

Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Licenciatura en Sistemas - 2023

Diseño de aplicaciones 2

**Obligatorio I (Evidencia de la ejecución de las pruebas de la API
con Postman.)**

**Agustín Fioritto
Joaquín Calvo
Chiara Sosa**

**Nro est. 238927
Nro est. 203832
Nro est. 241699**

ÍNDICE

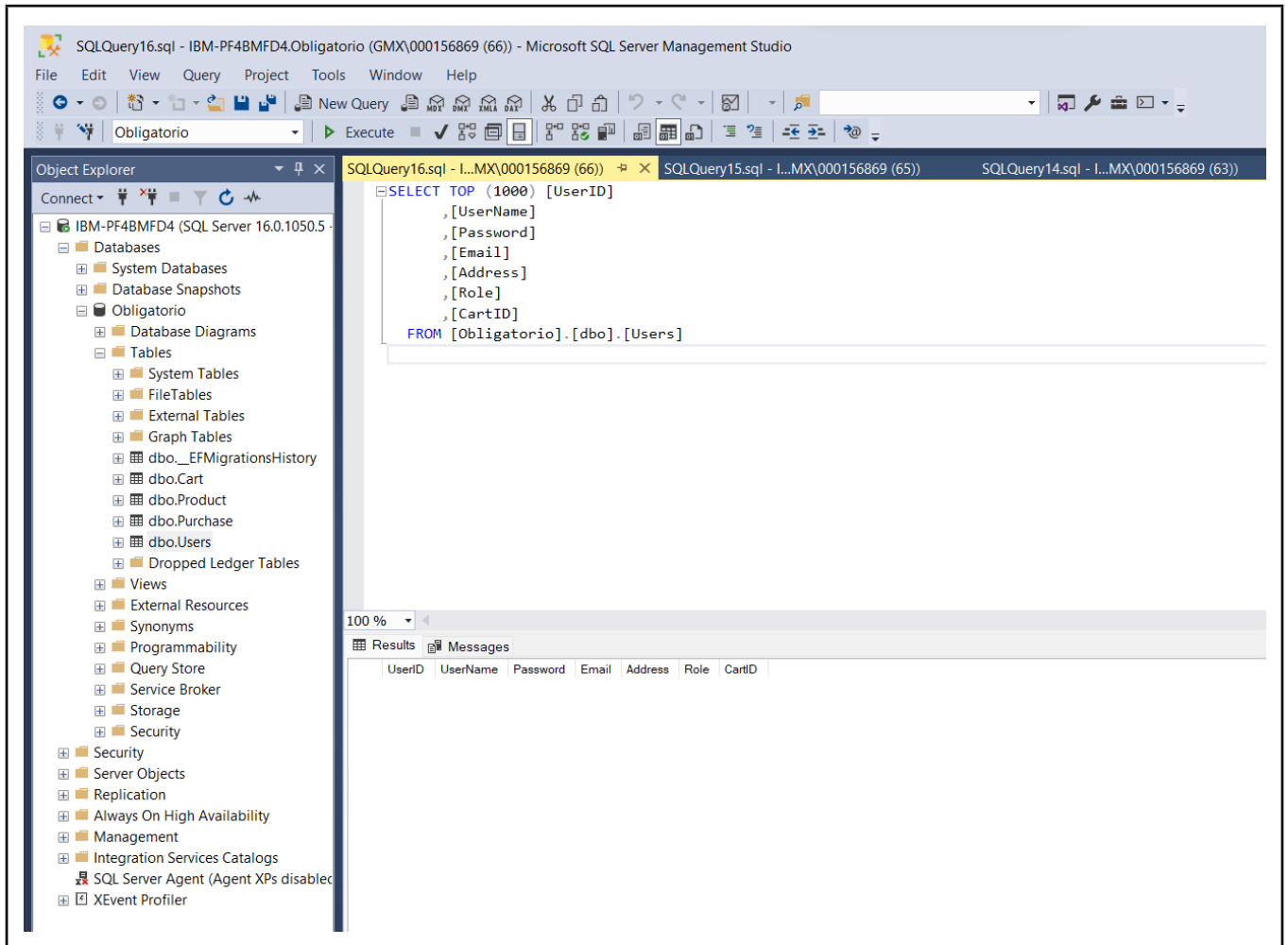
Evidencia Web API (video).....	3
Evidencia Operaciones API.....	3

Evidencia Web API (video)

Link al video: <https://youtu.be/9x2ucmy6ieM>

Evidencia Operaciones API

Estado inicial de la tabla Users en la BD:



- Endpoint:

POST /api/users: Registra un nuevo usuario en el sistema.

Parameters: User (Cuerpo de la solicitud): Datos del usuario a registrar.

Responses:

200 OK: Registro exitoso.

400 Bad Request: Error al registrar el usuario.

URL: <https://localhost:7004/api/users/>

Prueba 1: Se registra un usuario válido.

