

# PRUEBAS ADMINISTRADOR DE USUARIOS

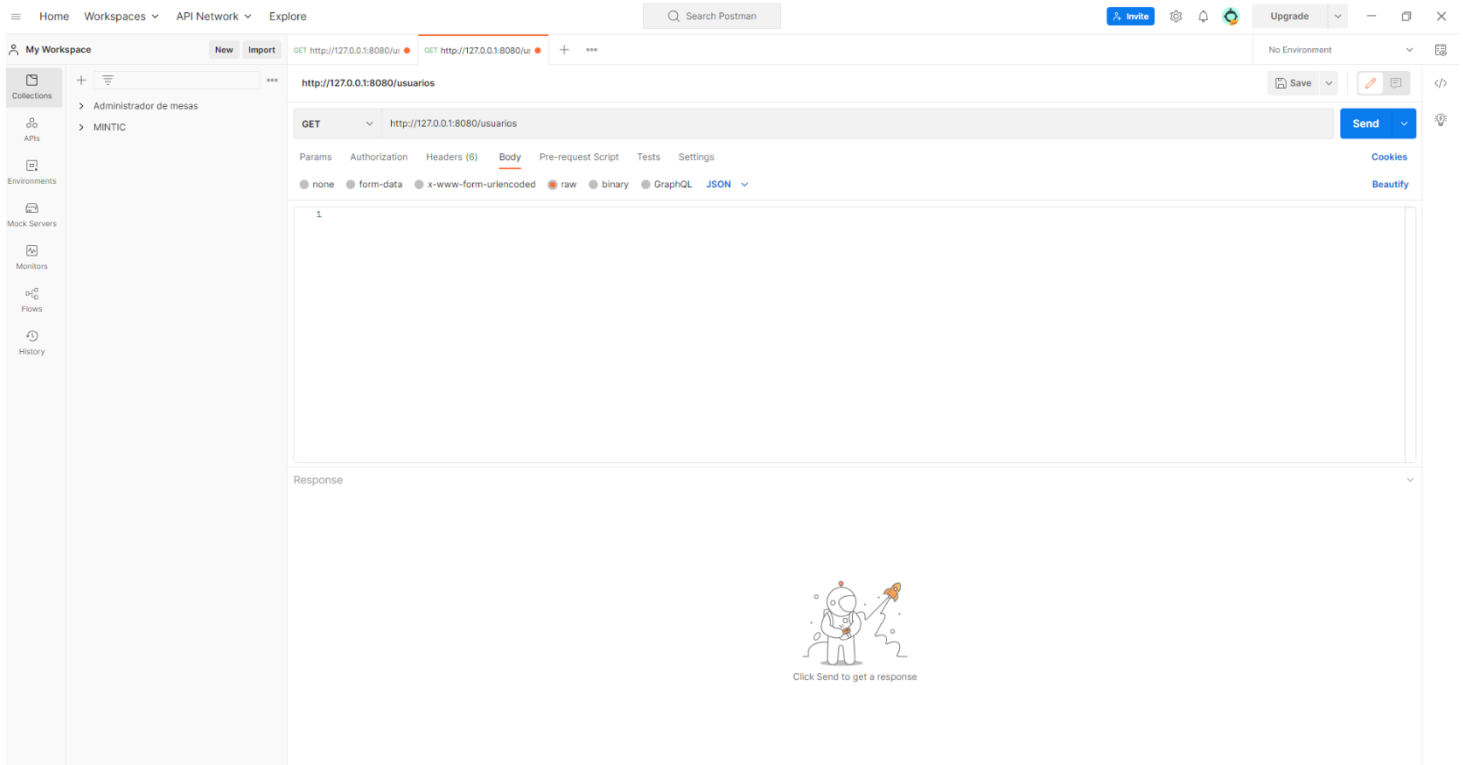
## Método Get

Para obtener el listado de los usuarios creados en el sistema se ingresa la url seguido del “/usuarios”

GET

▼

http://127.0.0.1:8080/usuarios



Se selecciona el botón send y en la parte inferior arroja el listado de usuarios que ya se encuentran en el sistema, como se ve a continuación:

