

CRUD de Clientes en el Backend

Sprint deadline	03/11/2023
Tarjeta	SCRUM-20
Responsable	Malleret, Luciano Joaquín

Tabla 1 Detalle de la tarjeta correspondiente en Jira

1. Objetivos. Contenido de la tarjeta:

- Controller: Definir API.
- Model:
 - Definir constraints.
 - Definir DTO para request y response.
- Service:
 - Validación de información recibida.
 - Validaciones previas a interacciones con la BD.
 - Logica de negocio.
- IRepository.

2. Dependencias

Fue necesario solamente la implementación del repositorio con los avances realizados para poder realizar la tarea.

3. Procedimientos

3.1. Creación de la rama SCRUM-20

Se creó la rama SCRUM-20 y se realizó un Pull a la rama Main de Git para trabajar con la versión estable del proyecto.

3.2. Creación de las clases e interfaces

Siguiendo como modelo lo que se realizó en el SCRUM-12 se procedió a crear las siguientes clases e interfaces:

- /controller
 - ClientController
- /model
 - /client
 - /dto
 - ClientRequest
 - ClientResponse
 - UpdateClientRequest
 - Client
 - ClientFactory
- /repository
 - ClientRepository
- /service
 - /interface
 - ClientService
 - ClientServiceImpl

3.3. Programación del modelo Cliente y dtoCliente

Se creó el modelo Cliente tomando como base el diagrama UML y ER para crear el modelo con sus atributos usando SpringBoot para el manejo de entidades y tablas. También se implementó SoftDeletable para la baja lógica.

Los dtos se programaron en base a las consultas y respuestas que se harían en el intercambio de mensajes entre el Front y el Back.

En el archivo 'EntitiesConstraints' se agregaron constraints para validar algunos atributos de Cliente.

3.4. Programación del Repository de Cliente

Se configuró el Repository para Cliente con Jpa usando como atributos el Id, Dni y Cuit de Cliente.

3.5. Programación del Service de Cliente

Se programó la interfaz 'ClientService' con los métodos necesarios para el CRUD de Cliente. Dicha interfaz se implementó en la clase 'ClientServiceImpl' donde se programó la lógica necesaria para que funcione correctamente el CRUD de Cliente.

Se modificó la interfaz 'Validator' y la clase 'ValidatorImpl'. Se agregaron validaciones necesarias para evitar posibles errores por parte del Back.

3.6. Programación del Controller de Cliente

Se programó la clase 'ClientController' con los métodos y end point necesarios para que el Front pueda comunicarse con el Back.