POSThttps://localhost:7004/api/users

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedorawbinaryGraphQLJSON

```
1 {
2   "userName": "UsuarioAdministrador",
3   "password": "ContraseñaAdministrador",
4   "email": "usuarioAdministrador@gmail.com",
5   "address": "DireccionAdministrador",
6   "role": "Administrador",
7   "purchases": null,
8   "cart": null
9 }
10
```

BodyCookiesHeaders (4)Test Results

PrettyRawPreviewVisualizeText

1 Usuario registrado exitosamente.

SQLQuery16.sql - L:\MX\000156869 (66)SQLQuery15.sql - L:\MX\000156869 (65)SQLQuery14.sql - L:\MX\000156869 (64)

```
SELECT TOP (1000) [UserID]
,[UserName]
,[Password]
,[Email]
,[Address]
,[Role]
,[CartID]
FROM [Obligatorio].[dbo].[Users]
```

100 %

ResultsMessages

	UserID	UserName	Password	Email	Address	Role	CartID
1	27	UsuarioAdministrador	ContraseñaAdministrador	usuarioAdministrador@gmail.com	DireccionAdministrador	Administrador	NULL

Prueba 2: Se registra un usuario invalido.

POSThttps://localhost:7004/api/users

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedorawbinaryGraphQLJSON

```
1 {
2   "userName": "",
3   "password": "",
4   "email": "usuarioMauro@gmail.com",
5   "address": "DireccionMauro",
6   "role": "Comprador",
7   "purchases": null,
8   "cart": null
9 }
10
```

BodyCookiesHeaders (4)Test Results

PrettyRawPreviewVisualizeText

1 Error al registrar el usuario: Usuario inválido

SQLQuery16.sql - L:\MX\000156869 (66)SQLQuery15.sql - L:\MX\000156869 (65)SQLQuery14.sql - L:\MX\000156869 (64)

```
SELECT TOP (1000) [UserID]
,[UserName]
,[Password]
,[Email]
,[Address]
,[Role]
,[CartID]
FROM [Obligatorio].[dbo].[Users]
```

100 %

ResultsMessages

	UserID	UserName	Password	Email	Address	Role	CartID
1	27	UsuarioAdministrador	ContraseñaAdministrador	usuarioAdministrador@gmail.com	DireccionAdministrador	Administrador	NULL
2	28	Actualizado	ContraseñaActualizada	usuarioActualizada@gmail.com	DireccionActualizada	Comprador	NULL
3	29	InformacionActualizada	InformacionActualizada	InformacionActualizada@gmail.com	DireccionActualizada	Comprador	NULL
4	30	UsuarioDemo	ContraseñaDemo	usuarioDemo@gmail.com	DireccionDemo	Comprador	NULL
5	32	Chiara	ContraseñaChiara	usuarioChiara@gmail.com	DireccionChiara	Comprador	NULL
6	33	Joaquin	ContraseñaJoaquin	usuarioJoaquin@gmail.com	DireccionJoaquin	Comprador	NULL
7	34	Marcos	ContraseñaMarcos	usuarioMarcos@gmail.com	DireccionMarcos	Comprador	NULL
8	35	Andres	ContraseñaAndres	usuarioAndres@gmail.com	DireccionAndres	Comprador	NULL
9	36	Mauro	ContraseñaMauro	usuarioMauro@gmail.com	DireccionMauro	Comprador	NULL

Notas:

Para las pruebas que se documentan a continuación se registraron nuevos usuarios en la base de datos. La tabla Users de la BD se ilustra en la imagen siguiente

Por otro lado, para las pruebas que requerían un usuario loggeado con rol administrador se utilizó el user:

```
{
  "userid": 27,
  "userName": "UsuarioAdministrador",
  "password": "ContraseñaAdministrador",
  "email": "usuarioAdministrador@gmail.com",
  "address": "DireccionAdministrador",
  "role": "Administrador",
  "purchases": null,
  "cart": null
}
```

- Endpoints:

POST /api/users/login: Inicia sesión de un usuario registrado.

Parameters: email (en el cuerpo de la solicitud): Correo electrónico del usuario, password (en el cuerpo de la solicitud): Contraseña del usuario.

Responses:

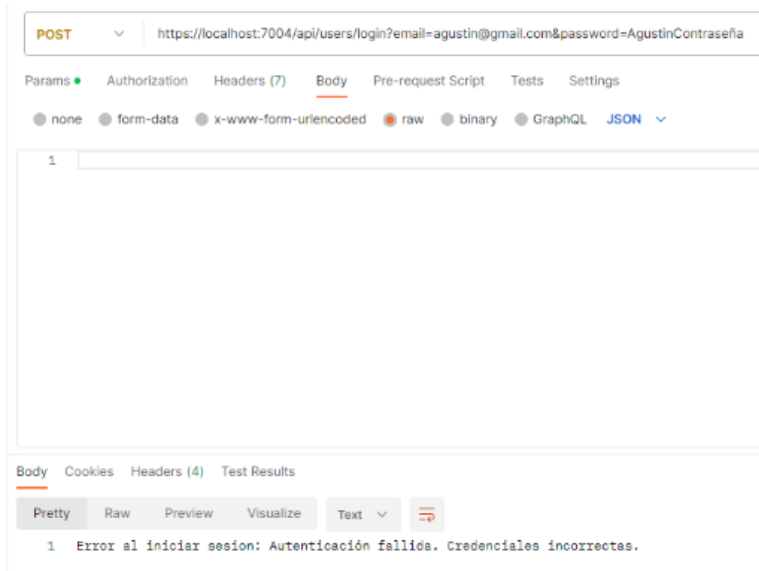
200 OK: Inicio de sesión exitoso.

