

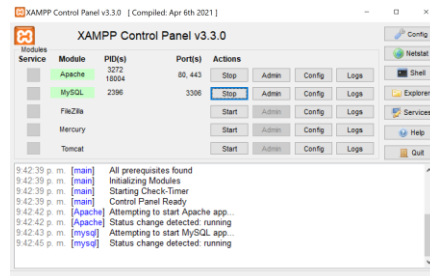
Nombre Estudiante: Herlendy Alejandro Sánchez Gaitán

Cédula: 100104499

Correo: Herlendy.sanchez11@gmail.com

PASOS

1. Se inicia el servidor



```
PS C:\Users\alejo\Documents\Universidad2023A\Curso\FullStack\project_backend> php artisan serve

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server

2023-06-13 22:01:12 .....
```

2. Se generan la migración del modelo correspondiente a la tabla de usuarios

```
alejo@LAPTOP-K4G6185F MINGW64 ~/Documents/Universidad2023A/Curso/FullStack/project_backend (master)
$ php artisan migrate

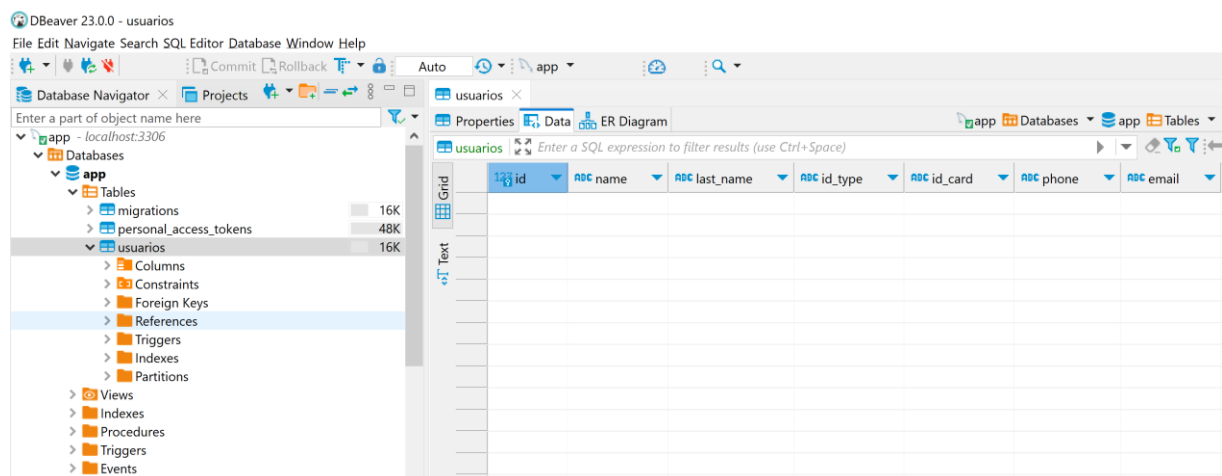
INFO Preparing database.

Creating migration table ..... 19ms DONE

INFO Running migrations.

2019_12_14_000001_create_personal_access_tokens_table ..... 58ms DONE
2023_06_06_015815_create_usuarios_table ..... 16ms DONE
```

3. Se verifica que se haya creado la tabla y por ende no existe ningún usuario



api_backend_project / Read

GET http://localhost:8000/api/usuario/read

Send

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

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (10) Test Results

404 Not Found 689 ms 372 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "No se encontraron usuarios en la base de datos"
3 }
```

4. Mediante postman se creará un nuevo usuario a través del método POST

POST Create

api_backend_project / Create

POST http://localhost:8000/api/usuario/create

Params Authorization Headers (9) Body Pre-request Script

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

