

# Manual de Uso de la API

Este manual describe el funcionamiento de una API desarrollada con Flask y RethinkDB, que permite realizar operaciones CRUD sobre clientes, usuarios, documentos y casos legales (documentos). Además, la API almacena logs de las operaciones realizadas. A continuación, se detallan las rutas disponibles, los métodos HTTP correspondientes y la estructura de los datos esperados.

## Configuración Inicial

Antes de utilizar la API, se debe asegurar que los siguientes paquetes estén instalados:

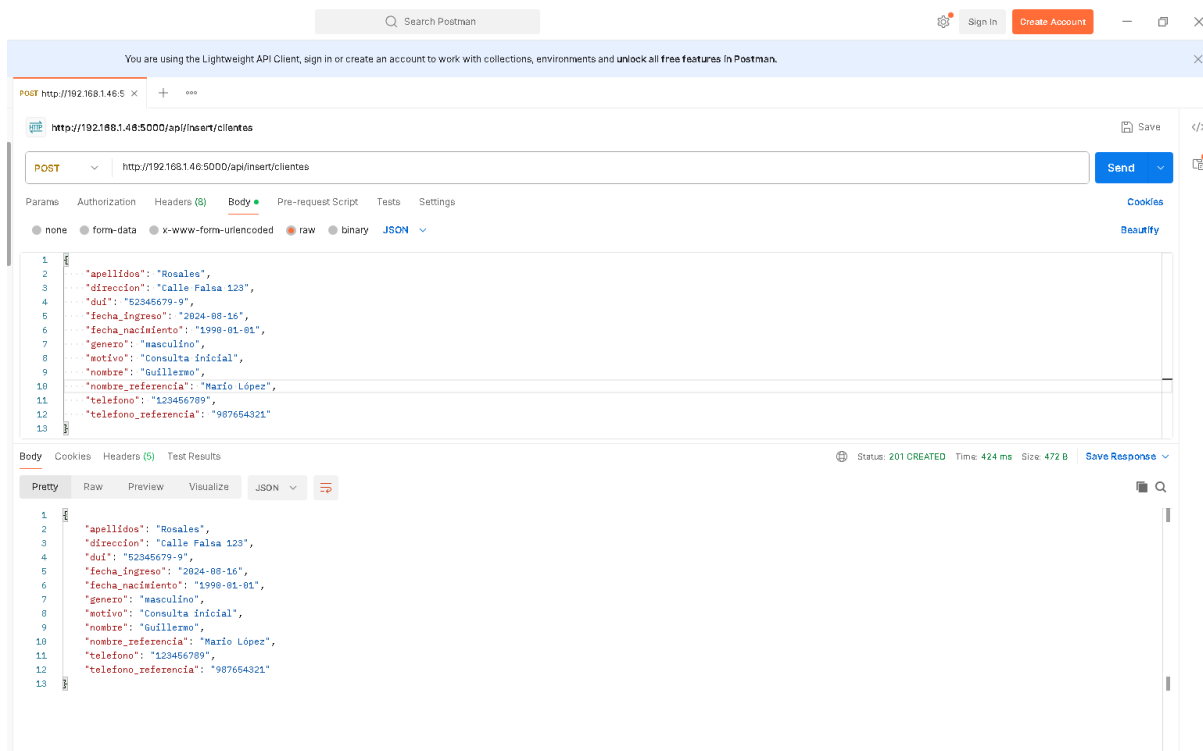
- `pip install flask rethinkdb werkzeug`

El archivo principal de la aplicación es “api.py”, que debe ser ejecutado para iniciar el servidor.

