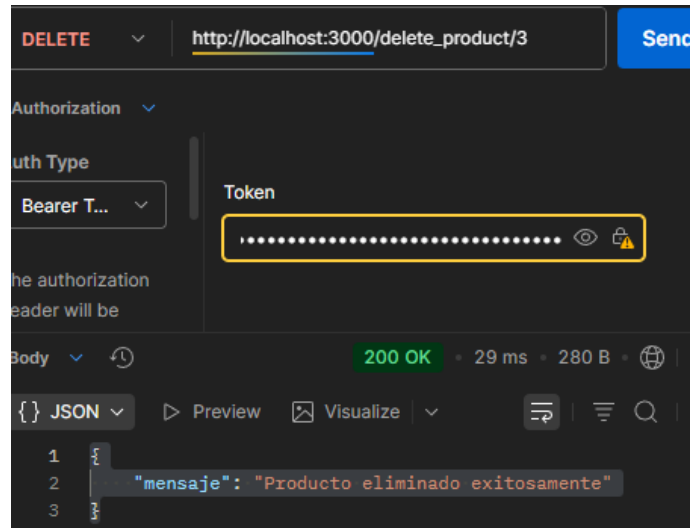


Eliminación Producto: DELETE: http://localhost:3000/delete_product/3

Requiere autenticación por token.

respuesta exitosa.

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



Compras

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

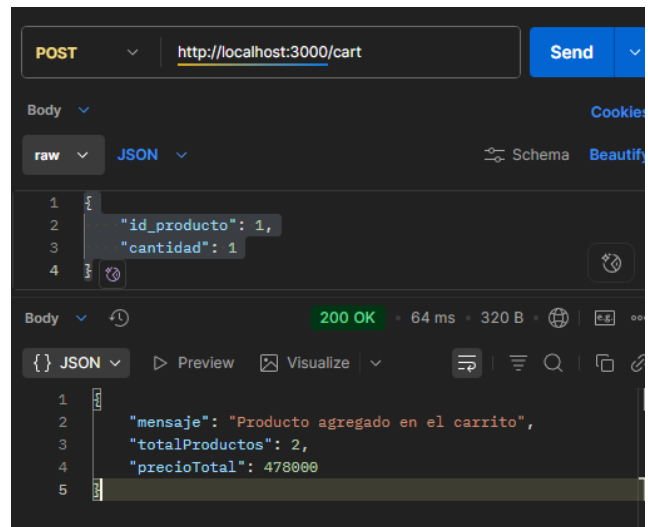
Requiere autenticación por token.

Json:

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

Respuesta exitosa.

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



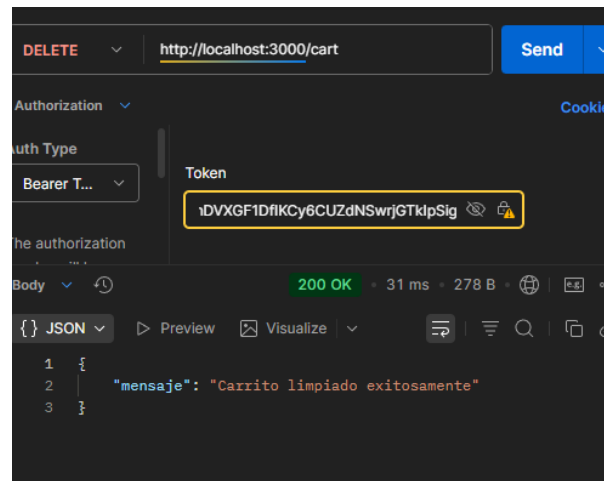
Limpiar Productos del Carrito: DELETE:

<http://localhost:3000/cart>

Requiere autenticación por token.

Respuesta exitosa.