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8080/usuarios`. The response is a JSON array of 4 user objects. The response status is 200 OK, with a time of 121 ms and a size of 1.07 KB.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

```
{
  "_id": "6329109bd0f50b3080756778",
  "seudonimo": "johan10",
  "correo": "joahn95@gmail.com",
  "contrasena": "a665a45920422f9d417e4867efdc4fb8a04a1f3ffff1fa07e998e86f7f7a27ae3"
},
{
  "_id": "63292bc322d2754df490dc98",
  "seudonimo": "prueba1",
  "correo": "prueba1@outlook.com",
  "contrasena": "b0a13c2fd25d4502a5b088a837b42f93d099e8d71659ec6b448210bdaa0e3890"
},
{
  "_id": "63292bd0d22d2754df490dc99",
  "seudonimo": "prueba2",
  "correo": "prueba2@outlook.com",
  "contrasena": "5d877c606120b6f21fed0fcb7fda5aa568f2f36f99291ab2212b7e1006c1301f"
},
{
  "_id": "63292be422d2754df490dc9a",
  "seudonimo": "prueba3",
  "correo": "prueba3@outlook.com",
  "contrasena": "81b281e87063ecf6bdf81b56cc6dfc3388df2ee82cd6363a37070f2a99d245"
},
{
  "_id": "63292bfff22d2754df490dc9b",
  "seudonimo": "prueba4"
}
```

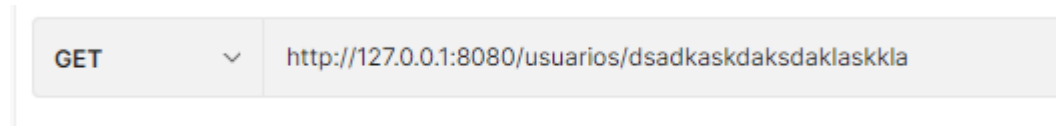
Estos son los mismos registros que se encuentran creados en el servidor de MongoDB

The screenshot shows the ClusterO MongoDB interface. The left sidebar shows the database structure: `Project0` > `usuario`. The main area displays the `Project0.usuario` collection with 5 documents. The documents are shown in a table format with columns for `correo`, `contrasena`, and `_class`.

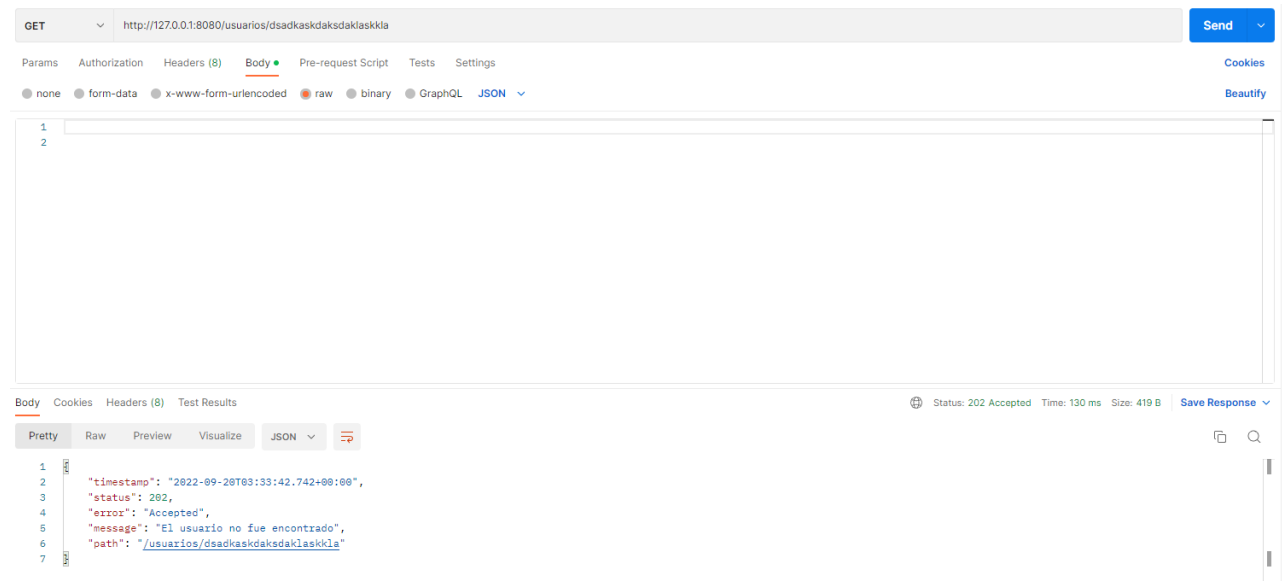
correo	contrasena	_class
joahn95@gmail.com	a665a45920422f9d417e4867efdc4fb8a04a1f3ffff1fa07e998e86f7f7a27ae3	Grupo2.Registraduria.seguridad.Modelos.Usuario
prueba1	b0a13c2fd25d4502a5b088a837b42f93d099e8d71659ec6b448210bdaa0e3890	Grupo2.Registraduria.seguridad.Modelos.Usuario
prueba2	5d877c606120b6f21fed0fcb7fda5aa568f2f36f99291ab2212b7e1006c1301f	Grupo2.Registraduria.seguridad.Modelos.Usuario
prueba3	81b281e87063ecf6bdf81b56cc6dfc3388df2ee82cd6363a37070f2a99d245	Grupo2.Registraduria.seguridad.Modelos.Usuario
prueba4		