400 Bad Request: Error al iniciar sesión.

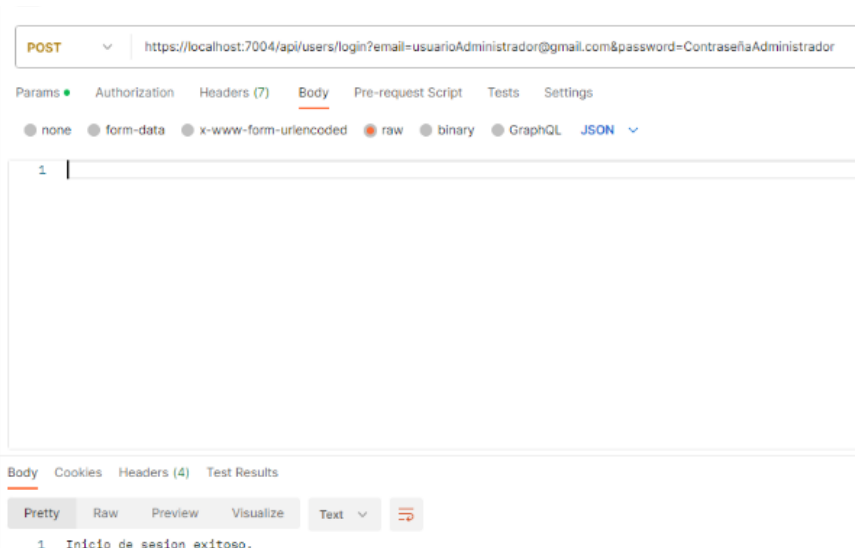
401 Unauthorized: Autenticación fallida.

URL: https://localhost:7004/api/users/login

Prueba 1: Se intenta hacer login de un usuario invalido.



Prueba 2: Se hace login de un usuario válido, registrado en el sistema, con rol administrador.



- Endpoints:

GET `/api/users`: Obtiene la lista de todos los usuarios registrados en el sistema.

Responses:

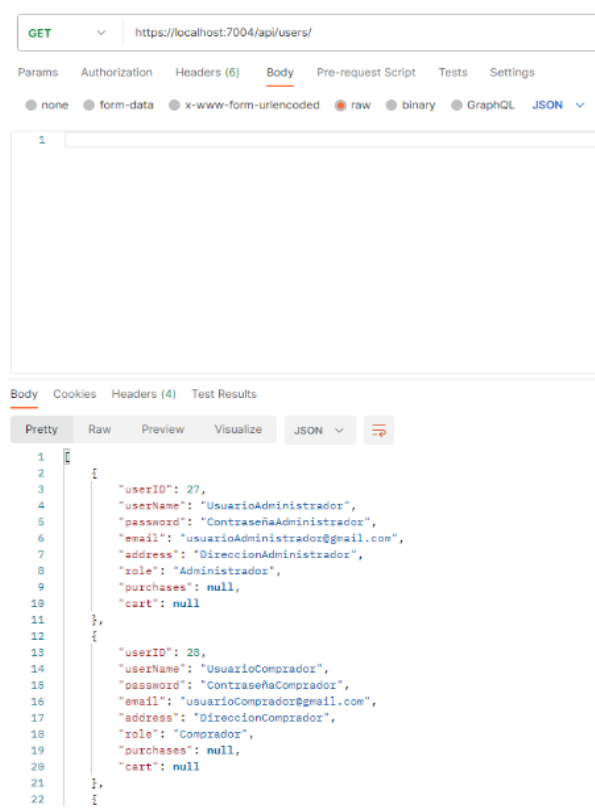
200 OK: Lista de usuarios.

400 Bad Request: Error al obtener usuarios.

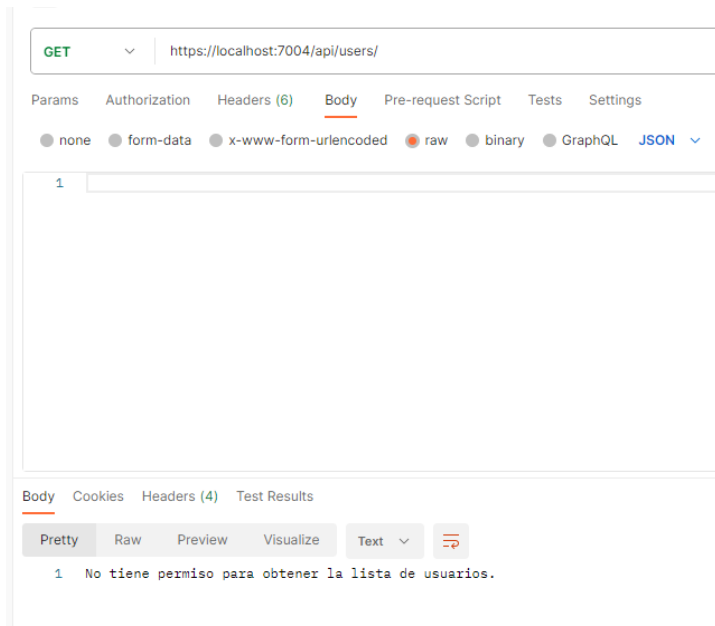
401 Unauthorized: No tiene permiso para obtener la lista de usuarios.