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



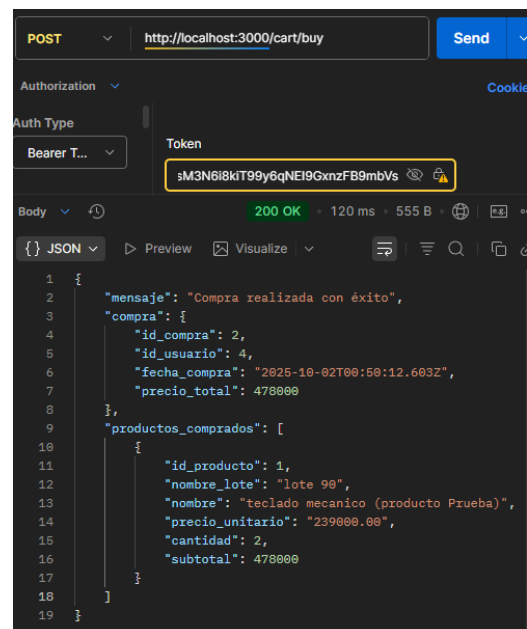
Comprar productos del carrito: POST: <http://localhost:3000/cart/buy>

Requiere autenticación 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
    }
}
]
}
}
}

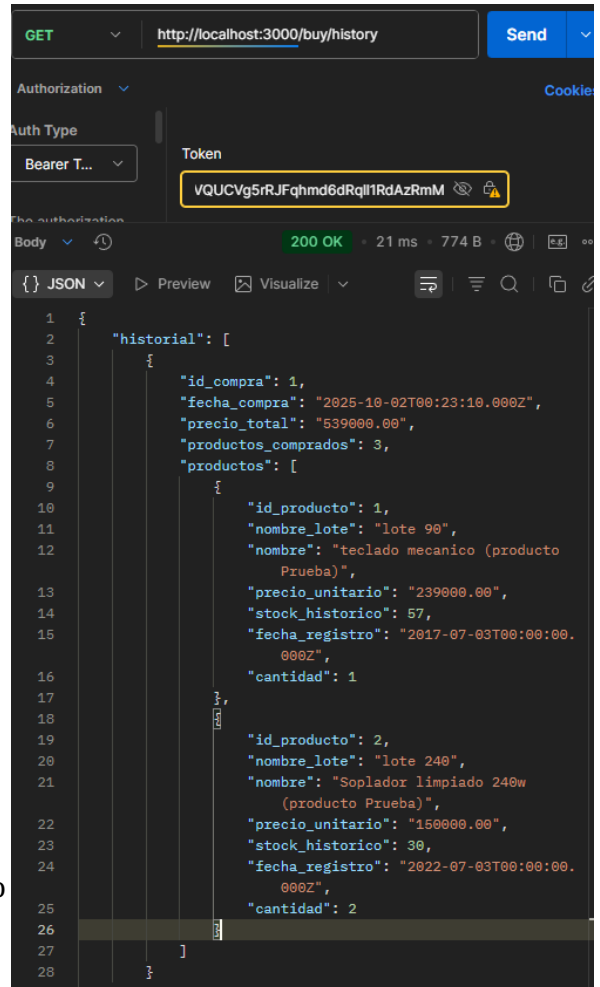
```

Historial de compras: GET: <http://localhost:3000/buy/history>
 Requiere autenticación 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
        }
      ]
    }
  ]
}