4. Resultados

4.1. API y Postman

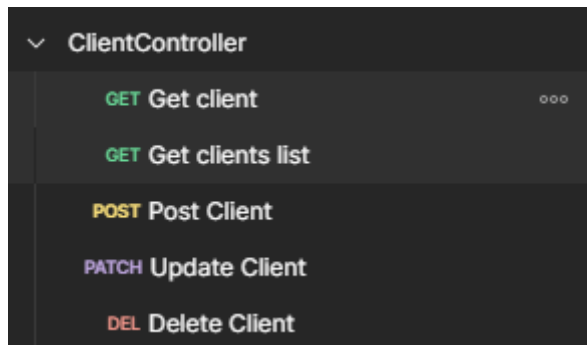


Ilustración 1: Captura de pantalla de Postman de la API para el CRUD de cliente

4.2. Get cliente y lista de clientes

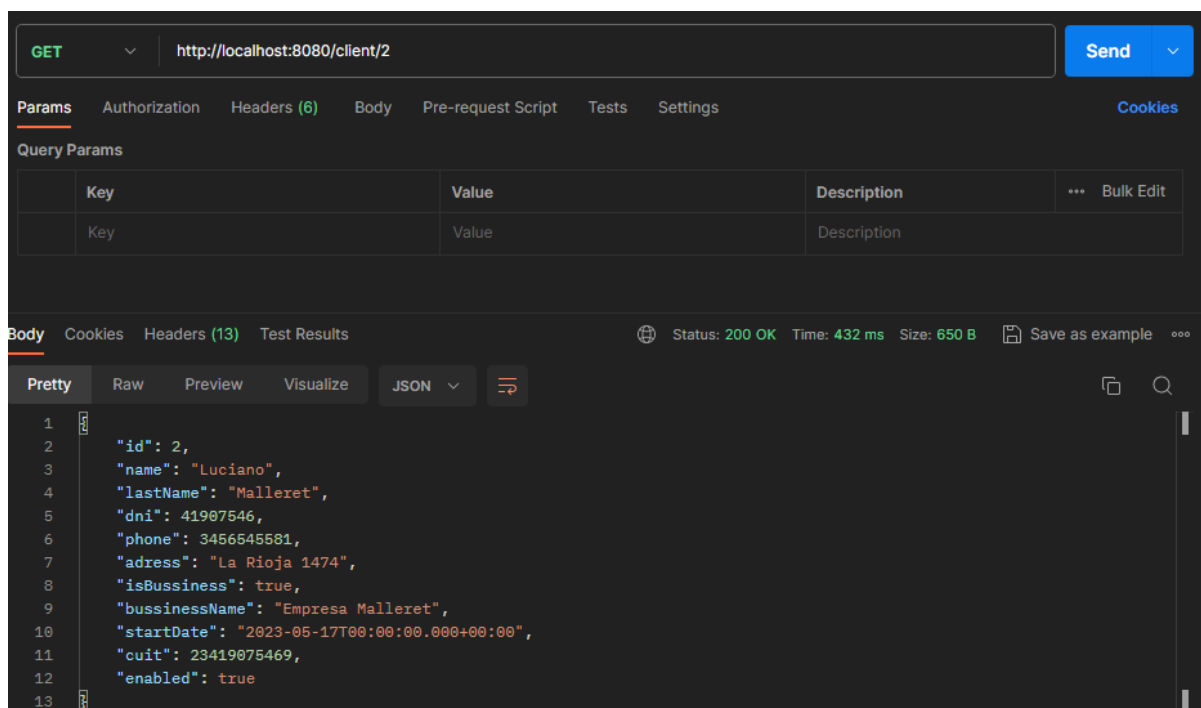


Ilustración 2: Captura de pantalla con consulta por un cliente con id 2 y respuesta del back

