



UNIVERSIDAD POLITECNICA  
INTERDISCIPLINARIA EN INGENIERIA  
Y TECNOLOGIAS AVANZADAS.



## PRACTICA 2

Materia :Aplicaciones distribuidas

Profesor : Noe Sierra Romero

TITULO :

---

### Práctica de repaso para API Restful

---

FECHA : 21 de Febrero del 2026. Hecho por :

Flores Hernandez Lluvia Nahivy  
Boleta : 2022640090

## Desarrollo de practica 2

### Ejercicio 1 : Servicio de Saludo Básico

The screenshot shows the Postman application interface. At the top, there is a header bar with the URL `http://localhost:3000/saludo`, a `Save` button, a `Share` button, and a copy icon. Below the header, the main interface has a search bar with `POST` selected and the URL `http://localhost:3000/saludo`, a `Send` button, and a dropdown menu. The main workspace is titled `Body` and has tabs for `Docs`, `Params`, `Auth`, `Headers (9)`, `Body` (selected), `Scripts`, `Tests`, `Settings`, and `Cookies`. Under the `Body` tab, there are two dropdown menus: `raw` and `JSON`. The `JSON` menu is currently selected. Below these menus, there is a code editor containing the following JSON:

```
1 {  
2   "nombre": "Lluvia Flores"  
3 }
```

At the bottom of the main workspace, there is a status bar showing `200 OK`, `5 ms`, `281 B`, and other metrics. Below the status bar, there is another section with a `Body` dropdown, a preview icon, and a `Visualize` button. The `Visualize` section shows the received JSON response:

```
1 {  
2   "estado": 200,  
3   "mensaje": "Hola, Lluvia Flores"  
4 }
```

At the very bottom of the interface, there are several navigation icons: Cloud View, Console, Terminal, Runner, Vault, and others.

Recibimos de regreso un saludo con el nombre de la persona, cualquier tipo de nombre que proporcione con una longitud menor o igual a 200 caracteres.

## Ejercicio 2 : Calculadora de Operaciones Básicas

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:3000/calcular`. Below the URL, there's a dropdown menu showing `POST` selected. To the right of the URL, there are buttons for `Save`, `Share`, and a copy icon. Further right is a `</>` button.

The main area is divided into sections: `Docs`, `Params`, `Auth`, `Headers (9)`, `Body` (which is currently selected), `Scripts`, `Tests`, `Settings`, and `Cookies`. Under `Body`, there are two options: `raw` and `JSON`. The `JSON` option is selected, and its dropdown shows `raw` and `Beautify`. The JSON payload is:

```
1 {  
2   "a": 15,  
3   "b": 5,  
4   "operacion": "suma"  
5 }
```

Below the body section, the response details are shown: `200 OK`, `5 ms`, `264 B`, and a globe icon indicating the response is local. The response body is displayed in JSON format:

```
{ } JSON ▾ ▶ Preview ▾ Visualize ▾
```

```
1 {  
2   "estado": 200,  
3   "resultado": 20  
4 }
```

At the bottom of the interface, there are several navigation icons: Cloud View, Console, Terminal, Runner, Vault, and others.

Resuelve sumas, si llegaran a ingresar letras, marcaria un error.

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:3000/calcular`. Below the URL, there's a dropdown for method (`POST`) and a target URL (`http://localhost:3000/calcular`). A large blue button labeled `Send` is prominently displayed. The main area is titled "Body" and has tabs for "raw" (selected) and "JSON". The JSON content is:

```
1 {
2   "a": 15,
3   "b": 5,
4   "operacion": "resta"
5 }
```

Below the body, the response status is `200 OK`, with a response time of `5 ms` and a size of `264 B`. The response content is:

```
{}
{
  "estado": 200,
  "resultado": 10
}
```

At the bottom, there are various navigation and utility buttons.

Devuelve el resultado de una resta.

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:3000/calcular`. Below the URL, a `POST` method is selected. The request body is displayed in JSON format:

```
1 {  
2   "a": 15,  
3   "b": 5,  
4   "operacion": "multiplicacion"  
5 }
```

The response status is `200 OK` with a response time of 3 ms and a size of 264 B. The response body is also in JSON format:

```
{ } JSON ▾ ▶ Preview ▶ Visualize ▾  
1 {  
2   "estado": 200,  
3   "resultado": 75  
4 }
```

Devuelve el resultado de una multiplicación.