URL: `https://localhost:7004/api/users/`

Prueba 1: Usuario loggeado tiene Rol de Administrador.



Prueba 2: El usuario loggeado no es administrador:



- EndPoints:

GET /api/users/{id}: Obtiene un usuario por su ID.

Parameters: id (en la URL): ID del usuario.

Responses:

200 OK: Usuario encontrado.

400 Bad Request: Error al obtener el usuario.

404 Not Found: Usuario no encontrado.

URL: https://localhost:7004/api/users/{id}

Prueba 1: Se obtiene usuario con id 31.

GET https://localhost:7004/api/users/31

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▾

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize **JSON** ▾

```
1  {
2    "userID": 31,
3    "userName": "Agustin",
4    "password": "ContraseñaAgustin",
5    "email": "usuarioAgustin@gmail.com",
6    "address": "DireccionAgustin",
7    "role": "Comprador",
8    "purchases": null,
9    "cart": null
10 }
```

Prueba 2: Se intenta obtener un usuario con id invalido.

GET https://localhost:7004/api/users/0

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (4) Test Results 🚫 Status: 400 Bad Request Time: 1198 ms

Pretty Raw Preview Visualize **Text** ▾

1 Error al obtener el usuario: ID de usuario inválido.

- EndPoints

Delete `/api/users/{id}`: Elimina un usuario registrado en el sistema por su ID.

Parameters: id (en la URL): ID del usuario.

Responses:

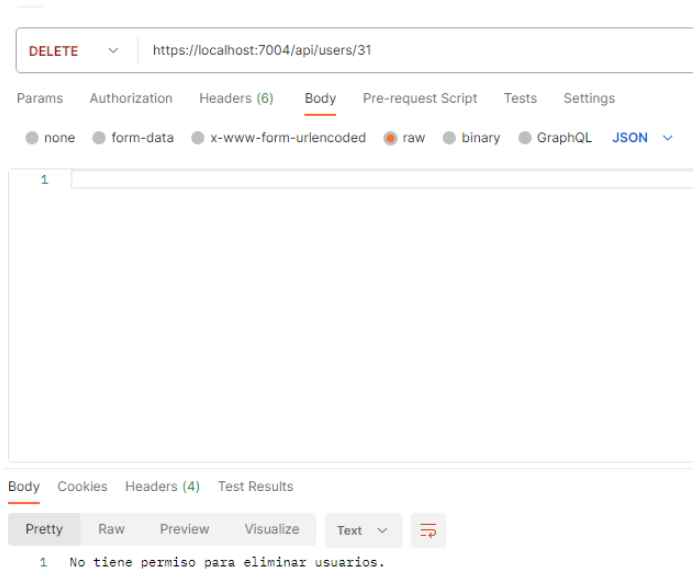
204 OK: El usuario se eliminó con éxito.

400 Bad Request: Error en la solicitud o al eliminar el usuario.

404 Not Found: El usuario con el id especificado no se encontró.

URL: `https://localhost:7004/api/users/{id}`

Prueba 1: Usuario loggeado no posee rol Administrador.