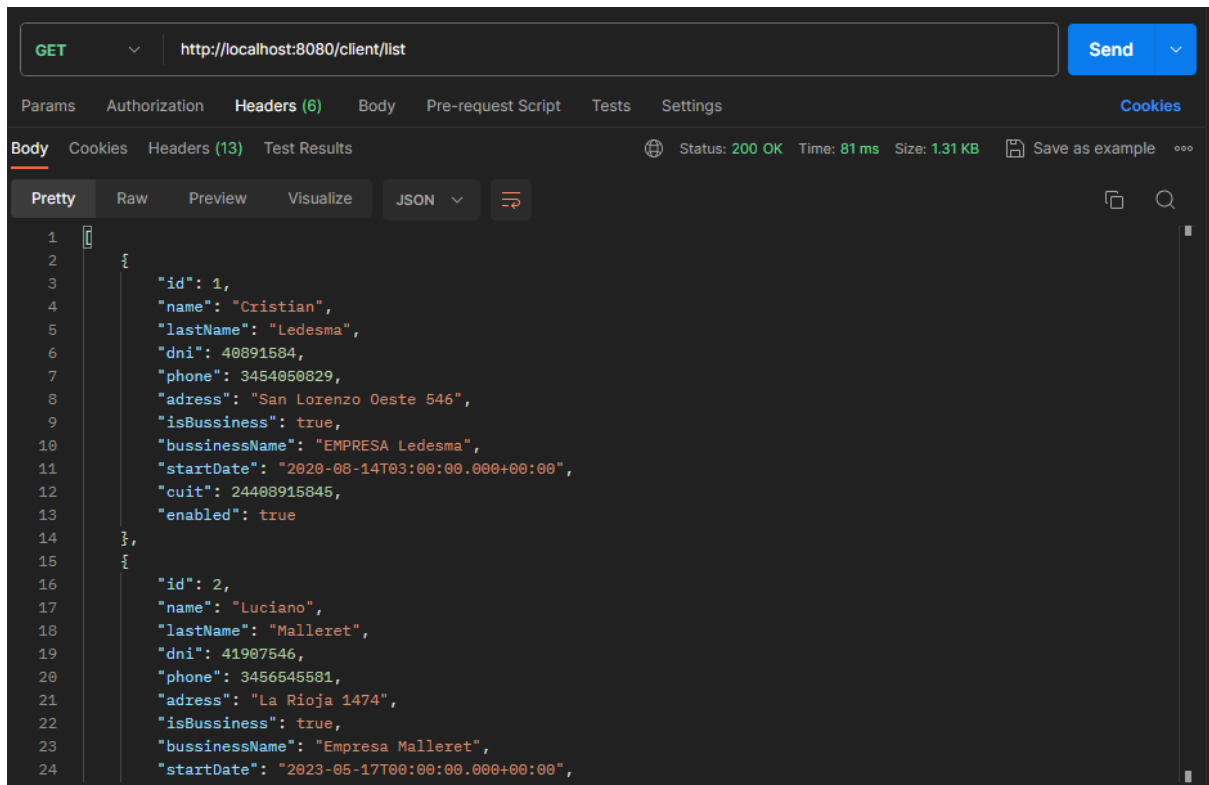


Ilustración 3: Captura de pantalla con consulta de lista de clientes y respuesta del back