## Rutas Disponibles

### 1. Insertar Cliente

- **URL:** /api/insert/clientes
- **Método:** POST
- **Descripción:** Inserta un nuevo cliente en la base de datos.
- **Datos Esperados (JSON)**

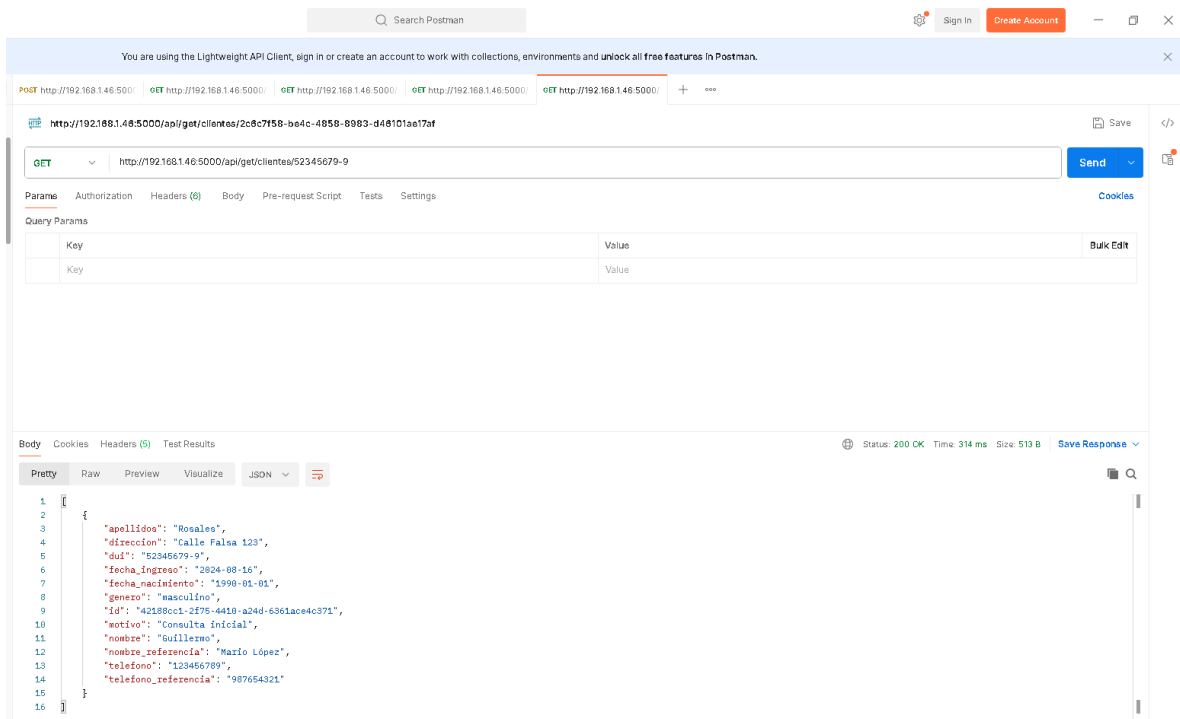


- **Respuesta:** Retorna un objeto JSON con los datos del cliente insertado y un código de estado 201.

```
osboxes@192.168.1.46:22 - Bitvise xterm - osboxes@osboxes: ~/proyecto.E8HtKP/app
(app) osboxes@osboxes:~/proyecto.E8HtKP/app$ flask --app api.py run --host=0.0.0.0
* Serving Flask app 'api.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.46:5000
Press CTRL+C to quit
192.168.1.36 - - [23/Aug/2024 09:01:28] "POST /api/insert/clientes HTTP/1.1" 201 -
```