```
1 {
2   "name": "Maria",
3   "last_name": "Gonzalez",
4   "id_type": "Tarjeta de identidad",
5   "id_card": "123456789",
6   "phone": "123456789",
7   "email": "maria@gmail.com",
8   "profession": "estudiante",
9   "role": "Espectador",
10  "message": ""
11 }
```

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Registro Exitoso!"
3 }
```

app - localhost:3306

Databases

- app
 - migrations 16K
 - personal_access_tokens 48K
 - usuarios 16K

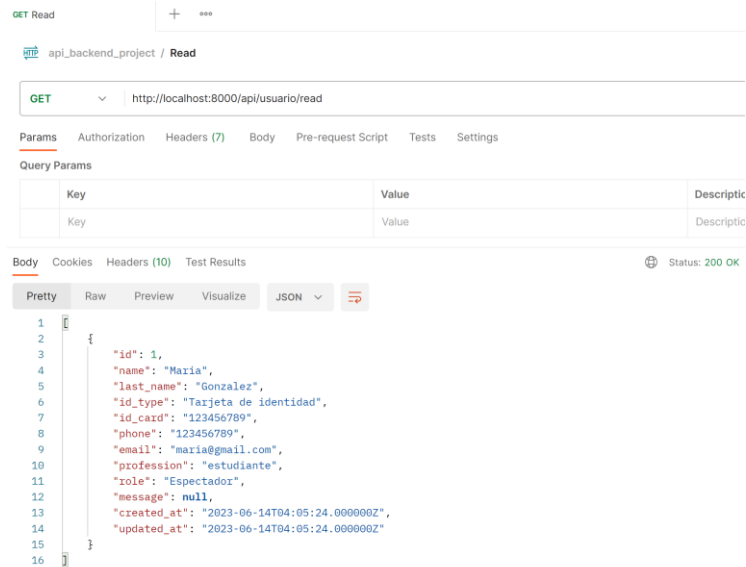
usuarios

id	name	last_name	id_type	id_card	phone	email
1	Maria	Gonzalez	Tarjeta de identidad	123456789	123456789	maria@gmail.com