## Método Get por ID

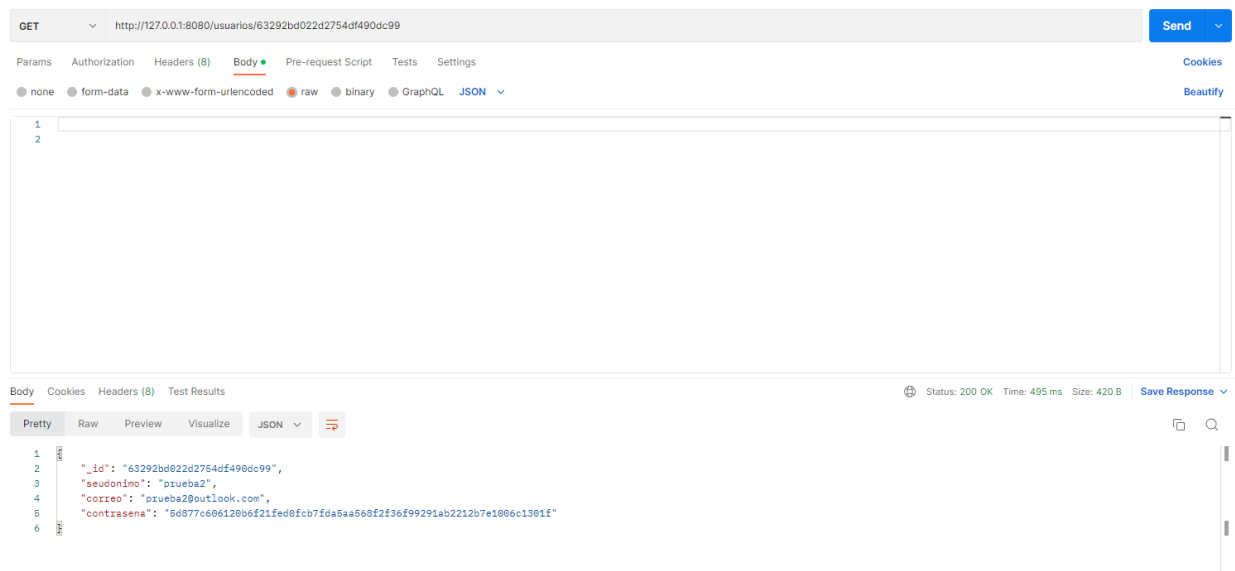
Para este método se debe agregar “/{id}”, después del “/usuarios”, como se ve a continuación



Si se ingresa un id que no esté registrado el sistema mostrará el siguiente mensaje



Por el contrario, si el id existe mostrará la información del usuario



## Método Post

Se selecciona el método post en el puntero como se muestra a continuación:

POST	▼	http://127.0.0.1:8080/usuarios
------	---	--------------------------------

Y en el cuerpo se ingresa un Json como este

```
{  
  "seudonimo": "prueba6",  
  "correo": "prueba6@outlook.com",  
  "contrasena": "axcerkfo1575"  
}
```

http://127.0.0.1:8080/usuarios

POST	▼	http://127.0.0.1:8080/usuarios
------	---	--------------------------------

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

```
1 {  
2   ...."seudonimo": "prueba6",  
3   ...."correo": "prueba6@outlook.com",  
4   ...."contrasena": "axcerkfo1575"  
5 }  
6 |
```

Al seleccionar el botón send debe mostrar como resultado los datos del usuario creado

The screenshot shows a REST client interface. At the top, a POST request is configured to `http://127.0.0.1:8080/usuarios`. The 'Body' tab is selected, showing a JSON payload: 

```
{  "seudonimo": "prueba6",  "correo": "prueba6@outlook.com",  "contrasena": "axcerkfo1575"}
```

. Below the request, the 'Test Results' section shows the response in 'Pretty' format: 

```
{  "_id": "63292fa322d2754df490dc9c",  "seudonimo": "prueba6",  "correo": "prueba6@outlook.com",  "contrasena": "c7bb492e4112e4d5beae2df5891afafa7b8a2a7f7880a0d761055fbd5b062395"}
```

. The status bar indicates a 201 Created status, 1174 ms time, and 425 B size.