4.3. Create Cliente

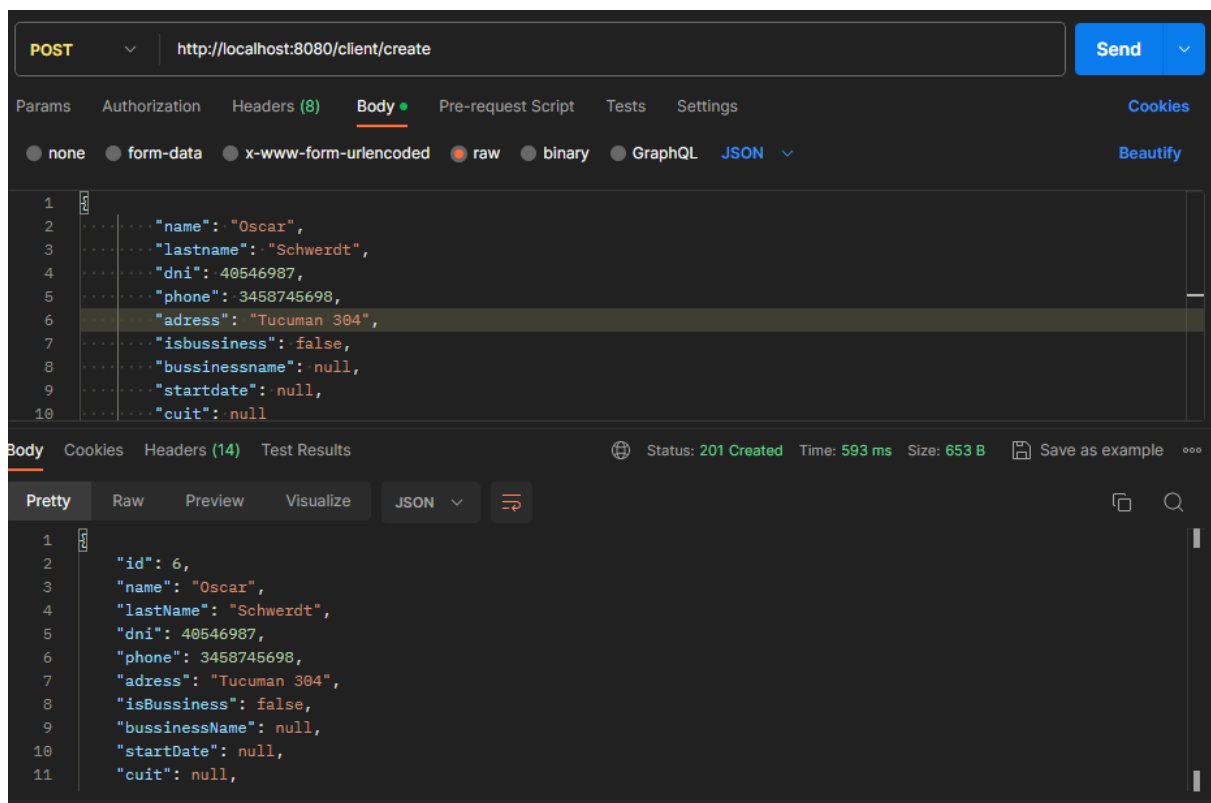
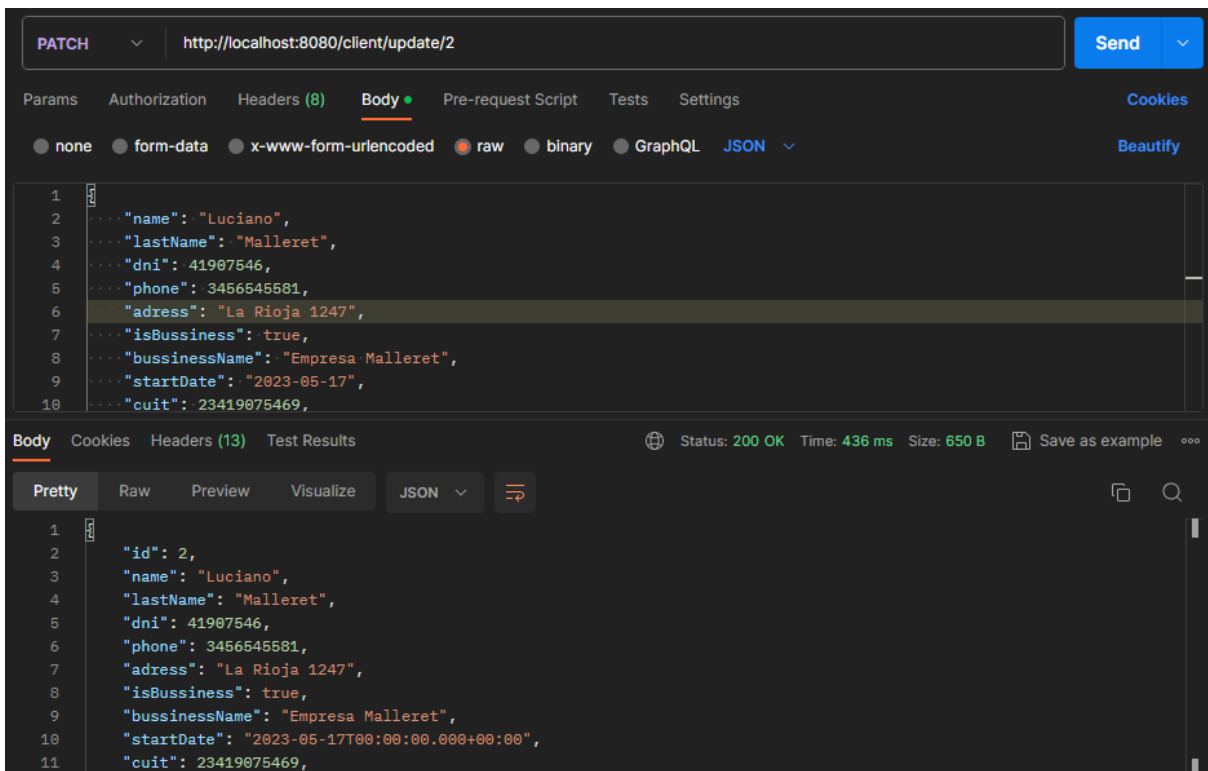


