

Taller Microservicios



Universidad
del Cauca

Vigilada Mineducación

Ingeniería de Software II

Presentado por:

Dana Isabella Romero Núñez

Profesor:

Wilson Libardo Pantoja Yopez

Universidad del Cauca

Facultad de Ingeniería Electrónica Y Telecomunicaciones

Ingeniería De Sistemas

Popayán, Septiembre de 2025

CONTENIDO

INTRODUCCIÓN.....	3
CREANDO UN MICROSERVICIO QUE HACE EL CRUD DE UNA ENTIDAD.....	1
URL para clonar el proyecto.....	1
Evidencia 1 - GET ALL.....	1
Evidencia 2 - GET ONE.....	2
Evidencia 3 - CREATE.....	3
Evidencia 4 - UPDATE.....	3
Evidencia 5 - DELETE.....	4
Evidencia 6 - TABLA AUDITORIA.....	4
Referencias.....	5

INTRODUCCIÓN

Este taller busca introducirnos en la creación de servicios web utilizando Spring Boot, un framework muy popular para el desarrollo de aplicaciones en Java. Durante el ejercicio se trabaja especialmente el concepto de API REST, entendiendo su lógica y cómo permite la comunicación entre cliente y servidor a través del protocolo HTTP, haciendo uso de verbos como GET, POST, PUT y DELETE para gestionar los recursos.

La práctica consiste en desarrollar un microservicio CRUD con una arquitectura en capas e integración de JPA para manejar la persistencia de los datos. Con esto, se construye una base sólida para comprender cómo se diseñan y consumen servicios REST en aplicaciones actuales, resaltando su papel clave en la comunicación y la integración de sistemas modernos.

CREANDO UN MICROSERVICIO QUE HACE EL CRUD DE UNA ENTIDAD

Para el desarrollo de este ejercicio se seguirán las instrucciones proporcionadas en el enlace compartido por el profesor

URL para clonar el proyecto

<https://github.com/isaromn/proyecto-microservicio.git>

Evidencia 1 - GET ALL

The screenshot shows a REST client interface with a tab for 'GET GET ALL'. The request is a GET method to the URL 'http://localhost:9000/api/v1/personas'. The response is a 200 OK status with a response time of 199 ms and a body size of 630 B. The response body is a JSON array of 7 objects, each representing a person with fields: id, nombre, apellido, and dni. The 'Preview' button is highlighted.

	id	nombre	apellido	dni
0	1	Florencia	Salcedo	40101774
1	3	Aleja	Pinto	1234567
2	4	Sebastian	Caicedo	7891234
3	5	Juliana	Sanchez	3458932
4	6	Laura	Castrillon	2348756
5	7	Dana	Romero	1276853

Evidencia 2 - GET ONE

GET GET ALI • GET GET ON • POST CREATI PUT UPDATE DEL DELETE • +

No environment

PERSONA API REQUEST / GET ONE

Save Share

GET

http://localhost:9000/api/v1/personas/4

Send

Params Auth Headers (6) Body Scripts Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

200 OK • 56 ms • 317 B • Save Response

{ } JSON

Preview Visualize

⌵ ⌵ 🔍 📄 🔗

```
1 {
2   "id": 4,
3   "nombre": "Sebastian",
4   "apellido": "Caicedo",
5   "dni": 7891234
6 }
```

Evidencia 3 - CREATE

GET GET ALI • GET GET ON • **POST CREAT** • PUT UPDATE • DEL DELETE • + ▾ No environment ▾

PERSONA API REQUEST / **CREATE** Save ▾ Share ▾

POST ▾ http://localhost:9000/api/v1/personas **Send** ▾

Params Auth Headers (9) **Body** • Scripts Settings Cookies

raw ▾ **JSON** ▾ Schema Beautify

```
1 {
2   "nombre": "Alejandra",
3   "apellido": "Núñez",
4   "dni": 87656778
5 }
```

Body ▾ 200 OK • 71 ms • 318 B • Save Response ...

{ } JSON ▾ Preview Visualize ▾

```
1 {
2   "id": 8,
3   "nombre": "Alejandra",
4   "apellido": "Núñez",
5   "dni": 87656778
6 }
```

Evidencia 4 - UPDATE

Workspaces ▾ More ▾ Upgrade ▾

PERSONA API REQUEST / **UPDATE** Save ▾ Share ▾

PUT ▾ http://localhost:9000/api/v1/personas/8 **Send** ▾

Params Auth Headers (9) **Body** • Scripts Settings Cookies

raw ▾ **JSON** ▾ Schema Beautify

```
1 {
2   "id": 8,
3   "nombre": "Leidy",
4   "apellido": "Núñez",
5   "dni": 87656778
6 }
```

Body ▾ 200 OK • 27 ms • 314 B • Save Response ...

{ } JSON ▾ Preview Visualize ▾

```
1 {
2   "id": 8,
3   "nombre": "Leidy",
4   "apellido": "Núñez",
5   "dni": 87656778
6 }
```

Evidencia 5 - DELETE

The screenshot shows the REST Client interface. At the top, there are tabs for different HTTP methods: GET, POST, PUT, and DELETE. The DELETE tab is selected. Below the tabs, the URL bar shows 'http://localhost:9000/api/v1/personas/1'. The 'Headers' tab is selected, and it shows a table with 'Key' and 'Value' columns. The 'Body' tab is also visible. At the bottom, the response is shown as '204 No Content'.

Evidencia 6 - TABLA AUDITORIA

1 Messages						
2 Table Data						
3 Info						
Limit rows First row 0						
<input type="checkbox"/>	id	rev	revtype	apellido	dni	nombre
<input type="checkbox"/>	9	6	0	Núñez	87656778	Leidy
<input type="checkbox"/>	8	7	1	Núñez	87656778	Leidy
<input type="checkbox"/>	1	8	2	(NULL)	(NULL)	(NULL)
*	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Referencias

- *Getting Started | Building a RESTful Web Service*. (s. f.). Getting Started | Building A RESTful Web Service. <https://spring.io/guides/gs/rest-service>