```

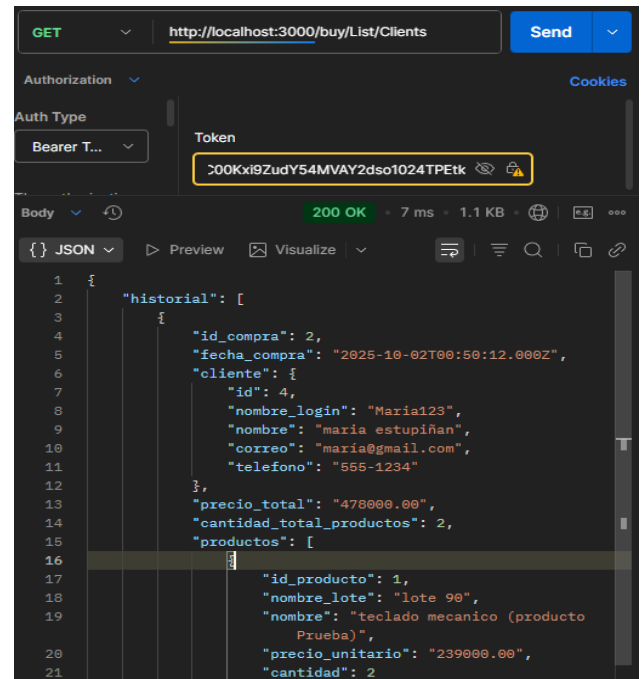


Listar Compras clientes: GET: <http://localhost:3000/buy/List/Clients>

Requiere autenticación 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>

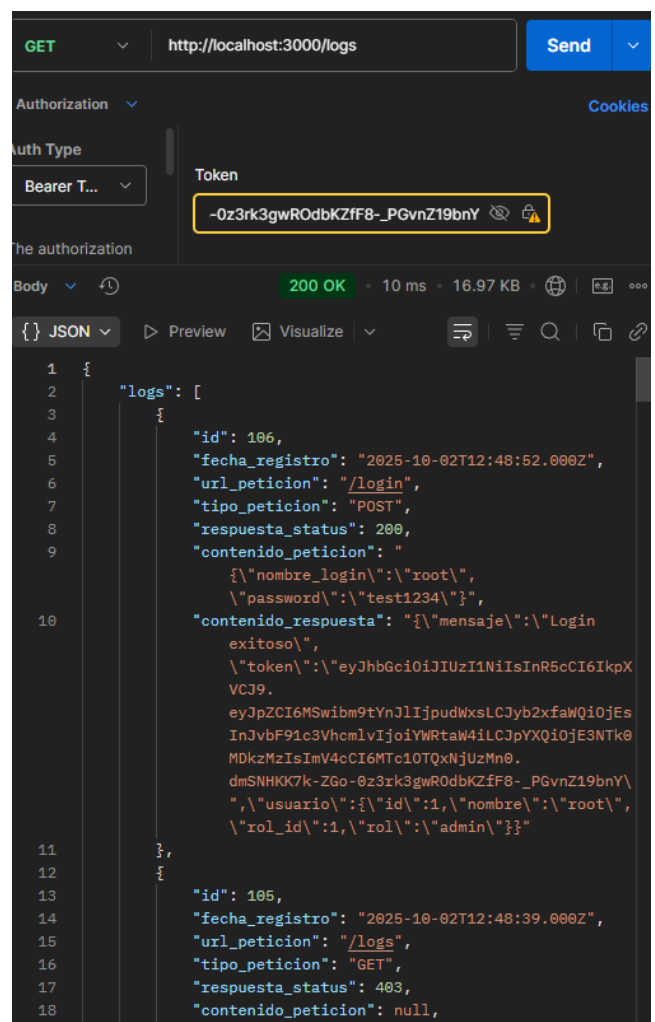
Requiere autenticación por token.