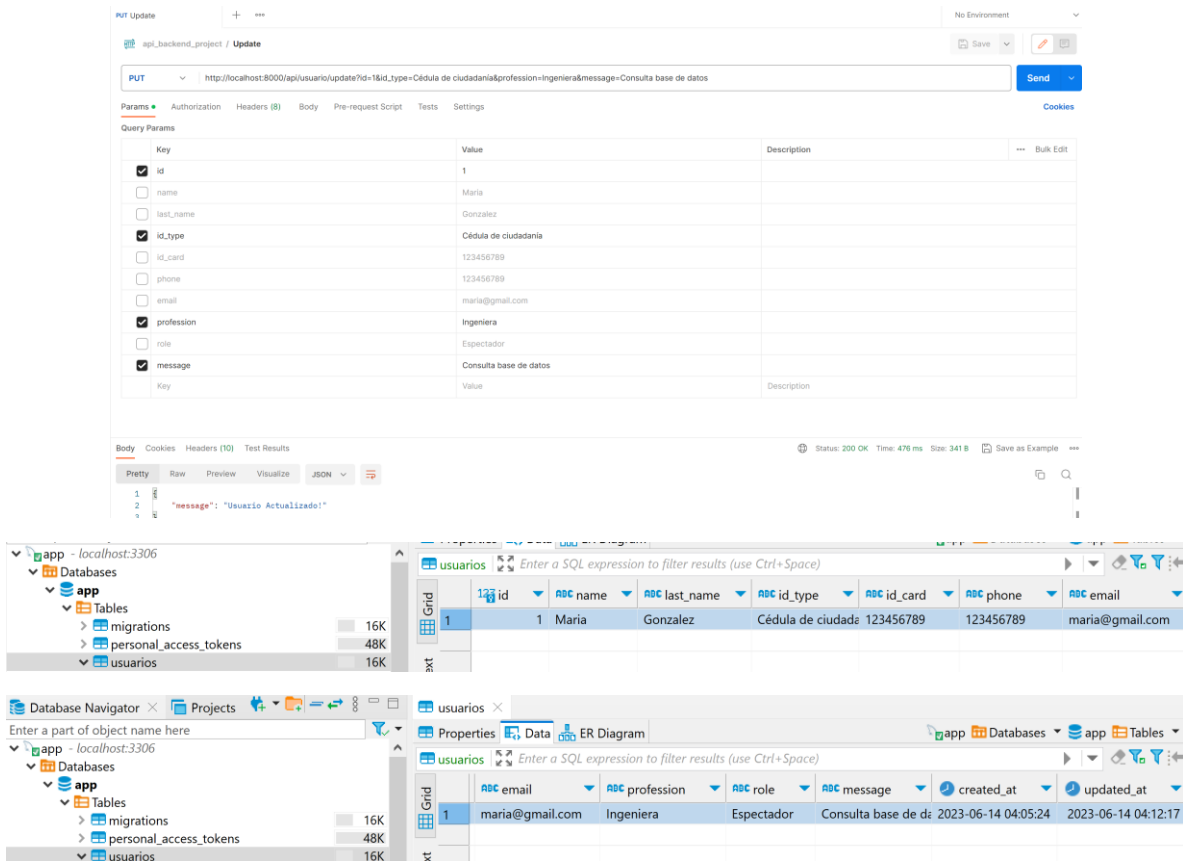
usuarios

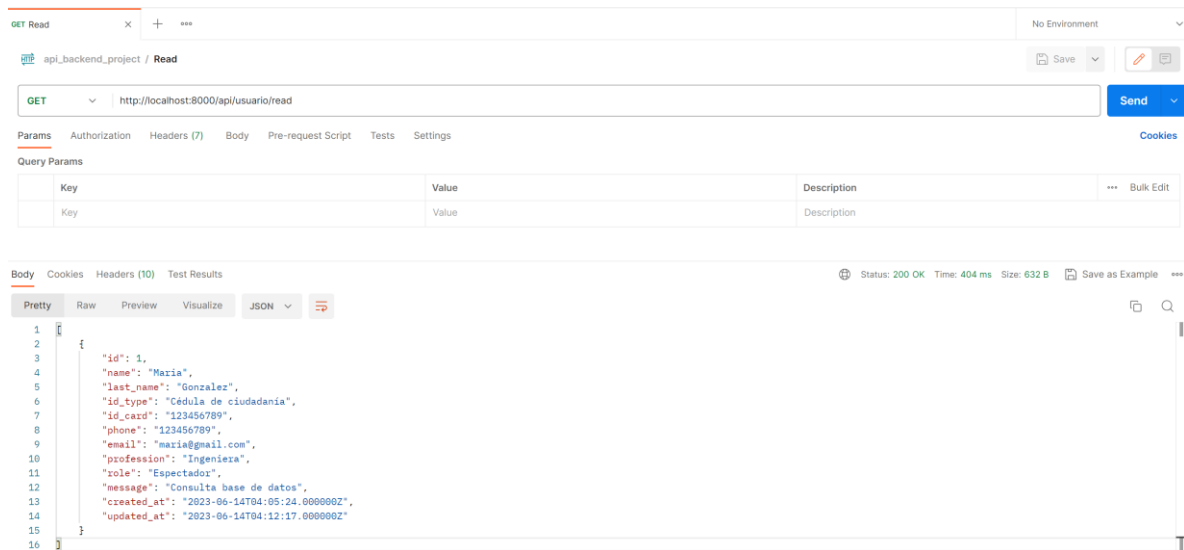
email	profession	role	message	created_at	updated_at
maria@gmail.com	estudiante	Espectador	[NULL]	2023-06-14 04:05:24	2023-06-14 04:05:24