Y se debe haber registrado en el servidor de mongoDB

The screenshot shows the MongoDB Compass interface. On the left, the 'Project0' database is selected, and the 'usuario' collection is highlighted. The main panel displays the 'usuario' collection with three documents. The third document is highlighted in yellow, showing the same data as the REST client response: 

```
{  "_id": "63292fa322d2754df490dc9c",  "seudonimo": "prueba6",  "correo": "prueba6@outlook.com",  "contrasena": "c7bb492e4112e4d5beae2df5891afafa7b8a2a7f7880a0d761055fbd5b062395",  "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario"}
```

. The top of the interface shows database statistics: STORAGE SIZE: 36KB, LOGICAL DATA SIZE: 1.27KB, TOTAL DOCUMENTS: 6, INDEXES TOTAL SIZE: 36KB.

El campo seudónimo, correo y contraseña son obligatorios, si se envían vacíos o no se ingresan en el cuerpo del Json se mostrará alguno de los siguientes mensajes:

POST http://127.0.0.1:8080/usuarios

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

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

```
1 {
2   "correo": "usuario5@hotmail.com",
3   "contrasena": "ndjdghf5454fwk"
4 }
5
```

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 188 ms Size: 377 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-09-22T03:06:56.668+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "El campo seudonimo es requerido.",
6   "path": "/usuarios"
7 }
```

Administración de usuarios / [Crar usuario](#)

Save ...

POST http://127.0.0.1:8080/usuarios

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

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

```
1 {
2   "seudonimo": "",
3   "correo": "usuario9@hotmail.com",
4   "contrasena": "ndjdghf5454fwk"
5 }
```

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 208 ms Size: 377 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-09-22T03:22:09.005+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "El campo seudonimo es requerido.",
6   "path": "/usuarios"
7 }
```

POST http://127.0.0.1:8080/usuarios Send

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

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

```
1 [JSON]
2 {
3   "seudonimo": "usuario8",
4   "contrasena": "ndjdghf6464fwk"
5 }
```

Body Cookies Headers (7) Test Results Status: 400 Bad Request Time: 12 ms Size: 374 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [JSON]
2 {
3   "timestamp": "2022-09-22T03:09:14.564+00:00",
4   "status": 400,
5   "error": "Bad Request",
6   "message": "El campo correo es requerido.",
7   "path": "/usuarios"
8 }
```

POST http://127.0.0.1:8080/usuarios Send

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

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

