

Descripción de la API trabajo Final

Api desplegada en Heroku: <https://rhg-final-coder.herokuapp.com/>

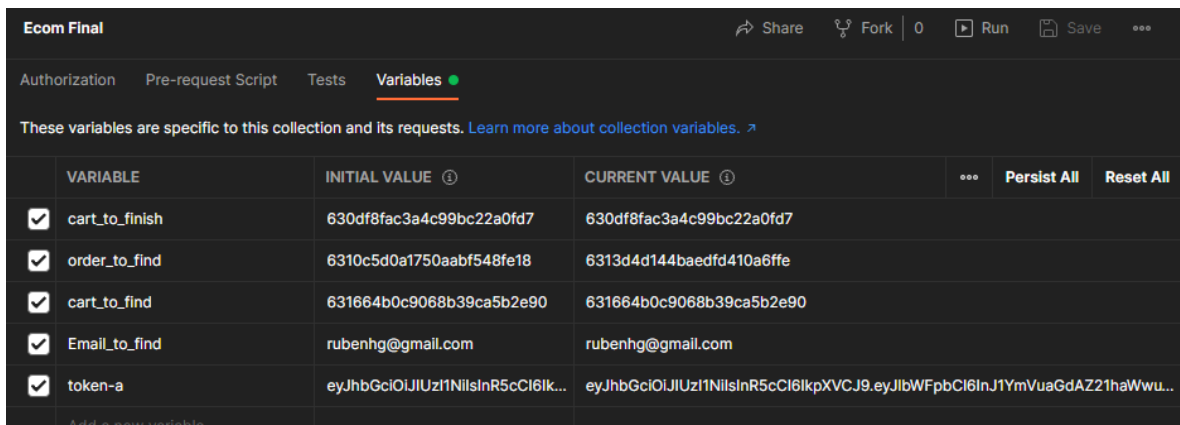
Configuración de ambientes en postman:

Variable URL

- DEV: <http://localhost:3000>
- PROD: <https://rhg-final-coder.herokuapp.com>

Collección Ecom Final

Variables para simplificar las pruebas



	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	cart_to_finish	630df8fac3a4c99bc22a0fd7	630df8fac3a4c99bc22a0fd7			
<input checked="" type="checkbox"/>	order_to_find	6310c5d0a1750aabf548fe18	6313d4d144baedfd410a6ffe			
<input checked="" type="checkbox"/>	cart_to_find	631664b0c9068b39ca5b2e90	631664b0c9068b39ca5b2e90			
<input checked="" type="checkbox"/>	Email_to_find	rubenhg@gmail.com	rubenhg@gmail.com			
<input checked="" type="checkbox"/>	token-a	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWVuaGdAZ21haWw...	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWVuaGdAZ21haWw...			

Endpoints

Auth

- Register: (/user/register) crea un usuario con un objeto en el body como parámetro y envía el mail al destinatario en la variable de entorno ADMIN_EMAIL (plantillas en Handlebars)

Nuevo Usuario ➡ Recibidos x



rubenhgodoy.dev@gmail.com

para mí ▼

Datos del nuevo registro:

Email: prueba1@prueba.com

← Responder

→ Reenviar

- Login (/user/login) genera el token para el usuario registrado

El token resultante se debe establecer en la variable "token-a" de la colección para que los request de los otros endpoints sean autenticados.

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFnZWVudGdAZ21hYWwY29tIiwiaWQiOiI2MzE2NWVhYmUxMTBiNTl1MWM5YTRmNjkiLCJpYXQ1OjE2NjI0MTQ4NzR9.BitEscotVYw1MtZUqdm748rYuaZ0Lu012YkISzrUW08o"
3 }
```

Ecom Final / products / **GetAll** [Save] [Send]

GET {{URL}}/products

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

Headers Hide auto-generated headers

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Postman-Token	<calculated when request is sent>				
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>				
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.29.2				
<input checked="" type="checkbox"/>	Accept	*/*				
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br				
<input checked="" type="checkbox"/>	Connection	keep-alive				
<input checked="" type="checkbox"/>	auth-token	{{token-a}}				

Products

- GetAll (/products) devuelve un array de objetos con todos los productos
- GetById (/products/{{product_to_find}}) devuelve un producto por su Id

- GetByCategory(/products/category/{{category_to_find}}) devuelve la lista de productos con la category indicada como parámetro
- Delete (/products/{{product_to_delete}}) elimina el producto con el id indicada como parámetro
- POST /products crear un producto con el objeto enviado como parámetro
- PUT (/products) modifica el producto con los datos enviados en un objeto como parámetro
- DELETE (/products/{{product_to_delete}}) elimina el producto con el id enviado como parámetro (devuelve el id del producto borrado)

En el caso que se busque un producto por id, si no se encuentra se recibe el siguiente resultado:

```

Body Cookies Headers (9) Test Results
Pretty Raw Preview Visualize JSON
1  {
2    "message": "Producto no encontrado"
3  }
  
```

Carts

Se exponen endpoints para el CRUD de Cart, se explican las operaciones no triviales:

- Create POST /carts: crea el carrito con los datos enviados con un objeto como parámetro
- AddProduct (/carts/{{cart_to_find}}/addProduct) agrega un producto al carrito especificado, envía como parámetro un objeto con el id del producto y la cantidad. Si el producto ya existe en el carrito aumenta la cantidad en “quantity”
- Devuelve el objeto cart (notese que incluye el Price que no se envió como parámetro)

```

POST {{URL}}/carts/{{cart_to_find}}/addProduct
Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

1  {
2    "productId": "630dffac952dd0f5dc99c6e6",
3    "quantity": 1
4  }

Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON
1  {
2    "_id": "6316a66586e9ab6750baed2f",
3    "email": "rubenhg@gmail.com",
4    "date": "9/6/2022, 1:46:13 AM",
5    "products": [
6      {
7        "id": "630dffac952dd0f5dc99c6e6",
8        "quantity": 1,
9        "price": 200
10     }
11   ],
12   "deliveryAddress": "Cordoba 915",
13   "v": 0
  
```

- DeleteProduct (DELETE /carts/{{cart_to_find}}/product/{{product_to_delete_from_cart}}) elimina el producto del carrito.
- GetProductsByCart (/carts/{{cart_to_find}}/products) obtiene los productos del carrito (incluyendo los datos actuales del catalogo de productos y el precio al momento de agregar el producto)

Orders

- CreateFromCart (POST /orders/cart/{{cart_to_finish}}) crea un registro de orden a partir de un carrito. Envía el mail con los artículos, cantidades, precio y total (plantillas en Handlebars)

Nuevo pedido de rubenhg@gmail.com ➤ Recibidos x

 **rubenhgodoy.dev@gmail.com**
para mí ▼

Cliente:

Email: rubenhg@gmail.com

Productos

Código	Producto	Precio	Cantidad
002	producto2	200	1
003	producto3	200	1

Total:\$ 400

