

# Probar las APIs

# Ejecutar Spring Boot

## Levantar Spring Boot, en el IDE.

The screenshot shows an IDE interface with several panes:

- Left pane (Code Editor):** Displays the Java code for `DemoApplication`. The `main` method is highlighted.
- Right pane (Outline View):** Shows the class hierarchy: `com.example.demo` contains `DemoApplication`, which has a method `main(String[] args)`.
- Bottom pane (Terminal):** Shows the command-line interface with the Spring Boot logo and version information, followed by the log output for the application.

```
2
3* import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DemoApplication.class, args);
10    }
11 }
12
13 }
14
```

Problems Javadoc Declaration Console ×  
DemoApplication [Java Application] C:\Users\lizju\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot\jre.full.win32.x86\_64\_17.0.11.v20240426-1830\jre\bin\javaw.exe (May 22, 2024, 12:15:44 PM) [pid: 15408]

```
.\_\_\_\_.
\_\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_
(( ))\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_
\_\_\_/\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_
=====\_|=====|/_=/\_/\_/\_
:: Spring Boot ::          (v3.2.5)

2024-05-22T12:15:47.611-03:00 INFO 15408 --- [demo] [main] com.example.demo.DemoApplication : Starting DemoApplication using Java 17
2024-05-22T12:15:47.621-03:00 INFO 15408 --- [demo] [main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles
2024-05-22T12:15:49.385-03:00 INFO 15408 --- [demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http://localhost:8080)
2024-05-22T12:15:49.401-03:00 INFO 15408 --- [demo] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-05-22T12:15:49.401-03:00 INFO 15408 --- [demo] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/17.0.11]
2024-05-22T12:15:49.519-03:00 INFO 15408 --- [demo] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-05-22T12:15:49.524-03:00 INFO 15408 --- [demo] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialized at [2024-05-22T12:15:49.524Z]
2024-05-22T12:15:50.126-03:00 INFO 15408 --- [demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http://localhost:8080) with 1 context(s); started in [3.188s]
2024-05-22T12:15:50.158-03:00 INFO 15408 --- [demo] [main] com.example.demo.DemoApplication : Started DemoApplication in 3.188 seconds (version: v3.2.5, environment: dev, profiles: none)
```

# Postman

Es una herramienta que, **entre otras cosas, permite probar APIs.**

Fue creada para facilitar el trabajo de desarrolladores. Permite **enviar solicitudes HTTP y ver las respuestas de forma rápida y sencilla.**

Algunas de sus funciones principales incluyen:  
Manejo de colecciones, entornos y variables,  
pruebas automatizadas, documentación de API,  
monitoreo de APIs.



Se puede descargar de [Postman.com](https://www.postman.com).

## POST

En Postman, seleccionar **JSON** y agregar a una persona.

The screenshot shows the Postman application interface. At the top, there is a header bar with the URL `http://localhost:8080/personas`. Below the header, the method is set to `POST` and the `Body` tab is selected. The body content is a JSON object representing a person:

```
1 {
2     "id": 1,
3     "nombre": "Juan Pérez",
4     "edad": 30,
5     "direccion": {
6         "calle": "Calle Falsa 123",
7         "ciudad": "Madrid",
8         "codigoPostal": 28013
9     },
10    "telefono": "+34 600 123 456",
11    "email": "juan.perez@example.com"
```

Below the body content, the status bar indicates a `201 Created` response with a size of `351 B`. The bottom section of the interface shows the `Raw` tab selected, displaying the JSON response:

```
{"id":1,"nombre":"Juan Pérez","edad":30,"direccion":{"calle":"Calle Falsa 123","ciudad":"Madrid","codigoPostal":28013},"telefono":"+34 600 123 456","email":"juan.perez@example.com"}
```

## GET

Cuidado con este error:

HTTP http://localhost:8080/personas/

GET http://localhost:8080/personas/

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

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

1

Body Cookies Headers (8) Test Results Status: 4

Pretty Raw Preview Visualize JSON

```
1 "timestamp": "2024-05-22T15:42:22.531+00:00",
2 "status": 404,
3 "error": "Not Found",
4 "path": "/personas/"
```

Ocurre porque la URL correcta es  
**http://localhost:8080/personas**  
sin barra “/” al final.



Si se coloca **/personas** con GET,  
traerá una lista de personas.



GET http://localhost:8080/personas

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

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

1

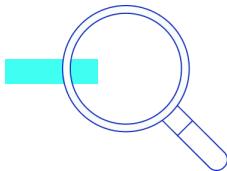
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
15   "id": 1,
16   "nombre": "Juan Pérez",
17   "edad": 30,
18   "direccion": {
19     "calle": "Calle Falsa 123",
20     "ciudad": "Madrid",
21     "codigoPostal": 28013
22   },
23   "telefono": "+34 600 123 456",
24   "email": "juan.perez@example.com"
```

## GET by id

Si se coloca `/personas/{id}` con GET, traerá una persona por su `id`.



GET

Params Authorization Headers (6) Body Pre-request Script Test

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

1

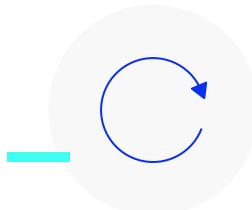
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON 

```
1
2   "id": 1,
3   "nombre": "Juan Pérez",
4   "edad": 30,
5   "direccion": {
6     "calle": "Calle Falsa 123",
7     "ciudad": "Madrid",
8     "codigoPostal": 28013
9   },
10  "telefono": "+34 600 123 456",
```

## PUT

Si se coloca `/personas/{id}` con PUT, actualizará una persona por su `id`.



PUT http://localhost:8080/personas/1

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

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

```
1 ... "id": 1,
2 ... "nombre": "Maxi Pérez",
3 ... "edad": 30,
4 ... "direccion": {
5 ...     "calle": "Calle Falsa 123",
6 ...     "ciudad": "Madrid",
7 ...     "codigoPostal": 28013
8 ... },
9 ... "telefono": "+34 600 123 456",
10 ... "email": "juan.perez@example.com"
```

Body Cookies Headers (5) Test Results

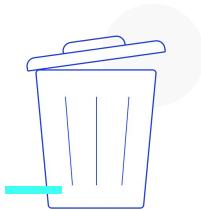
Pretty Raw Preview Visualize JSON

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
7 ...
8 ...
9 ...
10 ...
```

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
7 ...
8 ...
9 ...
10 ...
```

## DELETE

Si se coloca `/personas/{id}` con DELETE, borra una persona por su `id`.



HTTP <http://localhost:8080/personas/>

DELETE <http://localhost:8080/personas/1>

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

none  form-data  x-www-form-urlencoded  raw  binary **JSON** ▾

A screenshot of a Postman API testing interface. The URL is set to `http://localhost:8080/personas/`. A `DELETE` request is selected, and the endpoint is specified as `http://localhost:8080/personas/1`. The `Body` tab is active, showing the `x-www-form-urlencoded` option selected. Other tabs like `Params`, `Authorization`, and `Headers` are visible but inactive. Below the tabs, there are radio buttons for `none`, `form-data`, `x-www-form-urlencoded`, `raw`, `binary`, and a dropdown menu currently set to `JSON`.