```
1 [JSON]
2 {
3   "seudonimo": "usuario9",
4   "correo": "",
5   "contrasena": "ndjdghf6464fwk"
6 }
```

Body Cookies Headers (7) Test Results Status: 400 Bad Request Time: 15 ms Size: 374 B Save Response

Pretty Raw Preview Visualize JSON

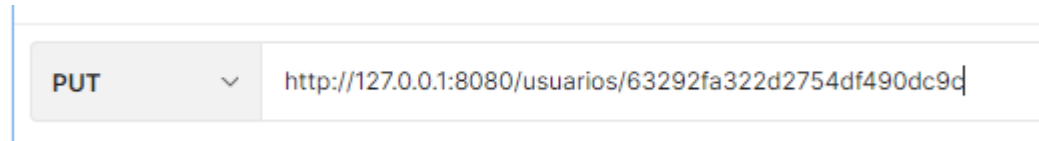
```
1 [JSON]
2 {
3   "timestamp": "2022-09-22T03:24:53.252+00:00",
4   "status": 400,
5   "error": "Bad Request",
6   "message": "El campo correo es requerido.",
7   "path": "/usuarios"
8 }
```





## Método Put

Para el método put se debe agregar “/{id}”, después del “/usuarios”, como se ve a continuación



A screenshot of a REST client interface. It shows a PUT request to the URL `http://127.0.0.1:8080/usuarios/63292fa322d2754df490dc9c`. The method 'PUT' is selected in a dropdown menu.

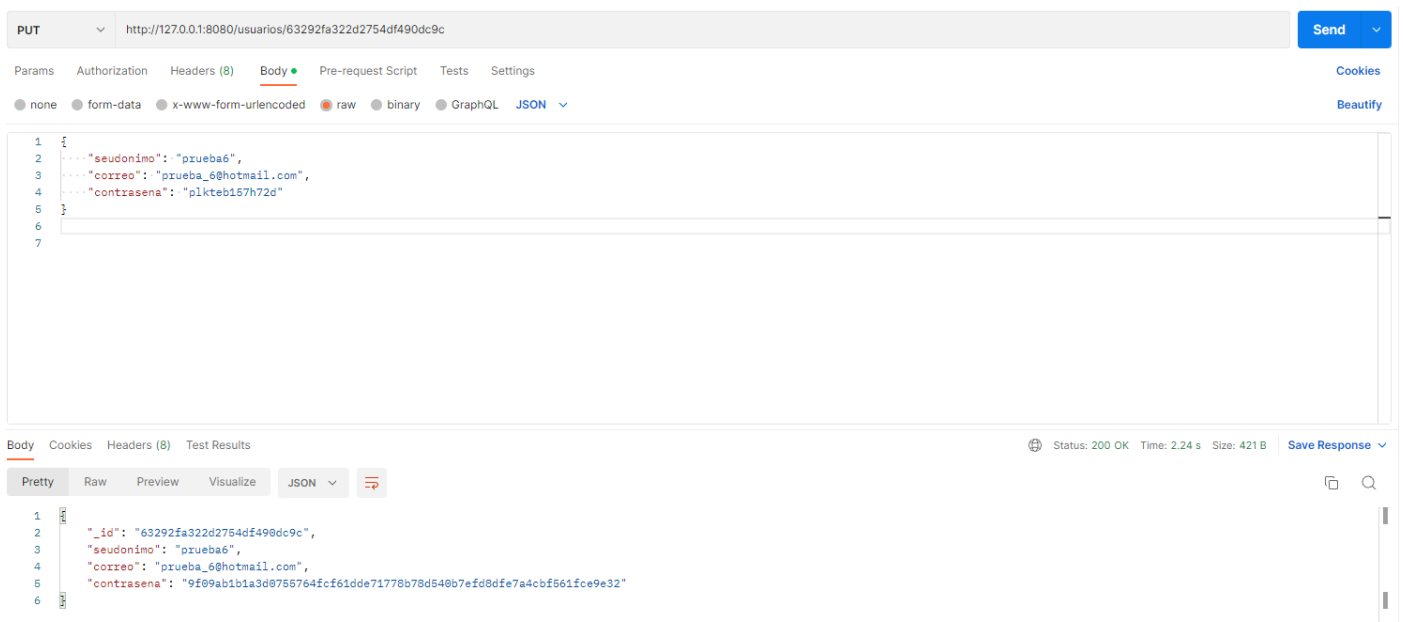
Y en el cuerpo del json se editan los campos que se quieren actualizar

```
{
  "seudonimo": "prueba6",
  "correo": "prueba_6@hotmail.com",
  "contrasena": "plkteb157h72d"
}
```



A screenshot of a REST client interface showing a PUT request. The URL is `http://127.0.0.1:8080/usuarios/63292fa322d2754df490dc9c`. The 'Body' tab is selected, and the request body is a JSON object: `{ "seudonimo": "prueba6", "correo": "prueba_6@hotmail.com", "contrasena": "plkteb157h72d" }`. The 'JSON' format is selected in the dropdown menu.

Y se selecciona el botón send para actualizar



A screenshot of a REST client interface showing a PUT request. The URL is `http://127.0.0.1:8080/usuarios/63292fa322d2754df490dc9c`. The 'Body' tab is selected, and the request body is a JSON object: `{ "seudonimo": "prueba6", "correo": "prueba_6@hotmail.com", "contrasena": "plkteb157h72d" }`. The 'JSON' format is selected in the dropdown menu. The 'Send' button is visible in the top right corner. Below the request body, the response is shown in the 'Body' tab, displaying a JSON object with the updated user information: `{ "_id": "63292fa322d2754df490dc9c", "seudonimo": "prueba6", "correo": "prueba_6@hotmail.com", "contrasena": "9f09ab1b1a3d0755764fcf61dde71778b78d540b7efd8df7a4cbf561fce9e32" }`. The status is 200 OK, and the response is saved.

Y se debe actualizar en el servidor de mongoDB

Project0.usuario

STORAGE SIZE: 34KB LOGICAL DATA SIZE: 1,27KB TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