Respuesta exitosa.

```

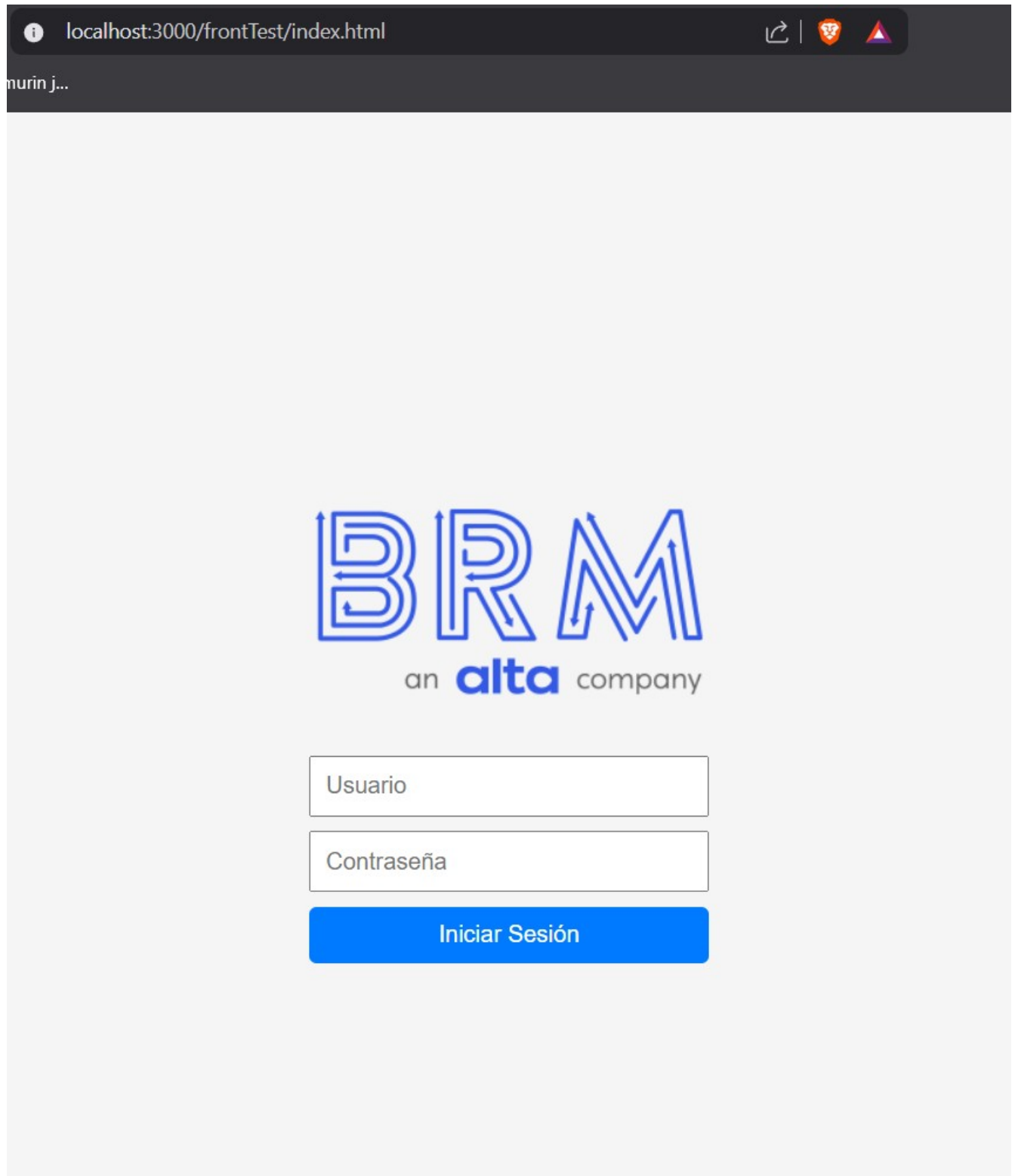
{
  "logs": [
    {
      "id": 106,
      "fecha_registro": "2025-10-02T12:48:52.000Z",
      "url_peticion": "/login",
      "tipo_peticion": "POST",
      "respuesta_status": 200,
      "contenido_peticion": "{ \"nombre_login\": \"root\", \"password\": \"test1234\" }",
      "contenido_respuesta": "{ \"mensaje\": \"Login exitoso\", \"token\": \"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibm9tYnJlIjpudWxsLCJyb2xfaWQjE3NTk0MDkzMzIsImV4cCI6MTc1OTQxNjUzMn0.dmSNHKK7k-ZGo-0z3rk3gwROdbkZfF8-_PGvnZ19bnY\", \"usuario\": { \"id\": 1, \"nombre\": \"root\", \"rol_id\": 1, \"rol\": \"admin\" } }"
    },
    {
      "id": 105,
      "fecha_registro": "2025-10-02T12:48:39.000Z",
      "url_peticion": "/logs",
      "tipo_peticion": "GET",
      "respuesta_status": 403,
      "contenido_peticion": null,
    }
  ]
}