5. Se obtiene la información del total de usuarios por medio de una petición GET

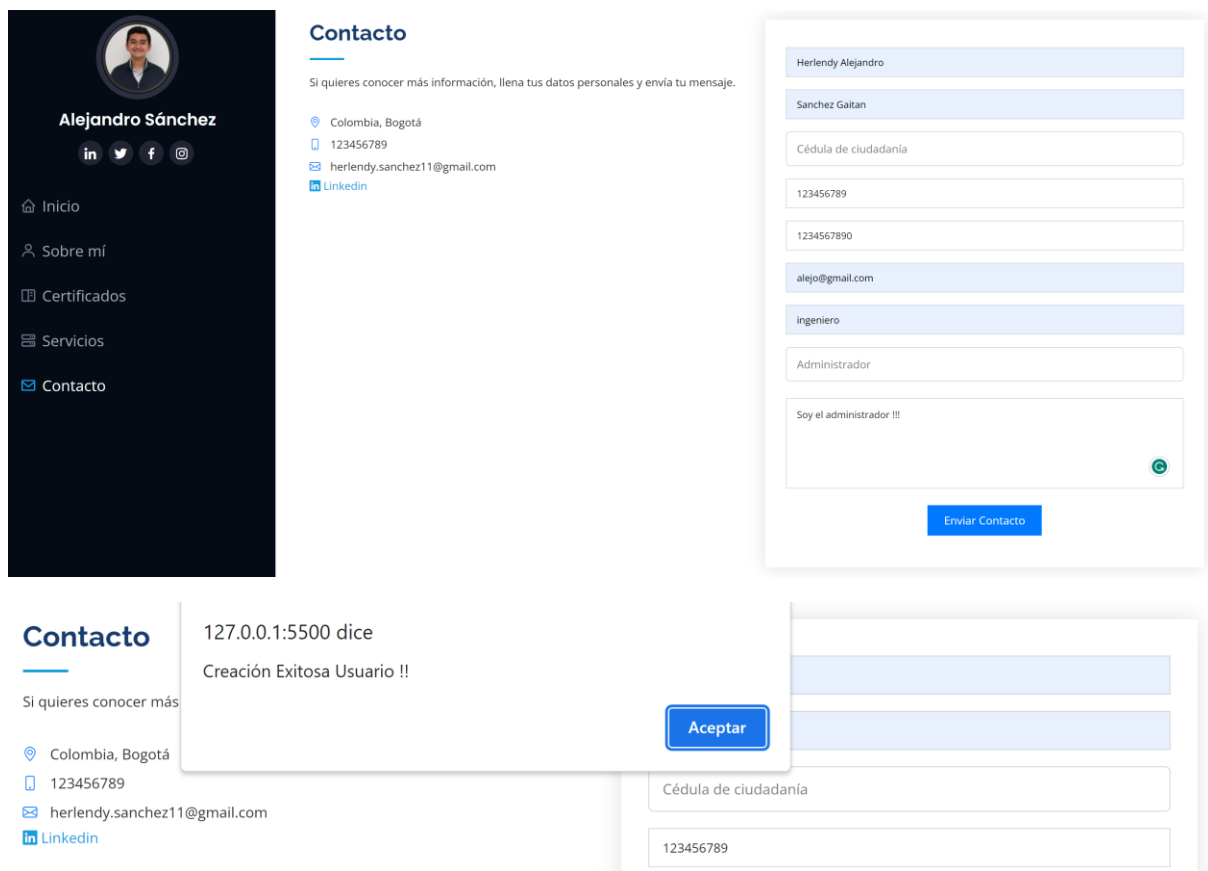


6. Se actualiza el tipo de identificación, el mensaje y la profesión del usuario creado anteriormente.

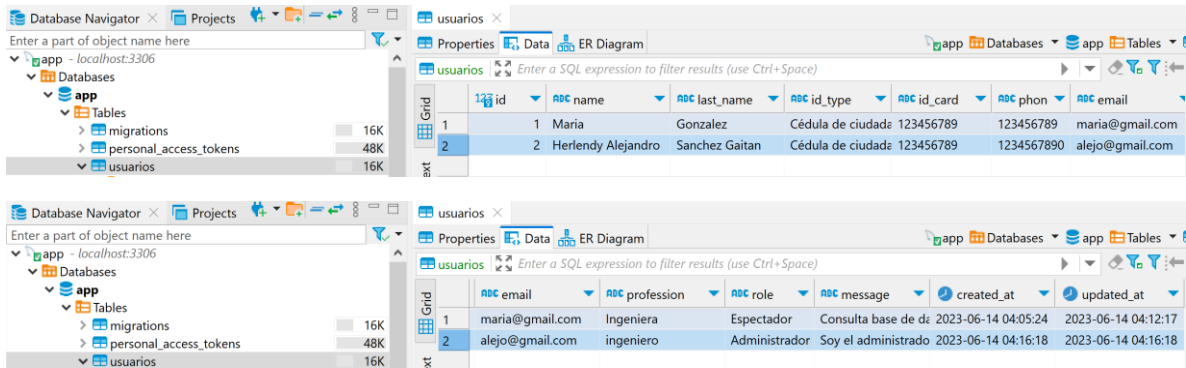




7. Se crea un nuevo usuario, pero esta vez por interfaz

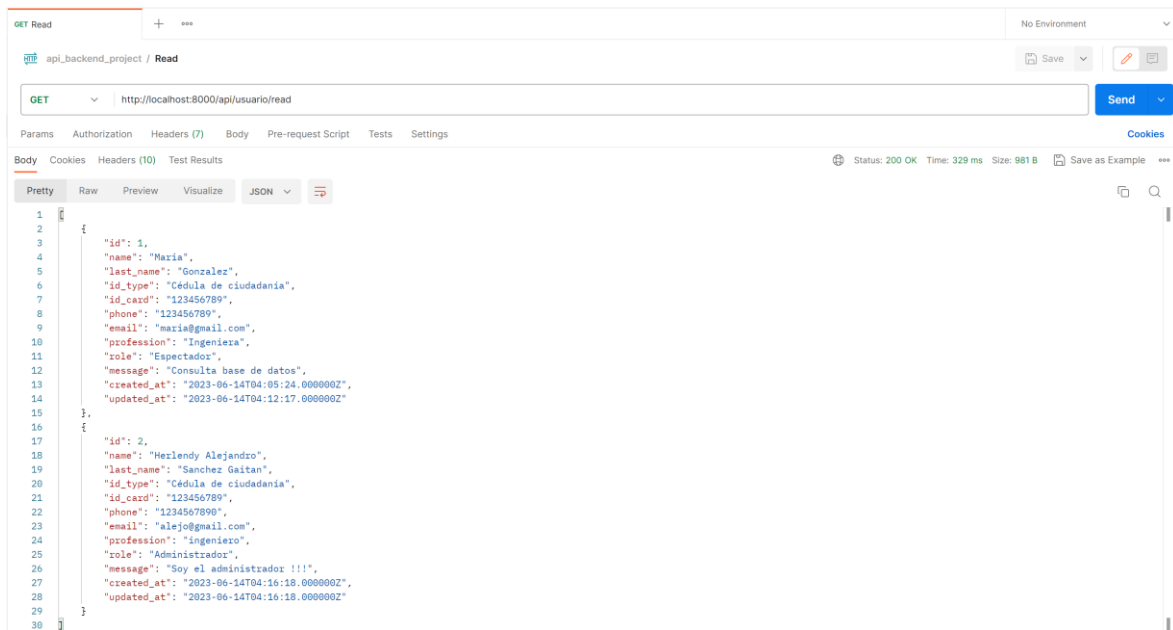


8. Se valida que el usuario aparezca en la base de datos