Ilustración 4: Captura de pantalla con consulta para crear un usuario y respuesta del back

	id	client_address	client_business_name	clientcuit	clientdni	enabled	is_business	client_last_name	client_name	client_phone	client_start_date
1	1	San Lorenzo Oeste 546	EMPRESA Ledesma	24408915845	40891584	1	1	Ledesma	Cristian	3454050829	2020-08-14 00:00:00.0000000
2	2	Espejo 1804	Empresa Malleret	23419075469	41907546	1	1	Malleret	Luciano	3456545581	2023-05-16 21:00:00.0000000
3	4	Rodriguez Peña 102	EMPRESA Medina	23416458719	41645871	1	1	Medina	Dario	3454975463	2019-06-15 00:00:00.0000000
4	5	Rodriguez Peña 102	NULL	NULL	42356891	1	0	Leivas	Marcos Joel	3454975463	NULL
5	7	Tucuman 304	NULL	NULL	40546987	1	0	Schwerdt	Oscar	3458745698	NULL

Ilustración 5: Captura de pantalla con comprobación de correcta creación de cliente

4.4. Update Cliente



PATCH `http://localhost:8080/client/update/2` **Send**

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

Body Cookies Headers (13) Test Results Status: 200 OK Time: 436 ms Size: 650 B Save as example

Body Pretty Raw Preview Visualize JSON

```

1 {
2   "name": "Luciano",
3   "lastName": "Malleret",
4   "dni": 41907546,
5   "phone": 3456545581,
6   "address": "La Rioja 1247",
7   "isBusiness": true,
8   "businessName": "Empresa Malleret",
9   "startDate": "2023-05-17",
10  "cuit": 23419075469,

```

```

1 {
2   "id": 2,
3   "name": "Luciano",
4   "lastName": "Malleret",
5   "dni": 41907546,
6   "phone": 3456545581,
7   "address": "La Rioja 1247",
8   "isBusiness": true,
9   "businessName": "Empresa Malleret",
10  "startDate": "2023-05-17T00:00:00.000+00:00",
11  "cuit": 23419075469,

```

Ilustración 6: Captura de pantalla consulta por actualización de cliente y respuesta del back

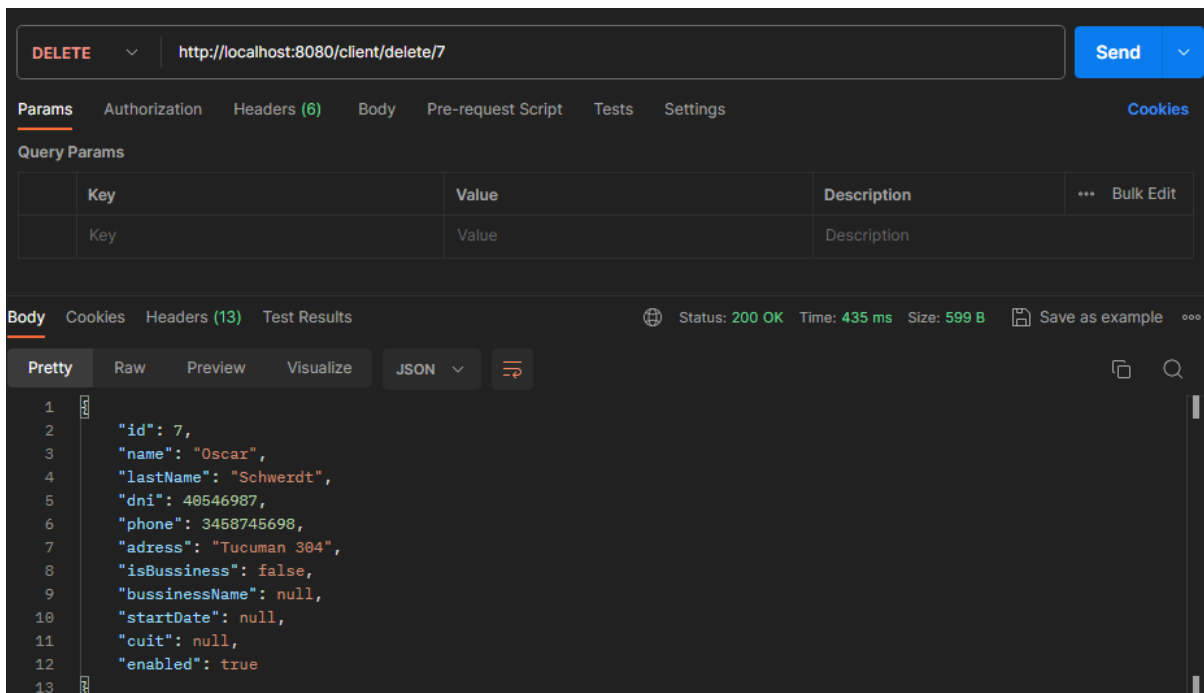
	id	client_address	client_business_name	clientcuit	clientdni	enabled	is_business	client_last_name	client_name	client_phone	client_start_date
1	1	San Lorenzo Oeste 546	EMPRESA Ledesma	24408915845	40891584	1	1	Ledesma	Cristian	3454050829	2020-08-14 00:00:00.0000000
2	2	Espejo 1804	Empresa Malleret	23419075469	41907546	1	1	Malleret	Luciano	3456545581	2023-05-16 21:00:00.0000000
3	4	Rodriguez Peña 102	EMPRESA Medina	23416458719	41645871	1	1	Medina	Dario	3454975463	2019-06-15 00:00:00.0000000
4	5	Rodriguez Peña 102	NULL	NULL	42356891	1	0	Leivas	Marcos Joel	3454975463	NULL
5	7	Tucuman 304	NULL	NULL	40546987	1	0	Schwerdt	Oscar	3458745698	NULL