```

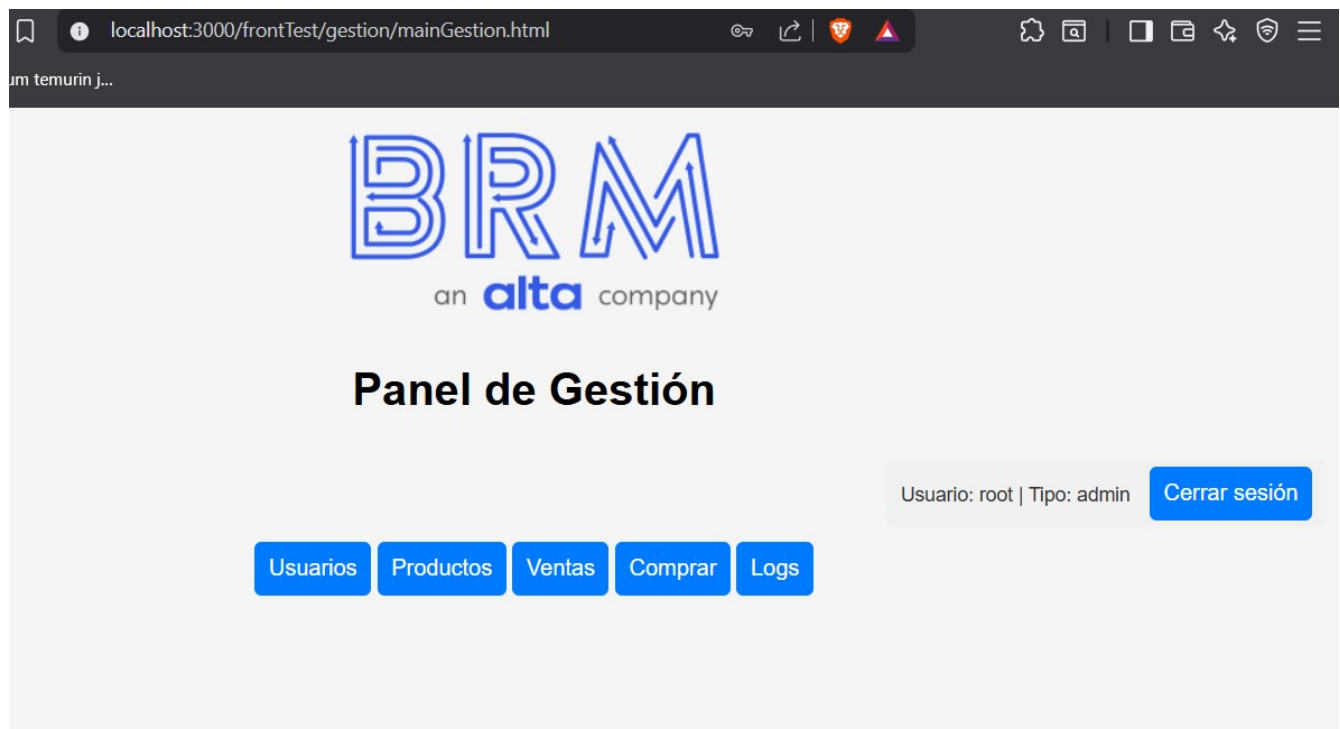


Se deja archivo de colección postman con todos los servicios creados, en el repositorio, para realizar pruebas, o también pueden realizar pruebas graficas mediante el ambiente grafico a continuación.