The screenshot shows the Database Navigator interface with the 'usuarios' table selected. The table structure is displayed in the 'Grid' view, showing columns: id, name, last_name, id_type, id_card, phone, and email. The data is as follows:

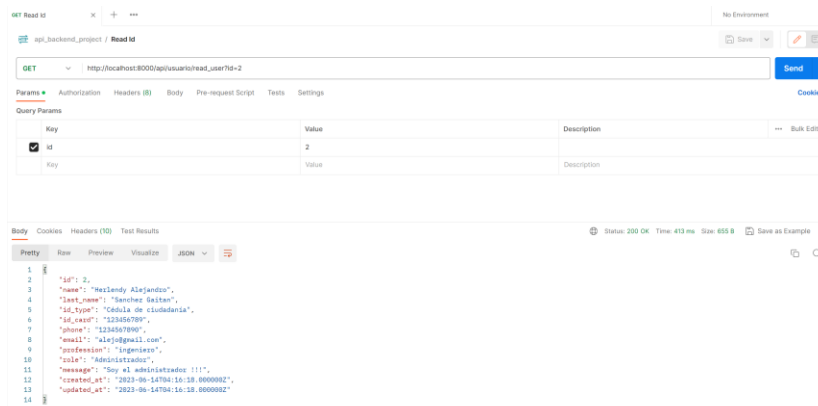
id	name	last_name	id_type	id_card	phone	email
1	Maria	Gonzalez	Cédula de ciudadanía	123456789	123456789	maria@gmail.com
2	Herlenny Alejandro	Sanchez Gaitan	Cédula de ciudadanía	123456789	123456789	alejo@gmail.com