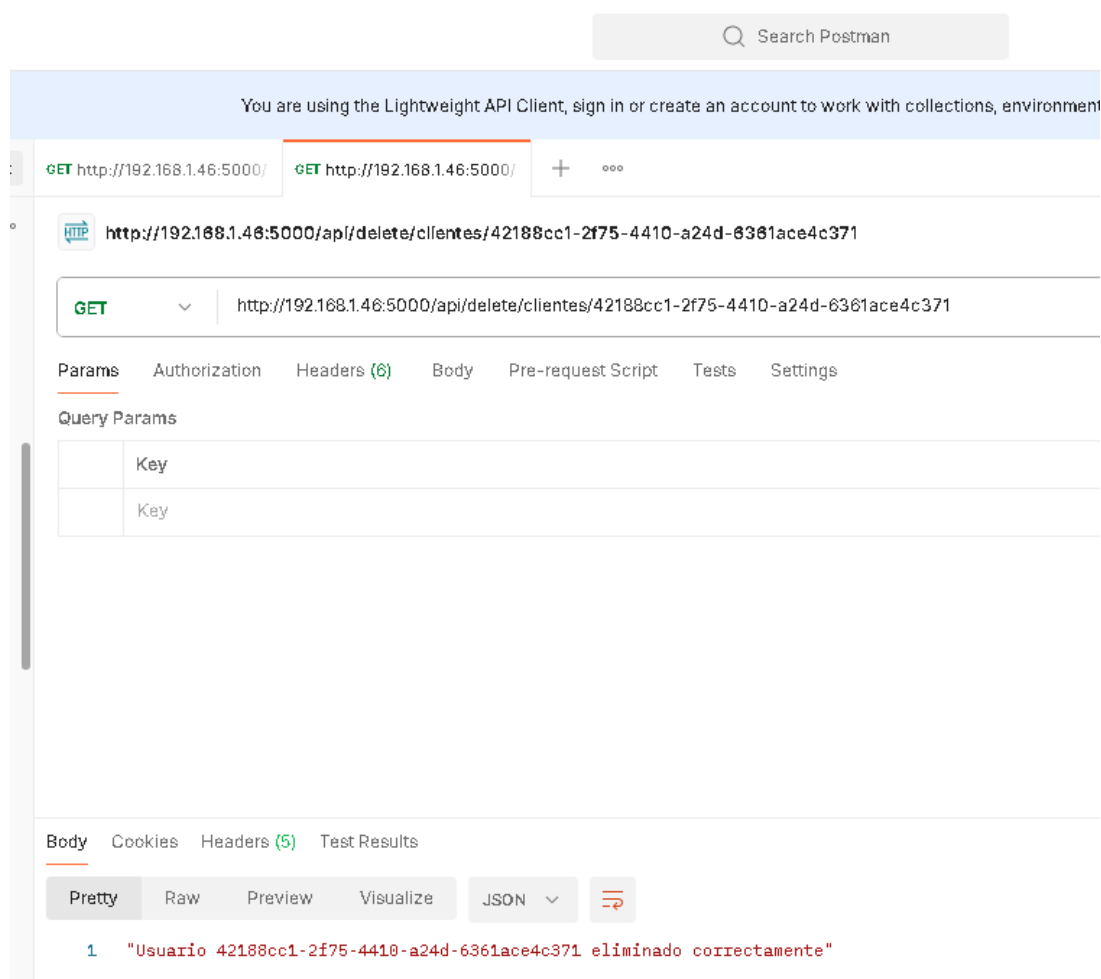
## 2. Obtener Cliente por ID (DUI)

- **URL:** /api/get/clientes/<id\_cliente>
- **Método:** GET
- **Descripción:** Retorna los datos de un cliente específico según su DUI.
- **Parámetro URL:**
  - id\_cliente: DUI del cliente.
- **Respuesta:** Un JSON con los datos del cliente.



### 3. Eliminar Cliente por ID

- **URL:** /api/delete/clientes/<id\_eliminar>
- **Método:** GET
- **Descripción:** Elimina un cliente de la base de datos según su número de ID.
- **Parámetro URL:**
  - id\_eliminar: número de ID.
- **Respuesta:** Un mensaje personalizado de confirmación.