Pruebas Entorno Grafico

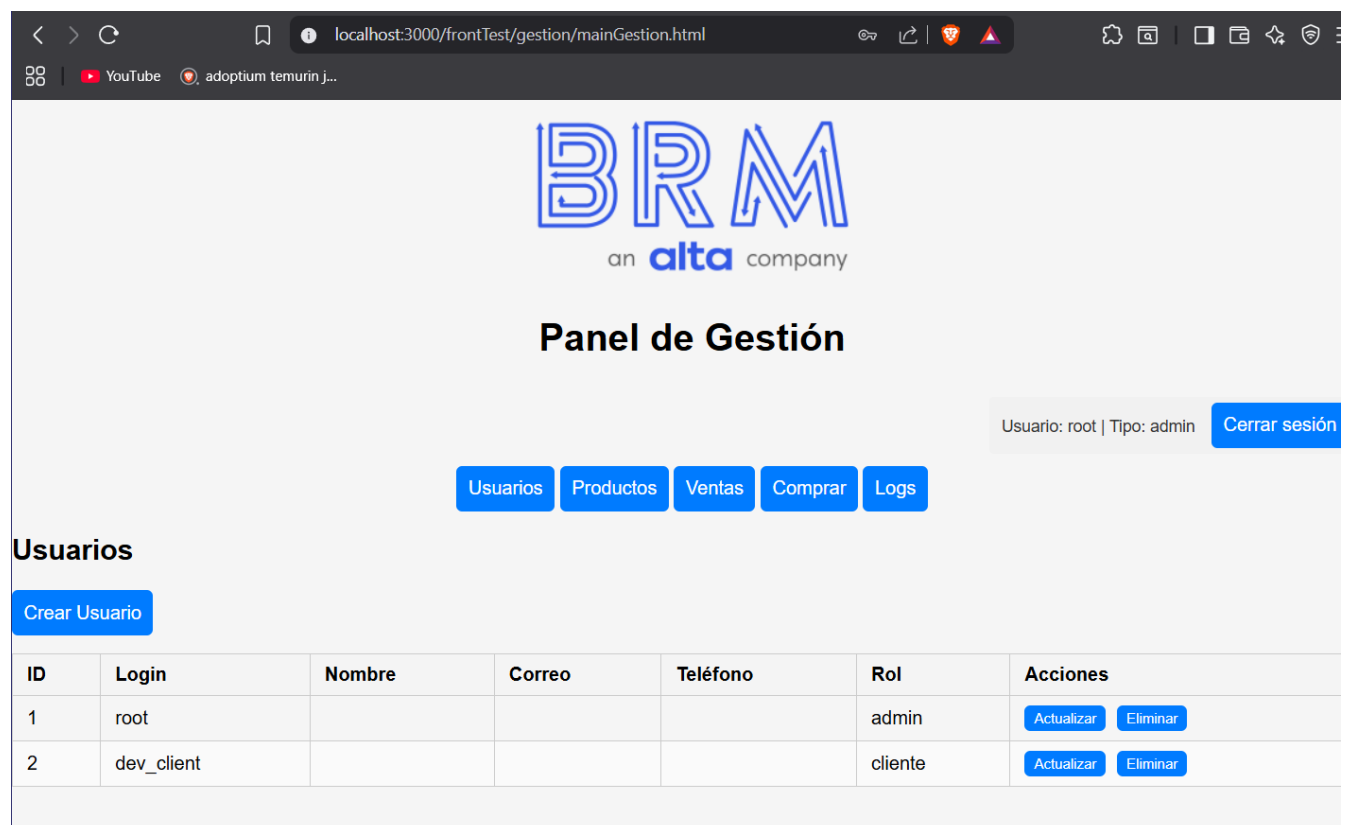


Generalmente se desarrollo un entorno grafico de pruebas para ver ejemplos de ejecución del backend, podemos acceder a este entorno haciendo un GET convencional desde un navegador a la url <http://localhost:3000/frontTest>, esta nos diseccionará al index.html de este apartado. Aquí podemos usar login 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.