DELETE ▼ | https://localhost:7004/api/users/31

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

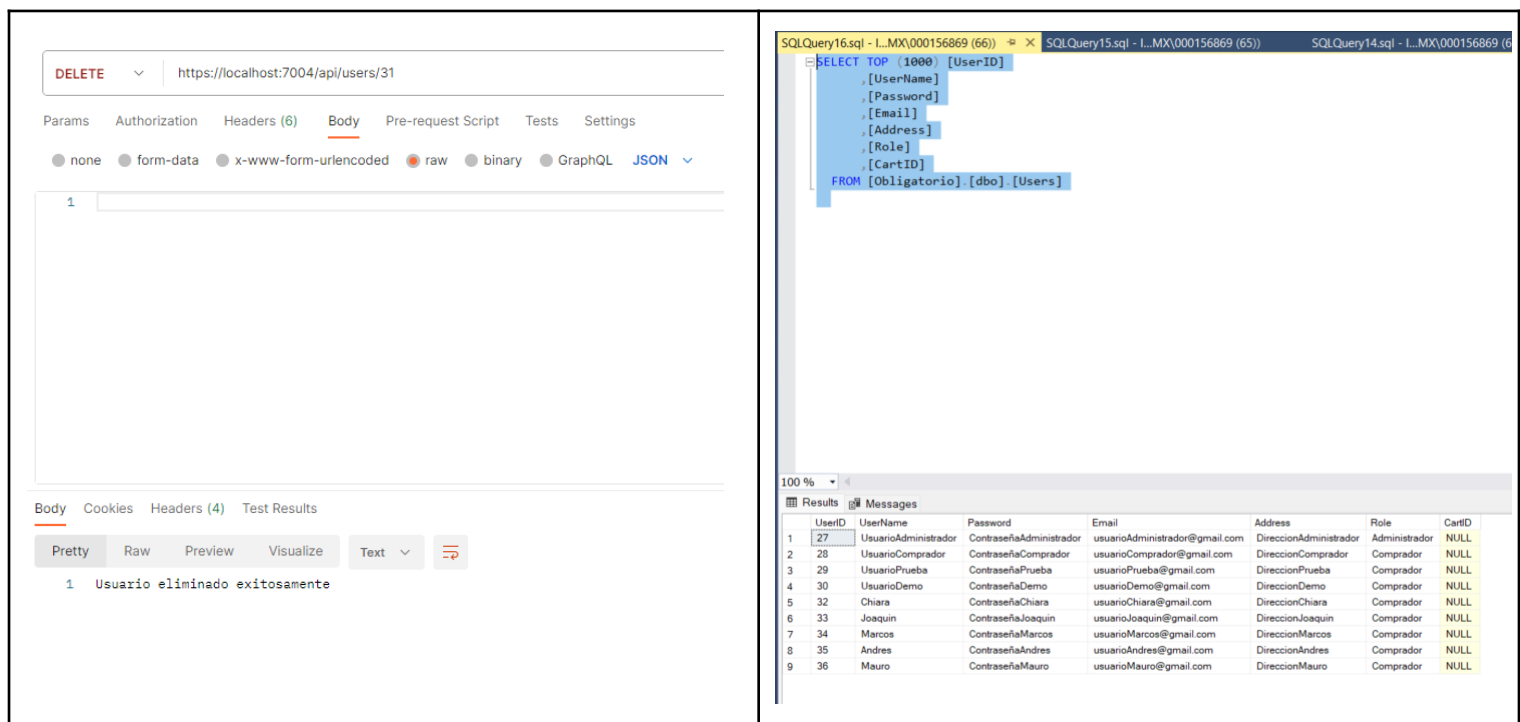
1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text ▼ ≡

1 No tiene permiso para eliminar usuarios.

Prueba 2: Usuario loggeado con rol administrador:



DELETE ▼ | https://localhost:7004/api/users/31

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text ▼ ≡

1 Usuario eliminado exitosamente

SQLQuery16.sql - L_MX\000156869 (66) X SQLQuery15.sql - L_MX\000156869 (65) SQLQuery14.sql - L_MX\000156869 (6)

```
SELECT TOP (1000) [UserID], [UserName], [Password], [Email], [Address], [Role], [CartID] FROM [Obligatorio].[dbo].[Users]
```

100 % ▼

Results Messages

	UserID	UserName	Password	Email	Address	Role	CartID
1	27	UsuarioAdministrador	ContraseñaAdministrador	usuarioAdministrador@gmail.com	DireccionAdministrador	Administrador	NULL
2	28	UsuarioComprador	ContraseñaComprador	usuarioComprador@gmail.com	DireccionComprador	Comprador	NULL
3	29	UsuarioPrueba	ContraseñaPrueba	usuarioPrueba@gmail.com	DireccionPrueba	Comprador	NULL
4	30	UsuarioDemo	ContraseñaDemo	usuarioDemo@gmail.com	DireccionDemo	Comprador	NULL
5	32	Chiara	ContraseñaChiara	usuarioChiara@gmail.com	DireccionChiara	Comprador	NULL
6	33	Joaquin	ContraseñaJoaquin	usuarioJoaquin@gmail.com	DireccionJoaquin	Comprador	NULL
7	34	Marcos	ContraseñaMarcos	usuarioMarcos@gmail.com	DireccionMarcos	Comprador	NULL
8	35	Andres	ContraseñaAndres	usuarioAndres@gmail.com	DireccionAndres	Comprador	NULL
9	36	Mauro	ContraseñaMauro	usuarioMauro@gmail.com	DireccionMauro	Comprador	NULL

- Endpoints:

PUT /api/users/UpdateUserProfile: Actualiza el perfil de un usuario registrado.

Parameters: user (en el cuerpo de la solicitud): Datos del usuario a actualizar.

Responses:

200 OK: Perfil del usuario actualizado exitosamente.

400 Bad Request: Error al actualizar el perfil del usuario.

URL: <https://localhost:7004/api/users/UpdateUserProfile>

Prueba 1: Se intenta actualizar un usuario que no es el loggeado. Esta operación permite actualizar solo el perfil del usuario loggeado.