#### 4. Actualizar Cliente por ID

- **URL:** /api/update/clientes/<id\_update>
- **Método:** POST
- **Descripción:** Actualiza los datos de un cliente existente.
- **Datos Esperados (JSON):**

The screenshot displays a REST client interface with the following details:

- URL:** `http://192.168.1.46:5000/api/update/clientes/64346ab8-bd1e-4c94-88d2-addd908c9175`
- Method:** POST
- Body:** A JSON object with the following fields:

```
1 {
2   "apellidos": "Morales",
3   "direccion": "Calle 5, av. 123",
4   "dui": "78945321-5",
5   "fecha_ingreso": "2024-08-16",
6   "fecha_nacimiento": "1990-01-01",
7   "genero": "masculino",
8   "motivo": "Consulta inicial",
9   "nombre": "Fernando",
10  "nombre_referencia": "Angelica Escobar",
11  "telefono": "123456789",
12  "telefono_referencia": "987654321"
13 }
```
- Response:** A single line of JSON:

```
1 "Cliente 64346ab8-bd1e-4c94-88d2-addd908c9175 actualizado correctamente"
```

- **Respuesta:** Se genera un mensaje personalizado de confirmación.

## 5. Insertar Usuario

- **URL:** /api/insertuser
- **Método:** POST
- **Descripción:** Inserta un nuevo usuario en la base de datos.
- **Datos Esperados (JSON):**

The screenshot displays a REST client interface. At the top, the URL is set to `http://192.168.1.46:5000/api/insertuser`. The method is **POST**. Below the URL bar, tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings are visible. The **Body** tab is selected, and the content type is set to **JSON**. The request body is a JSON object with the following fields:

```
1 {  
2   "nombre": "Prueba",  
3   "password": "prueba",  
4   "admin": "0",  
5   "abogado": "1",  
6   "cant_login": "0"  
7 }
```

At the bottom, the **Body** tab is selected in the response section, and the response is displayed in the **Pretty** view:

```
1 "Usuario ingresado"
```

- **Respuesta:** Un mensaje personalizado de confirmación.

## 6. Eliminar Usuario por ID

- **URL:** /api/delete/usuario/<id\_deleteuser>
- **Método:** GET
- **Descripción:** Elimina un usuario de la base de datos según su número de ID.
- **Parámetro URL:**
  - id\_deleteuser: numero de ID del usuario.

The screenshot displays a REST client interface with a GET request to the URL `http://192.168.1.46:5000/api/delete/usuario/e13c239c-e0b3-479f-9412-80af9ab20890`. The response body, shown in JSON format, contains a single message: `"Usuario e13c239c-e0b3-479f-9412-80af9ab20890 eliminado correctamente"`.

GET `http://192.168.1.46:5000/api/delete/usuario/e13c239c-e0b3-479f-9412-80af9ab20890`

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

Query Params

Key
Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 "Usuario e13c239c-e0b3-479f-9412-80af9ab20890 eliminado correctamente"
```

- **Respuesta:** Un mensaje personalizado de confirmación.

## 7. Obtener Todos los Usuarios

- **URL:** /api/get/usuario
- **Método:** GET
- **Descripción:** Retorna una lista de todos los usuarios en la base de datos.

The screenshot shows a REST client interface with a GET request to `http://192.168.1.46:5000/api/get/usuario`. The response is a JSON array of three user objects, displayed in the 'Body' tab.

**Query Params**

Key	Value
Key	Value

**Body**

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "abogado": "0",
4     "cant_login": 0,
5     "id": "299e41a5-aed1-488b-8225-7fc60c446f8c",
6     "password": "bcrypt:32768:8:1$jVGHuQaL31ZiPyek$f240d25e8bfad0edb77f9ead2135cb57776dda9031e672be190fb68b956929b01a04a46cab7b306010",
7     "rol": "1",
8     "username": "04"
9   },
10  {
11    "abogado": "1",
12    "cant_login": 0,
13    "id": "dac7db83-ef98-49f2-a6f9-b8b49c188b00",
14    "password": "bcrypt:32768:8:1$Yk4f2Wjggig6WpS$65c840466c09d929f2d58bf395b79f1a0182a8d3ba092a9f5a7ab5230633b5a10c7173a3087a69f46acb",
15    "rol": "0",
16    "username": "03"
17  },
18  {
19    "abogado": "1",
20    "cant_login": 0,
```

- **Respuesta:** Un JSON con los datos de todos los usuarios.

## 8. Obtener Todos los Documentos

- **URL:** /api/get/documentos
- **Método:** GET
- **Descripción:** Retorna una lista de todos los documentos almacenados.