Ilustración 7: Antes de la actualización

	id	client_address	client_business_name	clientcuit	clientdni	enabled	is_business	client_last_name	client_name	client_phone	client_start_date
1	1	San Lorenzo Oeste 546	EMPRESA Ledesma	24408915845	40891584	1	1	Ledesma	Cristian	3454050829	2020-08-14 00:00:00.0000000
2	2	La Rioja 1247	Empresa Malleret	23419075469	41907546	1	1	Malleret	Luciano	3456545581	2023-05-16 21:00:00.0000000
3	4	Rodriguez Peña 102	EMPRESA Medina	23416458719	41645871	1	1	Medina	Dario	3454975463	2019-06-15 00:00:00.0000000
4	5	Rodriguez Peña 102	NULL	NULL	42356891	1	0	Leivas	Marcos Joel	3454975463	NULL
5	7	Tucuman 304	NULL	NULL	40546987	1	0	Schwerdt	Oscar	3458745698	NULL

Ilustración 8: Después de la actualización

4.5. Delete Cliente



The screenshot shows a REST client interface with a DELETE request to `http://localhost:8080/client/delete/7`. The response is a JSON object with the following fields:

```

{
  "id": 7,
  "name": "Oscar",
  "lastName": "Schwerdt",
  "dni": 40546987,
  "phone": 3458745698,
  "adress": "Tucuman 304",
  "isBussiness": false,
  "bussinessName": null,
  "startDate": null,
  "cuit": null,
  "enabled": true
}

```

Ilustración 9: Captura de pantalla con consulta por eliminación de cliente con id 7 y respuesta del back

	id	client_adress	client_bussiness_name	clientcuit	clientdni	enabled	is_bussiness	client_last_name	client_name	client_phone	client_start_date
1	1	San Lorenzo Oeste 546	EMPRESA Ledesma	24408915845	40891584	1	1	Ledesma	Cristian	3454050829	2020-08-14 00:00:00.000000
2	2	La Rioja 1247	Empresa Halleret	23419075469	41907546	1	1	Halleret	Luciano	3456545581	2023-05-16 21:00:00.000000
3	4	Rodriguez Peña 102	EMPRESA Medina	23416458719	41645871	1	1	Medina	Dario	3454975463	2019-06-15 00:00:00.000000
4	5	Rodriguez Peña 102	NULL	NULL	42356891	1	0	Leivas	Marcos Joel	3454975463	NULL