```
{
  "_id": ObjectId("63292be422d2754df490dc9a"),
  "seudonimo": "prueba3",
  "correo": "prueba3@outlook.com",
  "contrasena": "81b281e87063ecf5bdf81b56cc6dfc3388fd2ee82cd6363a37070f2a99d245",
  "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario"
}
```

```
{
  "_id": ObjectId("63292bff22d2754df490dc9b"),
  "seudonimo": "prueba4",
  "correo": "prueba4@outlook.com",
  "contrasena": "037624b23f11d59fc4a585b453a6c4d8302baa9348bd5efda8b5fb2e0432c812",
  "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario"
}
```

```
{
  "_id": ObjectId("63292fa322d2754df490dc9c"),
  "seudonimo": "prueba6",
  "correo": "prueba6@hotmail.com",
  "contrasena": "9f09ab1b1a3d0755764fcf61dde71770b70d54eb7efd8dfe7a4cbf561fce9e32",
  "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario"
}
```

Al igual que con el método get por id, si se ingresa un id que no existe, no se podrá actualizar la información del usuario

PUT http://127.0.0.1:8080/usuarios/63292kjkfjdsjhfggh Send

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

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

```
1 {
2   "seudonimo": "prueba5",
3   "correo": "prueba5@outlook.com",
4   "contrasena": "fsdIsdIsdg4548"
5 }
6
```

Body Cookies Headers (8) Test Results Status: 404 Not Found Time: 135 ms Size: 433 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-09-20T03:37:17.755+00:00",
3   "status": 404,
4   "error": "Not Found",
5   "message": "No se pudo actualizar usuario, Id no existe",
6   "path": "/usuarios/63292kjkfjdsjhfggh"
7 }
```

Si al actualizar un usuario se ingresa un seudónimo o correo que ya existe mostrará alguno de los siguientes mensajes:

Administración de usuarios / Actualizar usuario

PUT http://127.0.0.1:8080/usuarios/63292bc322d2754df490dc98

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

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

```
1 {
2   "seudonimo": "usuario6",
3   "correo": "usuario6@hotmail.com",
4   "contrasena": "sdd1f5b56d6dbd"
5 }
```

Body Cookies Headers (8) Test Results

Status: 406 Not Acceptable Time: 148 ms Size: 448 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-09-22T03:12:52.597+00:00",
3   "status": 406,
4   "error": "Not Acceptable",
5   "message": "Este seudonimo ya existe, intenta con otro.",
6   "path": "/usuarios/63292bc322d2754df490dc98"
7 }
```

PUT http://127.0.0.1:8080/usuarios/63292bc322d2754df490dc98

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

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

```
1 {
2   "seudonimo": "usuario8",
3   "correo": "usuario6@hotmail.com",
4   "contrasena": "sdd1f5b56d6dbd"
5 }
```

Body Cookies Headers (8) Test Results

Status: 406 Not Acceptable Time: 1243 ms Size: 445 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-09-22T03:14:15.067+00:00",
3   "status": 406,
4   "error": "Not Acceptable",
5   "message": "Este correo ya existe, intenta con otro.",
6   "path": "/usuarios/63292bc322d2754df490dc98"
7 }
```

Al actualizar un campo este no puede ir vacío, de lo contrario arrojará los siguientes mensajes:

PUT http://127.0.0.1:8080/usuarios/632bcff1f10a0e6700e73621f

Send

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

Cookies

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

Beautify

```
1 [RAW]
2 {
3   "seudonimo": "",
4   "correo": "usuario6@hotmail.com",
5   "contrasena": "sdd1f6b56d6dbd"
6 }
```

Body Cookies Headers (7) Test Results

 Status: 400 Bad Request Time: 12 ms Size: 402 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [RAW]
2 {
3   "timestamp": "2022-09-22T03:29:55.201+00:00",
4   "status": 400,
5   "error": "Bad Request",
6   "message": "El camposeudonimo es requerido.",
7   "path": "/usuarios/632bcff1f10a0e6700e73621f"
8 }
```

PUT http://127.0.0.1:8080/usuarios/632bcff1f10a0e6700e73621f

Send

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


Cookies

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

Beautify