GET http://192.168.1.46:5000/ POST http://192.168.1.46:5000/ GET http://192.168.1.46:5000/ POST http://192.168.1.46:5000/

<http://192.168.1.46:5000/api/get/clientes/2c8c7f58-be4c-4858-8983-d48101ae17af>

GET http://192.168.1.46:5000/api/get/documentos

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

Query Params

Key
Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "abogado": "Cartagena",
4     "caso": "regimen",
5     "cliente": "Pablo",
6     "comentario": "Documento para revision caso #2",
7     "id": "1b2e81c3-c7ce-47e8-9a35-1ccc595160e2"
8   },
9   {
10    "abogado": "Cartagena",
11    "caso": "Inmuebles",
12    "cliente": "Denys Eduardo ",
13    "comentario": "prueba de fecha",
14    "fecha_subida": "2024-08-23 01:27:40",
15    "id": "07ea4104-653b-4147-9561-19fc10b777b7"
16  },
17  {
18    "abogado": "Marroquín",
19    "caso": "traspasos",
20    "cliente": "Rafael",
```

- **Respuesta:** Un JSON con los datos de todos los documentos.



## 9. Obtener Casos Penales

- **URL:** /api/get/penales
- **Método:** GET
- **Descripción:** Retorna una lista de casos penales.

The screenshot shows a REST client interface with the following details:

- URL:** `http://192.168.1.46:5000/api/get/penales`
- Method:** GET
- Query Params:** A table with two columns: Key and Value.
- Body:** A JSON array of three objects, each representing a case.

Key	Value
Key	Value
Key	Value

```
1 [
2   {
3     "abogado": "Marroquin",
4     "caso": "Penal",
5     "cliente": "Walter",
6     "comentario": "Penal",
7     "fecha_subida": "23-08-2024",
8     "id": "0900d917-61d6-4ca1-a87a-ac124c8330d8"
9   },
10  {
11    "abogado": "Cartagena",
12    "caso": "penal",
13    "cliente": "Pablo",
14    "comentario": "Mensaje",
15    "id": "f6eaf54-00ce-4894-bff6-d067998ee5d6"
16  },
17  {
18    "abogado": "Marroquin",
19    "caso": "penal",
20    "cliente": "Denys Eduardo ",
```

- **Respuesta:** Un JSON con los datos de los casos penales.

## 10. Obtener Casos de Inmuebles

- **URL:** /api/get/inmueble
- **Método:** GET
- **Descripción:** Retorna una lista de casos de inmuebles.

HTTP <http://192.168.1.46:5000/api/get/usuario>

GET <http://192.168.1.46:5000/api/get/inmueble>

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

Query Params

	Key
	Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "abogado": "Cartagena",
4     "caso": "Inmuebles",
5     "cliente": "Walter",
6     "comentario": "Hola nuevo mundo",
7     "id": "73a6e9e5-cea4-42dc-a3e5-9d2ca81142fc"
8   },
9   {
10    "abogado": "Cartagena",
11    "caso": "Inmuebles",
12    "cliente": "Denys Eduardo ",
13    "comentario": "prueba de fecha",
14    "fecha_subida": "2024-08-23 01:27:40",
15    "id": "50e8ef4b-eae5-4da6-b845-28832adba581"
16  },
17  {
18    "abogado": "Cartagena",
19    "caso": "Inmuebles",
20    "cliente": "Denys Eduardo ",
```

- **Respuesta:** Un JSON con los datos de los casos de inmuebles.

## 11. Obtener Casos de Vehículos

- **URL:** /api/get/vehiculos
- **Método:** GET
- **Descripción:** Retorna una lista de casos de vehículos asociados a un usuario específico.