PUT <https://localhost:7004/api/users/UpdateUserProfile>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "userid": 29,
3   "username": "Actualizado",
4   "password": "ContraseñaActualizada",
5   "email": "usuarioActualizada@gmail.com",
6   "address": "DireccionActualizada",
7   "role": "Comprador",
8   "purchases": null,
9   "cart": null
10 }
11
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text

1 El id del usuario a actualizar no coincide con el usuario loggeado.

Prueba 2: Se actualiza el usuario loggeado.

PUT <https://localhost:7004/api/users/UpdateUserProfile>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "userid": 28,
3   "username": "Actualizado",
4   "password": "ContraseñaActualizada",
5   "email": "usuarioActualizada@gmail.com",
6   "address": "DireccionActualizada",
7   "role": "Comprador",
8   "purchases": null,
9   "cart": null
10 }
11
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "userid": 28,
3   "username": "Actualizado",
4   "password": "ContraseñaActualizada",
5   "email": "usuarioActualizada@gmail.com",
6   "address": "DireccionActualizada",
7   "role": "Comprador",
8   "purchases": null,
9   "cart": null
10 }
```

SQLQuery16.sql - L_MX\000156869 (66)

```
SELECT TOP (1000) [UserID]
      ,[UserName]
      ,[Password]
      ,[Email]
      ,[Address]
      ,[Role]
      ,[CartID]
FROM [Obligatorio].[dbo].[Users]
```

	UserID	UserName	Password	Email	Address	Role	CartID
1	27	UsuarioAdministrador	ContraseñaAdministrador	usuarioAdministrador@gmail.com	DireccionAdministrador	Administrador	NULL
2	28	Actualizado	ContraseñaActualizada	usuarioActualizada@gmail.com	DireccionActualizada	Comprador	NULL
3	29	UsuarioPrueba	ContraseñaPrueba	usuarioPrueba@gmail.com	DireccionPrueba	Comprador	NULL
4	30	UsuarioDemo	ContraseñaDemo	usuarioDemo@gmail.com	DireccionDemo	Comprador	NULL
5	32	Chiara	ContraseñaChiara	usuarioChiara@gmail.com	DireccionChiara	Comprador	NULL
6	33	Joaquin	ContraseñaJoaquin	usuarioJoaquin@gmail.com	DireccionJoaquin	Comprador	NULL
7	34	Marcos	ContraseñaMarcos	usuarioMarcos@gmail.com	DireccionMarcos	Comprador	NULL
8	35	Andres	ContraseñaAndres	usuarioAndres@gmail.com	DireccionAndres	Comprador	NULL
9	36	Mauro	ContraseñaMauro	usuarioMauro@gmail.com	DireccionMauro	Comprador	NULL

- Endpoints:

PUT /api/users/UpdateUserInfo: Actualiza la información de un usuario registrado (solo para administradores).

Parameters: user (en el cuerpo de la solicitud): Datos del usuario a actualizar.

Responses:

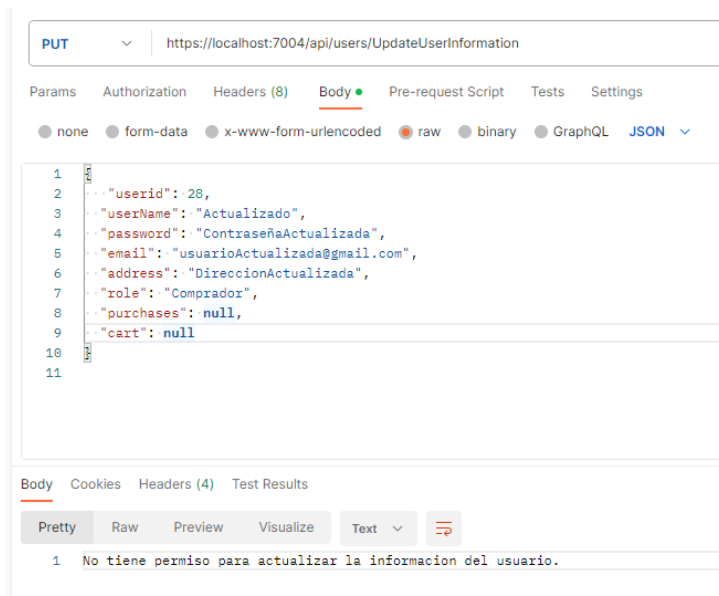
200 OK: Información del usuario actualizada exitosamente.

400 Bad Request: Error al actualizar la información del usuario.

401 Unauthorized: No tiene permiso para actualizar la información del usuario.

URL: https://localhost:7004/api/users/UpdateUserInfo

Prueba 1: Usuario loggeado no posee rol administrador.



Prueba 2: Usuario loggeado posee rol administrador.