an **alta** company

Panel de Gestión

Usuario: root | Tipo: admin [Cerrar sesión](#)

Usuarios

[Crear Usuario](#)

ID	Login	Nombre	Correo	Teléfono	Rol	Acciones
1	root					Actualizar Eliminar
2	dev_client				cliente	Actualizar Eliminar

La pantalla se actualizará y nos mostrará el proceso exitoso.

an **alta** company

Panel de Gestión

Usuario: root | Tipo: admin [Cerrar sesión](#)

[Usuarios](#) [Productos](#) [Ventas](#) [Comprar](#) [Logs](#)

Usuarios

[Crear Usuario](#)

ID	Login	Nombre	Correo	Teléfono	Rol	Acciones
1	root				admin	Actualizar Eliminar
2	dev_client				cliente	Actualizar Eliminar
4	NuevoUser22	Armando Mezas	armando839@gmail.com	3191938392	admin	Actualizar Eliminar

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

Usuarios

Usuario

Actualizar Usuario

Usuario: root | Tipo: admin

Cerrar sesión

Actualizar

Login	Nombre	Correo	Teléfono	Rol	Acciones
root				admin	Actualizar Eliminar
dev_client				cliente	Actualizar Eliminar
NuevoUser22	Armando Mezas	armando839@gmail.com	3191938392	admin	Actualizar Eliminar

BRM

an alta company

Panel de Gestión

Usuario: root | Tipo: admin

Cerrar sesión

Usuario actualizado con éxito

Usuarios Productos Ventas Comprar Logs

Usuarios

Crear Usuario

Login	Nombre	Correo	Teléfono	Rol	Acciones
root				admin	Actualizar Eliminar
dev_client	dev cliente actualización	unCorreo@comerciodevs.com		cliente	Actualizar Eliminar
NuevoUser22	Armando Mezas	armando839@gmail.com	3191938392	admin	Actualizar Eliminar

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

localhost:3000 dice
¿Seguro que quieres eliminar este usuario?

Aceptar Cancelar

Usuario creado con éxito

an **alta** company

Panel de Gestión

Usuario: root | Tipo: admin Cerrar sesión

Usuarios Productos Ventas Comprar Logs

Usuarios

Crear Usuario

Login	Nombre	Correo	Teléfono	Rol	Acciones
root				admin	Actualizar Eliminar
dev_client	dev cliente actualización	unCorreo@comerciodevs.com		cliente	Actualizar Eliminar
NuevoUser22	Armando Mezas	armando839@gmail.com	3191938392	admin	Actualizar Eliminar
UsuarioAEliminar	eliminar	eliminar@gmail.com	3999999	cliente	Actualizar Eliminar

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

Usuario eliminado correctamente

Se tiene el mismo comportamiento con productos.

Usuarios Productos Ventas Comprar Logs

Productos

Crear Producto

ID	Lote	Nombre	Precio	Stock	Fecha ingreso	Acciones
1	lote 90	teclado mecanico (producto Prueba)	239000.00	57	2/7/2017	Actualizar Eliminar
2	lote 200	Secadora de pelo plus	310000.00	30	2/12/2023	Actualizar Eliminar

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

Compras (Carrito)

Comprar (0)

Limpiar Carrito

ID	Lote	Nombre	Precio	Stock Disponible	Fecha Ingreso	Acciones
1	lote 90	teclado mecanico (producto Prueba)	239000.00	57	2/7/2017	Añadir al carrito
2	lote 200	Secadora de pelo plus	310000.00	30	2/12/2023	Añadir al carrito

Compras (Carrito)

Comprar (0) Limpiar Carrito

ID	Lote	Nombre	Precio	Stock Disponible	Fecha Ingreso	Acciones
1	lote 90	teclado mecanico (producto Prueba)	239000.00	57	2/7/2017	Añadir al carrito
2	lote 200	Secadora de pelo plus	310000.00	30	2/12/2023	Añadir al carrito