Ilustración 10: Comprobación de la correcta eliminación del cliente

4.6. Pruebas de validación en las consultas

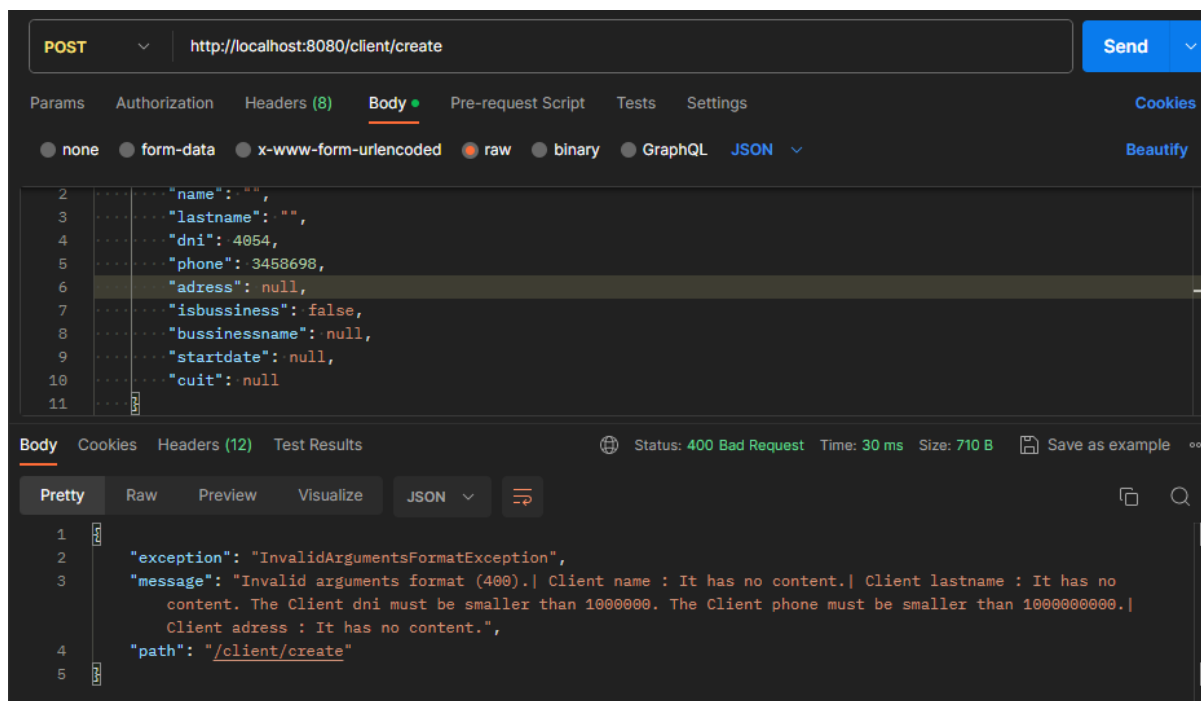


Ilustración 11: Captura de pantalla intento de ingresar un cliente con campos invalidos