The screenshot shows a Postman interface. At the top, the URL is `http://localhost:3000/calcular`. Below it, a POST request is being sent to the same URL. The Body tab is selected, showing a JSON payload:

```
1 {
2   "a": 15,
3   "b": 5,
4   "operacion": "division"
5 }
```

Below the request, the response is displayed. It shows a green `200 OK` status with a response time of 6 ms and a size of 263 B. The response body is also JSON:

```
1 {
2   "estado": 200,
3   "resultado": 3
4 }
```

At the bottom, there are various navigation and utility buttons.

Devuelve el resultado de una división.

### Ejercicio 3 : Gestor de Tareas (CRUD Básico)

The screenshot shows the Postman interface. At the top, the URL is set to `http://localhost:3000/tareas`. Below the URL, there's a dropdown menu showing `POST` selected. To the right of the URL, there are buttons for `Save`, `Share`, and a copy icon. Further right is a code editor area with the text `</>`.

The main workspace has tabs for `Docs`, `Params`, `Auth`, `Headers (9)`, `Body` (which is currently active), `Scripts`, `Tests`, `Settings`, and `Cookies`. Under the `Body` tab, there are two dropdowns: `raw` and `JSON`. The `JSON` dropdown is selected, and its value is:

```
1 {  
2   "id": 1,  
3   "titulo": "Terminar mi tarea de aplicaciones distribuidas",  
4   "completada": false  
5 }  
6
```

Below the body editor, the response status is shown as `200 OK` with a timestamp of `4 ms` and a size of `344 B`. The response payload is displayed in the `JSON` view:

```
1 {  
2   "estado": 201,  
3   "tarea": {  
4     "id": 1,  
5     "titulo": "Terminar mi tarea de aplicaciones distribuidas",  
6     "completada": false  
7   }  
8 }
```

At the bottom of the interface, there are several navigation and utility buttons: `Cloud View`, `Console`, `Terminal`, `Runner`, `Vault`, and others.

En este caso utiliza el método POST para guardar la tarea con el Id que le quieras asignar después para poder localizar la tarea vuelves a usar ese id.

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:3000/tareas/1`. The method is selected as `PUT`. The `Body` tab is active, showing the following JSON payload:

```
1 {
2   "titulo": "Terminar mi tarea de aplicaciones distribuidas",
3   "completada": true
4 }
```

Below the body, the response status is `200 OK` with a response time of `5 ms` and a size of `343 B`. The response body is displayed as:

```
1 {
2   "estado": 200,
3   "tarea": {
4     "id": 1,
5     "titulo": "Terminar mi tarea de aplicaciones distribuidas",
6     "completada": true
7   }
8 }
```

At the bottom of the interface, there are various navigation and utility buttons.

En este ejemplo se recupera la tarea que se ha guardado previamente. En este caso de una un método PUT.

## Ejercicio 4 : Validador de Contraseñas

The screenshot shows a Postman interface with the following details:

- URL:** `http://localhost:3000/validar-password`
- Method:** POST
- Body (JSON):**

```
1  {
2    "password": "Password123"
3  }
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1  {
2    "estado": 200,
3    "esValida": true,
4    "errores": []
5  }
```

Se hicieron diferentes validaciones para que la contraseña sera valida una vez que la ingresen.

The screenshot shows a Postman interface for testing an API endpoint at `http://localhost:3000/validar-password`. The request method is POST, and the body contains the JSON payload:

```
1 {
2   "password": "Paswwwoei"
3 }
4 
```

The response status is 200 OK, with a response time of 4 ms and a response size of 311 B. The response body is:

```
1 {
2   "estado": 200,
3   "esValida": false,
4   "errores": [
5     "Debe tener al menos un número"
6   ]
7 }
```

Esta opción para no hacer valida cuando lo ingresan una contraseña que le hacen falta caracteres en mayúsculas.

The screenshot shows a POST request to `http://localhost:3000/validar-password`. The request body is JSON, containing a single key-value pair: `"password": "13246t543665"`. The response status is `200 OK`, with a response time of 4 ms and a response size of 315 B. The response body is a JSON object with keys `estado`, `esValida`, and `errores`. The value for `estado` is `200`, `esValida` is `false`, and `errores` is a list containing the message `"Debe tener al menos una mayúscula"`.

```
1 {  
2   "password": "13246t543665"  
3 }  
  
1 {  
2   "estado": 200,  
3   "esValida": false,  
4   "errores": [  
5     "Debe tener al menos una mayúscula"  
6   ]  
7 }
```

Esta otra opción donde se argumenta que no ingresaron números en el password.