The screenshot shows a REST client interface with a PUT request to `https://localhost:7004/api/users/UpdateUserInfo`. The body is a JSON object with the following fields: `userid: 29`, `username: "InformacionActualizada"`, `password: "InformacionActualizada"`, `email: "InformacionActualizada@gmail.com"`, `address: "DireccionActualizada"`, `role: "Comprador"`, `purchases: null`, and `cart: null`. The response is a 200 OK status.

The screenshot shows a SQL query in SQL Server Enterprise Manager. The query is: `SELECT TOP (1000) [UserID], [UserName], [Password], [Email], [Address], [Role], [CartID] FROM [Obligatorio].[dbo].[Users]`. The results table shows 9 rows of user data:

UserID	UserName	Password	Email	Address	Role	CartID
27	UsuarioAdministrador	ContraseñaAdministrador	usuarioAdministrador@gmail.com	DireccionAdministrador	Administrador	NULL
28	Actualizado	ContraseñaActualizada	usuarioActualizada@gmail.com	DireccionActualizada	Comprador	NULL
29	InformacionActualizada	InformacionActualizada	informacionActualizada@gmail.com	DireccionActualizada	Comprador	NULL
30	UsuarioDemo	ContraseñaDemo	usuarioDemo@gmail.com	DireccionDemo	Comprador	NULL
32	Chiara	ContraseñaChiara	usuarioChiara@gmail.com	DireccionChiara	Comprador	NULL
33	Joaquin	ContraseñaJoaquin	usuarioJoaquin@gmail.com	DireccionJoaquin	Comprador	NULL
34	Marcos	ContraseñaMarcos	usuarioMarcos@gmail.com	DireccionMarcos	Comprador	NULL
35	Andres	ContraseñaAndres	usuarioAndres@gmail.com	DireccionAndres	Comprador	NULL
36	Mauro	ContraseñaMauro	usuarioMauro@gmail.com	DireccionMauro	Comprador	NULL

- Endpoints:

GET /api/users/AllPurchases: Obtiene todas las compras realizadas en el sistema (solo para administradores).

Responses:

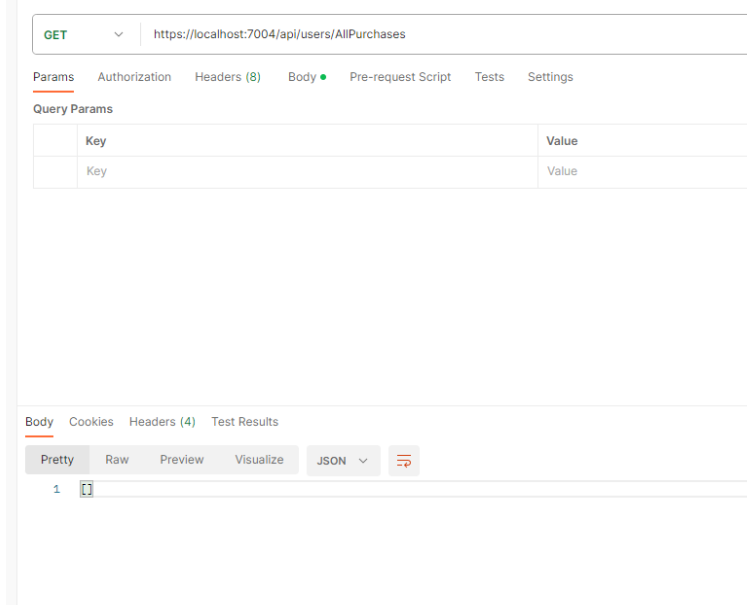
200 OK: Compras obtenidas exitosamente.

400 Bad Request: Error al obtener compras.

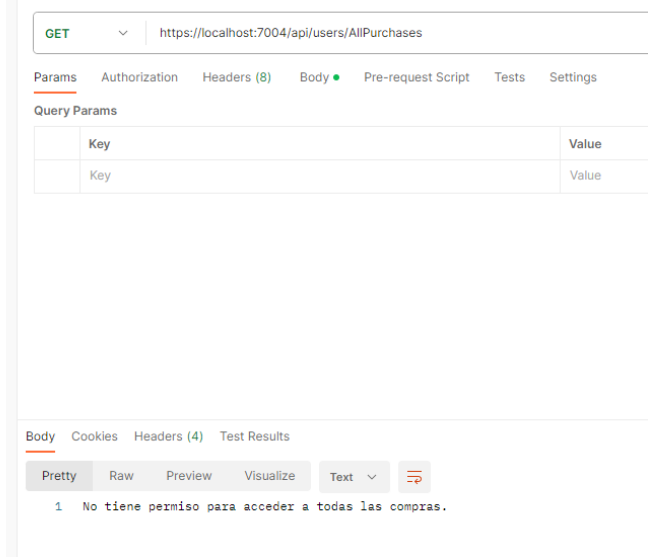
403 Forbidden: No tiene permiso para acceder a todas las compras (solo para usuarios no administradores).

URL: https://localhost:7004/api/users/AllPurchases

Prueba 1: Usuario loggeado posee rol Administrador. Devuelve una lista vacía.



Prueba 2: Usuario loggeado no posee rol administrador.



- Endpoints:

GET /api/users/GetPurchaseHistory/{id}: Obtiene el historial de compras de un usuario registrado por su ID.

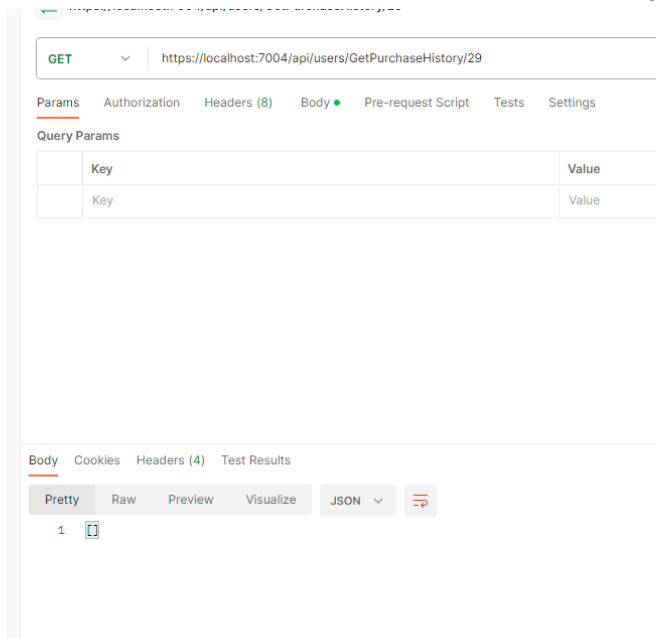
Parameters: id (en la URL): ID del usuario.

Responses:

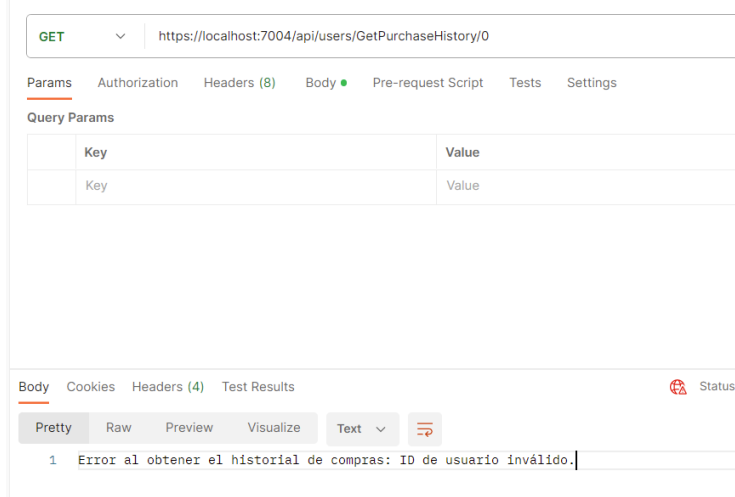
200 OK: Historial de compras obtenido exitosamente.

URL: <https://localhost:7004/api/users/GetPurchaseHistory/{id}>

Prueba 1: Se obtiene el historial de un usuario registrado en el sistema.



Prueba 2: Se intenta obtener el historial de un usuario invalido.



- EndPoints:

POST /api/productst: Registra un producto en el sistema.

Parameters: product(en el cuerpo de la solicitud): Datos del producto.

Responses:

200 Ok: Sesión cerrada exitosamente.

400 Bad Request: Error al registrar el producto

Prueba 1: Se inserta un producto incorrecto.

https://localhost:7004/api/products

POST https://localhost:7004/api/products

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "",
3   "price": 3,
4   "description": "Paraguas",
5   "brand": 4,
6   "category": 5,
7   "color": "Rosado"
8 }
```