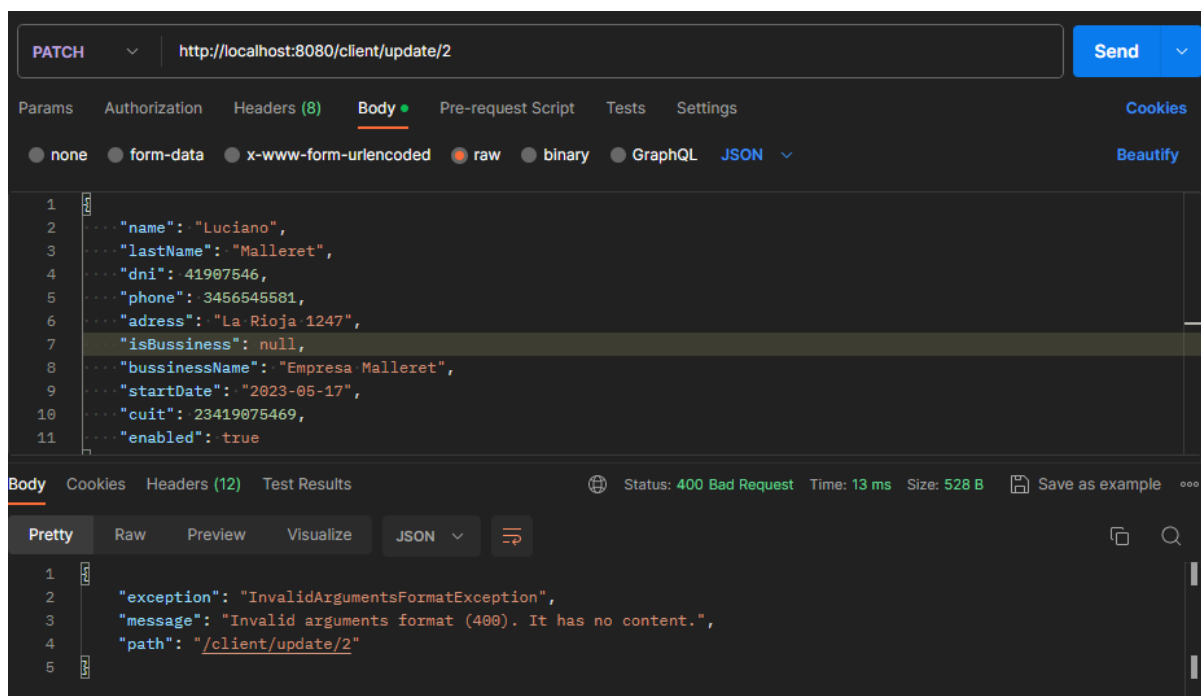


Ilustración 12: Captura de pantalla con intento de actualizar in cliente con campos invalidos

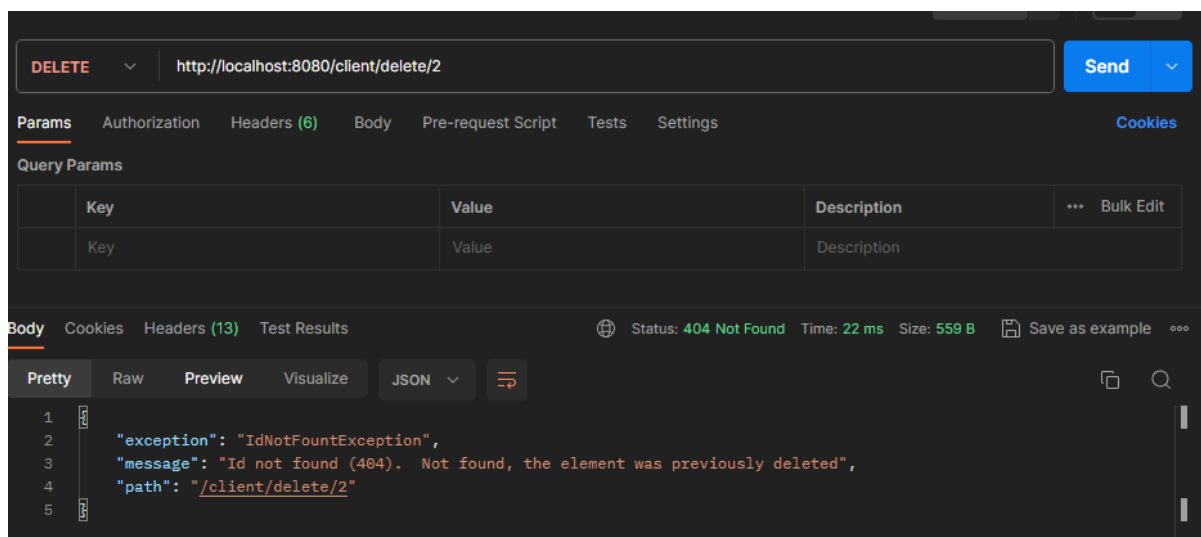


Ilustración 13: Captura de pantalla con intento de eliminar un cliente que ya esté dado de baja

5. Conclusión

La aplicación ahora cuenta con un CRUD de Cliente con sus end points, consultas y respuestas, y validaciones correspondientes.