Añadir al Carrito

×

Producto: teclado mecanico (producto Prueba)

Cantidad:

[Añadir](#)

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

Compras (Carrito)

Comprar (2) Limpiar Carrito

ID	Lote	Nombre
1	lote 90	teclado mecanico

Carrito de Compras

×

ID Producto	Lote	Nombre	Precio Unitario	Cantidad	Acciones
1	lote 90	teclado mecanico (producto Prueba)	undefined	6	Eliminar
2	lote 200	Secadora de pelo plus	undefined	4	Eliminar

[Confirmar Compra](#)

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.

an **alta** company

el de Gestión

Usuario: root | Tipo: admin

Cerrar sesión

ProductosVentasComprarLogs

Precio	Stock Disponible	Fecha Ingreso	Acciones
239000.00	51	2/7/2017	Añadir al carrito
310000.00	26	2/12/2023	Añadir al carrito

UsuariosProductosVentasComprarLogs

Compras

ID Compra	Cliente	Correo	Teléfono	Fecha	Precio Total	Cantidad Productos	Acciones
1	root (-)	-	-	2/10/2025, 8:23:38	2674000.00	10	Ver Detalle

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

Detalle de la Compra

ID Compra: 1

Cliente: root (-)

Correo: -

Teléfono: -

Fecha: 2/10/2025, 8:23:38

Precio Total: 2674000.00

Cantidad Productos: 10

Productos de esta compra:

ID Producto	Lote	Nombre	Precio Unitario	Cantidad
1	lote 90	teclado mecanico (producto Prueba)	239000.00	6
2	lote 200	Secadora de pelo plus	310000.00	4

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

Usuario: root | Tipo: admin

Cerrar sesión

Usuarios

Productos

Ventas

Comprar

Logs

Registros del sistema

<

>

ID	Fecha	URL	Tipo	Acciones	
149	2/10/2025, 8:24:04	/buy/list/clients	GET	200	<div>Ver detalle</div>
145	2/10/2025, 8:23:38	/cart/buy	POST	200	<div>Ver detalle</div>
146	2/10/2025, 8:23:38	/products	GET	200	<div>Ver detalle</div>
147	2/10/2025, 8:23:38	/cart	GET	200	<div>Ver detalle</div>
148	2/10/2025, 8:23:38	/cart	GET	304	<div>Ver detalle</div>
144	2/10/2025, 8:22:48	/cart	GET	200	<div>Ver detalle</div>
142	2/10/2025, 8:22:47	/products	GET	200	<div>Ver detalle</div>
143	2/10/2025, 8:22:47	/cart	GET	200	<div>Ver detalle</div>

Detalle del Log

ID: 149

Fecha: 2/10/2025, 8:24:04

URL: /buy/list/clients

Tipo: GET

Status: 200

Contenido de la petición:

N/A

Contenido de la respuesta:

```
{
  "historial": [
    {
      "id_compra": 1,
      "fecha_compra": "2025-10-02T13:23:38.000Z",
      "cliente": {
        "id": 1,
        "nombre_login": "root",
        "nombre": null,
        "correo": null,
        "telefono": null
      },
      "precio_total": "2674000.00",
      "cantidad_total_productos": 10,
      "productos": [
        {
          "id_producto": 1,
          "nombre_lote": "lote 90",
          "nombre": "teclado mecanico (producto Prueba)",
          "precio_unitario": "239000.00",
          "cantidad": 6
        },
        {
          "id_producto": 2,
          "nombre_lote": "lote 200",
          "nombre": "Secadora de pelo plus",
          "precio_unitario": "310000.00",
          "cantidad": 4
        }
      ]
    }
  ]
}
```

El contenido de la petición son los datos que enviaron en la petición y el contenido de respuesta es lo que respondió 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