The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/usuario/read`. The response is a JSON array of two user objects, each containing fields: id, name, last_name, id_type, id_card, phone, email, profession, role, message, created_at, and updated_at.

```
1 {
2   "id": 1,
3   "name": "Maria",
4   "last_name": "Gonzalez",
5   "id_type": "Cédula de ciudadanía",
6   "id_card": "123456789",
7   "phone": "123456789",
8   "email": "maria@gmail.com",
9   "profession": "Ingeniera",
10  "role": "Espectador",
11  "message": "Consulta base de datos",
12  "created_at": "2023-06-14T04:05:24.000000Z",
13  "updated_at": "2023-06-14T04:12:17.000000Z"
14 },
15 {
16   "id": 2,
17   "name": "Herlenny Alejandro",
18   "last_name": "Sanchez Gaitan",
19   "id_type": "Cédula de ciudadanía",
20   "id_card": "123456789",
21   "phone": "123456789",
22   "email": "alejo@gmail.com",
23   "profession": "ingeniero",
24   "role": "Administrador",
25   "message": "Soy el administrador !!!",
26   "created_at": "2023-06-14T04:16:18.000000Z",
27   "updated_at": "2023-06-14T04:16:18.000000Z"
28 }
29 ]
30 }
```

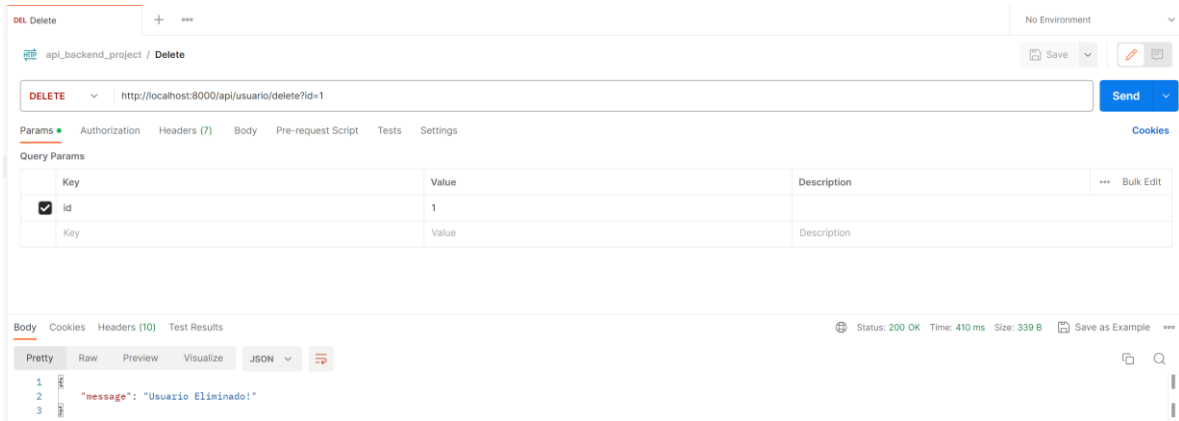
9. Se busca únicamente el usuario con id 2



The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/usuario/read?id=2`. The response is a JSON object representing the user with id 2, containing fields: id, name, last_name, id_type, id_card, phone, email, profession, role, message, created_at, and updated_at.

```
1 {
2   "id": 2,
3   "name": "Herlenny Alejandro",
4   "last_name": "Sanchez Gaitan",
5   "id_type": "Cédula de ciudadanía",
6   "id_card": "123456789",
7   "phone": "123456789",
8   "email": "alejo@gmail.com",
9   "profession": "ingeniero",
10  "role": "Administrador",
11  "message": "Soy el administrador !!!",
12  "created_at": "2023-06-14T04:16:18.000000Z",
13  "updated_at": "2023-06-14T04:16:18.000000Z"
14 }
```