The screenshot shows a REST client interface with the following details:

- URL:** `http://192.168.1.46:5000/api/get/vehiculos`
- Method:** GET
- Params:** Authorization, Headers (8), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with two rows, each with a 'Key' column.
- Body:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh button.
- Response:** A JSON array of three objects, each representing a vehicle case.

```
1 [
2   {
3     "abogado": "Marroquin",
4     "caso": "Traspasos",
5     "cliente": "Rafael",
6     "comentario": "Prueba de fecha 2",
7     "fecha_subida": "2024-08-23",
8     "id": "543ef636-4257-42f4-82d7-ef2bf133f5f5"
9   },
10  {
11    "abogado": "Marroquin",
12    "caso": "Traspasos",
13    "cliente": "Pablo",
14    "comentario": "Nueva prueba",
15    "id": "d8697024-3a73-4647-9f88-a56b8163dad3"
16  },
17  {
18    "abogado": "Cartagena",
19    "caso": "Traspasos",
20    "cliente": "Denys Eduardo ",
```

- **Respuesta:** Un JSON con los datos de los casos de vehículos.

## 12. Obtener Logs

- **URL:** /api/get/logs
- **Método:** GET
- **Descripción:** Retorna una lista de todos los logs almacenados en la base de datos.

The screenshot displays a REST client interface with a tab for the GET request to `http://192.168.1.46:5000/api/get/logs`. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Query Params section is empty. The Body tab is selected, showing the response in JSON format. The response is a list of three log entries, each containing fields for Fecha, Hora, Mensaje, Servicio, and id.

```
1 [{"Fecha": "2024-08-21",
2   "Hora": "23:39",
3   "Mensaje": "Usuario con IP: 192.168.0.14 se logueo",
4   "Servicio": "info",
5   "id": "0e649c70-4eaf-4e9b-ab5c-3f666c1c619d"}],
6 [{"Fecha": "2024-08-22",
7   "Hora": "23:51",
8   "Mensaje": "Usuario con IP: 192.168.0.14 se logueo",
9   "Servicio": "info",
10  "id": "26ac6d9b-cb35-4d14-adb6-59cc9074dcbf"}],
11 [{"Fecha": "2024-08-22",
12   "Hora": "01:23",
13   "Mensaje": "Usuario con IP: 192.168.1.36 se logueo",
14   "Servicio": "info",
```

- **Respuesta:** Un JSON con los datos de todos los logs.

### 13. Realizar búsqueda

- **URL:** /api/get/<texto>
- **Método:** GET
- **Descripción:** Retorna una lista de todas las rutas donde se encontró el texto.

[CONFLICT] GET http://192.168.1.46:5000/ POST http://192.168.1.46:5000/ GET http://192.168.1.46:5000/ POST http://192.168.1.46:5000/

HTTP http://192.168.1.46:5000/api/get/usuario

GET http://192.168.1.46:5000/api/get/ezequiel

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

Query Params

Key
Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   "/uploads/Modificaciones.pdf:1:EZEQUIEL",
3   "/PDF/Documento Segunda Defensa.pdf:1:EZEQUIEL",
4   "/PDF/Modificaciones-V1.pdf:1:EZEQUIEL",
5   "/PDF/Tesis-APA.pdf:1:EZEQUIEL",
6   "/PDF/Modificaciones.pdf:1:EZEQUIEL",
7   "/PDF/Modificaciones-V2.pdf:1:EZEQUIEL",
8   "/PDF/Tesis Defensa #2.pdf:1:EZEQUIEL",
9   "/casos/Ezequiel/penal/Modificaciones-V1.pdf:1:EZEQUIEL"
10 ]
```

## **Ejecución de la API**

Para ejecutar la API, asegurar que todos los paquetes necesarios estén instalados y que el archivo `api.py` esté configurado correctamente. Luego, se debe iniciar el servidor utilizando el siguiente comando:

```
flask --app api.py run --host=0.0.0.0
```

## **Notas Finales**

- La API utiliza RethinkDB para la gestión de la base de datos.
- Los logs de todas las operaciones realizadas se almacenan en la base de datos en la colección logs.
- Asegurar que las rutas de los directorios para los casos legales estén configuradas correctamente y que los permisos de escritura sean los adecuados para evitar errores al crear directorios.