Body Cookies Headers (4) Test Results Status: 400 Bad Request

Pretty Raw Preview Visualize Text

1 Error al registrar el producto: Uno de los datos es incorrecto

Prueba 2: Se inserta un producto válido.

https://localhost:7004/api/products

POST https://localhost:7004/api/products

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "Reloj",
3   "price": 3,
4   "description": "Reloj",
5   "brand": 4,
6   "category": 5,
7   "color": "Rosado"
8 }
```

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize Text

1 Producto registrado correctamente

- EndPoints:


Get /api/products/: Obtiene la lista de productos registrados en el sistema.

Responses:

200 Ok: Sesión cerrada exitosamente.


400 Bad Request: Error al obtener los productos

Prueba 1: Se obtiene la lista de productos.

 https://localhost:7004/api/products



GET https://localhost:7004/api/products

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** 

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON  

```
1  [
2    {
3      "productID": 2,
4      "name": "ProductoDemo",
5      "price": 45,
6      "description": "Descripcion",
7      "brand": 1,
8      "category": 0,
9      "color": "Amarillo"
10   },
11   {
12     "productID": 3,
13     "name": "ProductoYoutube",
14     "price": 45,
15     "description": "DescripcionYoutube",
16     "brand": 5,
17     "category": 10,
18     "color": "Rojo"
19   },
20   {
21     "productID": 4,
22     "name": "Producto",
23     "price": 13,
24     "description": "Descripcion",
25     "brand": 3,
26     "category": 4,
27     "color": "Rojo"
28   },
29 ]
30 {
31   "productID": 5,
32   "name": "Ropa",
33   "price": 1343,
```