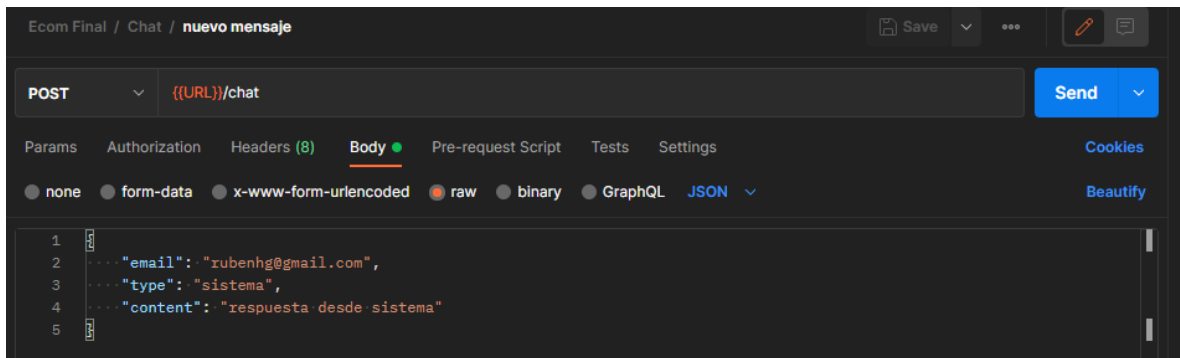
- GetByEmail (/orders/email/{{order_email_to_find}}) devuelve el listado de ordenes de un determinado email enviado como parámetro

Chat

- Nuevo mensaje (POST /chat) crea un mensaje con los datos establecidos en el objeto enviado como parámetro y devuelve la lista de mensajes para el email especificado
- GetByEmail (/chat/{{Email_to_find}}) devuelve la lista de mensajes para el email especificado

Implementación: cuando un cliente se conecta al websocket se crea una “room” identificada con su email. Al enviar un post a la ruta POST /chat con el objeto especificado se almacena el mensaje

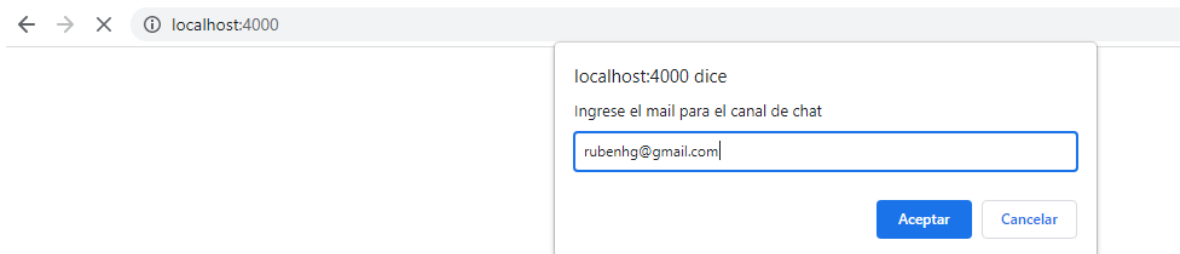
en la collection Messages de MongoDB y se transmite por socket a la “room” identificada con el email del cuerpo del request.



FrontEnd de Canal de Chal 1 a 1

Publicado en Heroku para probar en producción: <https://backend-rhg-coder.herokuapp.com/>

En forma local en la carpeta client.



1. Enviar mensaje desde FrontEnd

Chat FrontEnd

Chat FrontEnd

usuario: *mensaje 1 desde FE 9/6/2022, 1:05:26 PM*

2. Consulta a la api por email

The screenshot shows a web browser with a chat application and a REST client displaying the API response.

Chat Application:

- URL: `{{URL}}/chat/{{Email_to_find}}`
- Method: GET
- Send button
- Params: Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params table:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

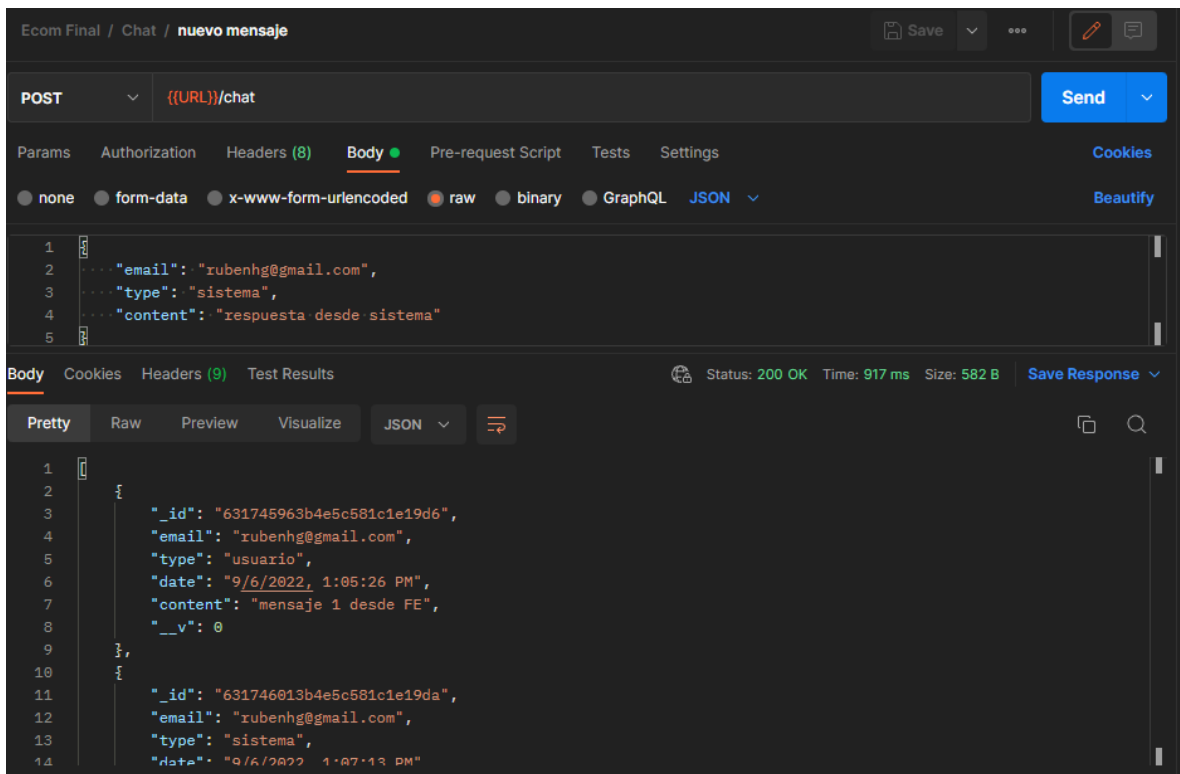
Body: Cookies, Headers (9), Test Results

Status: 200 OK Time: 760 ms Size: 427 B Save Response

Response Body (JSON):

```
{  "id": "631745963b4e5c581c1e19d6",  "email": "rubenhg@gmail.com",  "type": "usuario",  "date": "9/6/2022, 1:05:26 PM",  "content": "mensaje 1 desde FE",  "__v": 0}
```

2. Hacer respuesta desde la api



3. Se recibe la respuesta en el frontend por websocket



Info

- `getInfo (/info)` permite ver la configuración del servidor