10. Se elimina el primer usuario que se creo es decir cuyo id es el 1



DELETE http://localhost:8000/api/usuario/delete?id=1

Params: Authorization, Headers (7), Body, Pre-request Script, Tests, Settings

Query Params

Key	Value	Description
id	1	

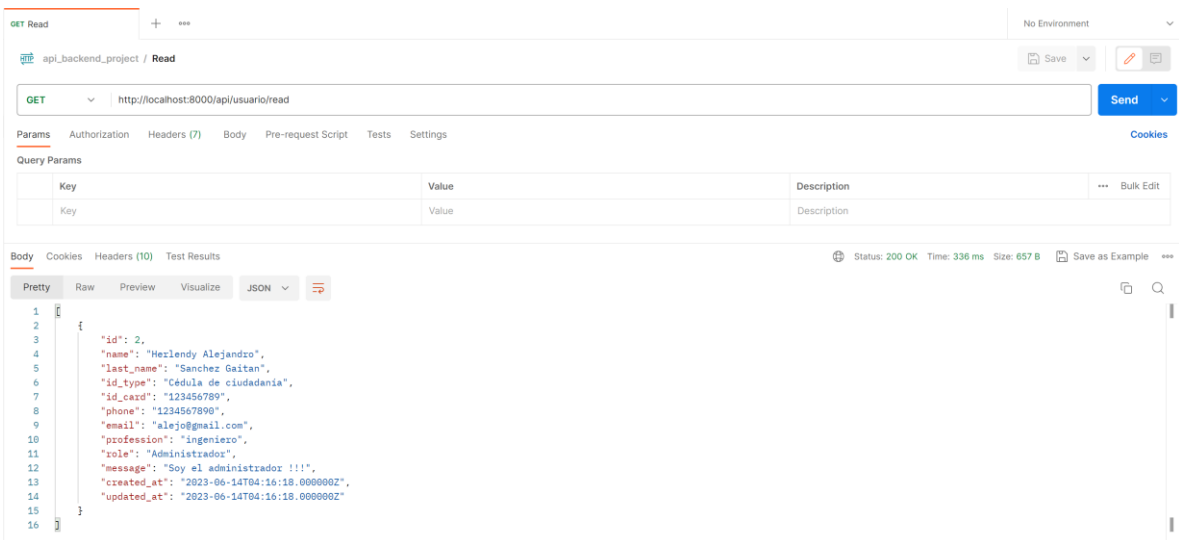
Body: Cookies, Headers (10), Test Results

Status: 200 OK, Time: 410 ms, Size: 339 B

Save as Example

```
1 {
2   "message": "Usuario Eliminado!"
3 }
```

11. Se valida que ya no exista en la base de datos



GET http://localhost:8000/api/usuario/read

Params: Authorization, Headers (7), Body, Pre-request Script, Tests, Settings

Query Params

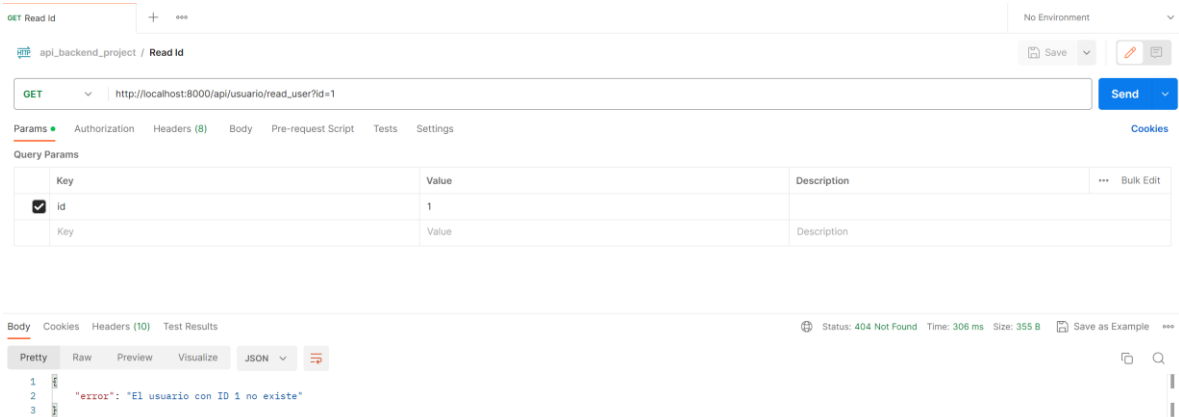
Key	Value	Description
Key	Value	Description