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:3000/validar-password`. The method is selected as `POST`, and the target URL is also `http://localhost:3000/validar-password`. The `Body` tab is active, showing the following JSON payload:

```
1 {  
2   "password": "13246543665.,,,"  
3 }  
4
```

Below the body, the response status is `200 OK` with a response time of `3 ms` and a size of `353 B`. The response content is displayed in JSON format:

```
1 {  
2   "estado": 200,  
3   "esValida": false,  
4   "errores": [  
5     "Debe tener al menos una mayúscula",  
6     "Debe tener al menos una minúscula"  
7   ]  
8 }
```

Al igual que en los anteriores faltó algún carácter por completar.

## Ejercicio 5 : Conversor de Temperatura

The screenshot shows a POST request to `http://localhost:3000/convertir-temperatura`. The request body is a JSON object:

```
1 {  
2   "valor": 100,  
3   "desde": "C",  
4   "hacia": "F"  
5 }
```

The response is a `200 OK` status with the following JSON data:

```
1 {  
2   "estado": 200,  
3   "valorOriginal": 100,  
4   "valorConvertido": 212,  
5   "escalaOriginal": "C",  
6   "escalaConvertida": "F"  
7 }
```

Below the interface, there are navigation links: Cloud View, Console, Terminal, Runner, Vault, and others.

El ejercicio pedía una conversión en los valores en que se mide la temperatura y se muestra la validación para la conversión en la temperatura, en este caso de C a F.

The screenshot shows a POST request to `http://localhost:3000/convertir-temperatura`. The request body is a JSON object:

```
1 {  
2   "valor": 123.12,  
3   "desde": "F",  
4   "hacia": "C"  
5 }
```

The response is a `200 OK` status with the following JSON data:

```
1 {  
2   "estado": 200,  
3   "valorOriginal": 123.12,  
4   "valorConvertido": 50.62,  
5   "escalaOriginal": "F",  
6   "escalaConvertida": "C"  
7 }
```

En este caso de ver el caso al contrario se convierte de F a C.

## Ejercicio 6 : Buscador en Array

The screenshot shows a Postman interface with the following details:

**Request URL:** `http://localhost:3000/buscar`

**Method:** POST

**Headers:** `Content-Type: application/json`

**Body (JSON):**

```
1 {
2   "array": ["manzana", "pera", "uva", "naranja"],
3   "elemento": "uva"
4 }
```

**Response:**

**Status:** 200 OK

**Time:** 3 ms

**Size:** 302 B

**Content:**

```
1 {
2   "estado": 200,
3   "encontrado": true,
4   "indice": 2,
5   "tipoElemento": "string"
6 }
```

Below the main interface, there are navigation links: Cloud View, Console, Terminal, Runner, Vault, and a help icon.

Dentro de una lista de objetos se pide ingresar uno. El programa te devolverá el objeto que seleccionaste y si realmente existe dentro de la lista.

The screenshot shows a POST request to `http://localhost:3000/buscar`. The request body is a JSON object:

```
1 {
2   "array": ["manzana", "pera", "uva", "naranja"],
3   "elemento": "melon"
4 }
```

The response status is `200 OK` with a response time of 5 ms, a size of 304 B, and a global icon. The response body is:

```
1 {
2   "estado": 200,
3   "encontrado": false,
4   "indice": -1,
5   "tipoElemento": "string"
6 }
```

En este ejemplo se buscó un objeto que no está dentro de la lista.

## Ejercicio 7 : Contador de Palabras

The screenshot shows a Postman interface for a POST request to `http://localhost:3000/contar-palabras`. The request body contains the JSON object `{"texto": "Hola mundo hola de nuevo mundo"}`. The response is a 200 OK status with a duration of 4 ms and a size of 307 B. The response body is a JSON object with the following structure:

```
1 {  
2   "estado": 200,  
3   "totalPalabras": 6,  
4   "totalCaracteres": 30,  
5   "palabrasUnicas": 4  
6 }
```

En este ejercicio se cuenta el total de palabras, los caracteres (se cuenta el carácter de espacio para hacer ese conteo) y nos muestra cuantas palabras no se repiten.

## Conclusiones

Las API son conocidas por poder estar en cualquier lugar y en cualquier dispositivo y una vez que se combina con Restful que trabaja directamente con JSON se vuelve en una transmisión mucho más rápida de la información, esto también debido a que utilizan el protocolo TCP y aunque también podrían utilizar algún otro protocolo como UDP este por ejemplo es muy más lento, también más confiable pero TCP tiene una velocidad que no se le comprara y en nuestra comunicación actual siempre buscamos la velocidad.