```
1 [RAW]
2 {
3   "seudonimo": "usuario10",
4   "correo": "",
5   "contrasena": "sdd1f6b56d6dbd"
6 }
```

Body Cookies Headers (7) Test Results

 Status: 400 Bad Request Time: 7 ms Size: 399 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [RAW]
2 {
3   "timestamp": "2022-09-22T03:31:36.044+00:00",
4   "status": 400,
5   "error": "Bad Request",
6   "message": "El campo correo es requerido.",
7   "path": "/usuarios/632bcff1f10a0e6700e73621f"
8 }
```

PUT http://127.0.0.1:8080/usuarios/632bcff1f10a0e6700e73621f

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

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

```
1 {
2   "seudonimo": "usuario10",
3   "correo": "usuario10@hotmail.com",
4   "contrasena": ""
5 }
```

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 13 ms Size: 404 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-09-22T03:32:19.578+00:00",
3   "status": 400,
4   "error": "Bad Request",
5   "message": "El campo contraseña es requerido.",
6   "path": "/usuarios/632bcff1f10a0e6700e73621f"
7 }
```

## Método delete

Para el método delete se debe agregar “/{id}”, después del “/usuarios”, como se ve a continuación

DELETE http://127.0.0.1:8080/usuarios/63292fa322d2754df490dc9c

No hay que ingresar un json como cuerpo para poder eliminar un registro

DELETE http://127.0.0.1:8080/usuarios/63292fa322d2754df490dc9c

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

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

```
1
2
```

Cookies Beautify

## Finalmente se selecciona el botón send para eliminar el usuario por ID

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

- URL:** `http://127.0.0.1:8080/usuarios/63292bc322d2754df490dc98`
- Method:** DELETE
- Body:** 1
- Response:** Status: 202 Accepted, Time: 245 ms, Size: 419 B. The response body is a JSON object: 

```
{  "timestamp": "2022-09-22T03:15:16.512+00:00",  "status": 202,  "error": "Accepted",  "message": "El usuario fue eliminado.",  "path": "/usuarios/63292bc322d2754df490dc98"}
```

## De igual forma se tuvo que eliminar el registro desde mongoDB

The screenshot shows the MongoDB Compass interface with the following details:

- Database:** Project0
- Collection:** usuario
- Documents:** 3 documents are listed, each with fields: `_id`, `seudonimo`, `correo`, `contrasena`, and `_class`.

The documents shown are:

- `{ "_id": ObjectId("63292bd022d2754df490dc99"), "seudonimo": "prueba2", "correo": "prueba2@outlook.com", "contrasena": "5d877c68612086f21fed0fcb7fda5aa568f2f36f99291ab2212b7e1006c1301f", "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario" }`
- `{ "_id": ObjectId("63292be422d2754df490dc9a"), "seudonimo": "prueba3", "correo": "prueba3@outlook.com", "contrasena": "81b281e87863ecf5bdf81b56cc6dfc3388fd2ee82cd6363a37070f2a99d245", "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario" }`
- `{ "_id": ObjectId("63292bff22d2754df490dc9b"), "seudonimo": "prueba4", "correo": "prueba4@outlook.com", "contrasena": "937624b23f11d59fca585b45306c4d8302baa9348bd5efda8b5fb2e0432c812", "_class": "Grupo2.Registraduria.seguridad.Modelos.Usuario" }`

Al ingresar un id que no esté registrado el sistema mostrará el siguiente mensaje

The screenshot displays a REST client interface with a DELETE request to `http://127.0.0.1:8080/usuarios/dsadmksdkdskdaskklia`. The response is a 404 Not Found status with a JSON body indicating the user does not exist.

**Request Details:**

- Method: DELETE
- URL: `http://127.0.0.1:8080/usuarios/dsadmksdkdskdaskklia`
- Body: Empty

**Response Details:**

- Status: 404 Not Found
- Time: 156 ms
- Size: 430 B

**Response Body (JSON):**

```
{
  "timestamp": "2022-09-20T03:38:53.762+00:00",
  "status": 404,
  "error": "Not Found",
  "message": "El Id del usuario ingresado no existe",
  "path": "/usuarios/dsadmksdkdskdaskklia"
}
```