Body: Cookies, Headers (10), Test Results

Status: 200 OK, Time: 336 ms, Size: 657 B

Save as Example

```
1 {
2   "id": 2,
3   "name": "Hecleidy Alejandro",
4   "last_name": "Sanchez Galitan",
5   "id_type": "Cédula de ciudadanía",
6   "id_card": "123456789",
7   "phone": "1234567890",
8   "email": "alej@ej@gmail.com",
9   "profession": "ingeniero",
10  "role": "Administrador",
11  "message": "Soy el administrador !!!",
12  "created_at": "2023-06-14T04:16:18.000000Z",
13  "updated_at": "2023-06-14T04:16:18.000000Z"
14 }
15
16 }
```



GET http://localhost:8000/api/usuario/read_user?id=1

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

Query Params

Key	Value	Description
id	1	

Body: Cookies, Headers (10), Test Results

Status: 404 Not Found, Time: 306 ms, Size: 355 B

Save as Example

```
1 {
2   "error": "El usuario con ID 1 no existe"
3 }
```

Database Navigator | Projects | usuarios | Properties | Data | ER Diagram

Enter a part of object name here

app - localhost:3306

Databases

- app
 - migrations 16K
 - personal_access_tokens 48K
 - usuarios 16K

Enter a SQL expression to filter results (use Ctrl+Space)

id	name	last_name	id_type	id_card	phon	email
1	Herlenny	Alejandro	Cédula de ciudadania	123456789	1234567890	alejo@gmail.com
2						

email	profession	role	message	created_at	updated_at
alejo@gmail.com	ingeniero	Administrador	Soy el administrado	2023-06-14 04:16:18	2023-06-14 04:16:18

A continuación, la documentación de cada uno de los métodos utilizados anteriormente.

api_backend_project

+

...

api_backend_project

Version CURRENT

Language cURL

⚙️

api_backend_project

Add collection description...

POST Create

Open Request →

http://localhost:8000/api/usuario/create

Método "Create" se realiza una petición de tipo POST para la creación de un Usuario, se envían los datos como JSON en el body.

Body raw (json)

json

```
{
  "name": "Alejandro",
  "last_name": "Sanchez",
  "id_type": "Cédula de ciudadania",
  "id_card": "123456789",
  "phone": "123456789",
  "email": "alejo@gmail.com",
  "profession": "Ingeniero",
  "role": "Administrador",
  "message": ""
}
```

JUMP TO

Introduction

POST Create

GET Read

GET Read Id

DEL Delete

PUT Update

api_backend_project

+

...

api_backend_project

Version

CURRENT

Language

cURL

⌵

⚙

GET

Read

Open Request→

http://localhost:8000/api/usuario/read

Método "Read" se realiza una petición de tipo GET, la cual permite leer todos los Usuarios que se encuentran almacenados en la base de datos

GET

Read Id

Open Request→

http://localhost:8000/api/usuario/read_user?id=1

Método "Read Id" se realiza una petición de tipo GET para leer un usuario en particular, se envían el id como parametro en la url.

Query Params

id	1
----	---

DELETE

Delete

Open Request→

http://localhost:8000/api/usuario/delete?id=1

Método "Delete" se realiza una petición de tipo DELETE para eliminar un usuario en particular, se envían el id como parametro en la url.

Query Params

id	1
----	---

JUMP TO

Introduction

POST Create

GET Read

GET Read Id

DEL Delete

PUT Update

api_backend_project

+

...

api_backend_project

Version

CURRENT

Language

cURL

⌵

⚙

PUT

Update

Open Request→

http://localhost:8000/api/usuario/update?id=1&id_type=Cédula de ciudadanía&profession=Ingeniera&message=Consulta base de datos

Método "Update" se realiza una petición de tipo PUT para actualizar un usuario en particular, se envían el id como parametro en la url, y a su vez los campos que se desean modificar del usuario.

Query Params

id	1
name	Maria
last_name	Gonzalez
id_type	Cédula de ciudadanía
id_card	123456789
phone	123456789
email	maria@gmail.com
profession	Ingeniera
role	Espectador
message	Consulta base de datos

JUMP TO

Introduction

POST Create

GET Read

GET Read Id

DEL Delete

PUT Update

Proyecto Frontend: <https://github.com/AlejandroSanchez01/project-frontend>

Proyecto Backend: <https://github.com/AlejandroSanchez01/project-backend>