

## Actividad de la Lección

En esta actividad, los estudiantes aplicarán los conceptos aprendidos sobre la manipulación de datos y la configuración de respuestas JSON en Flask. El enfoque es desarrollar habilidades prácticas para diseñar estructuras de datos, leer y validar datos JSON, y configurar rutas en un servidor Flask funcional.

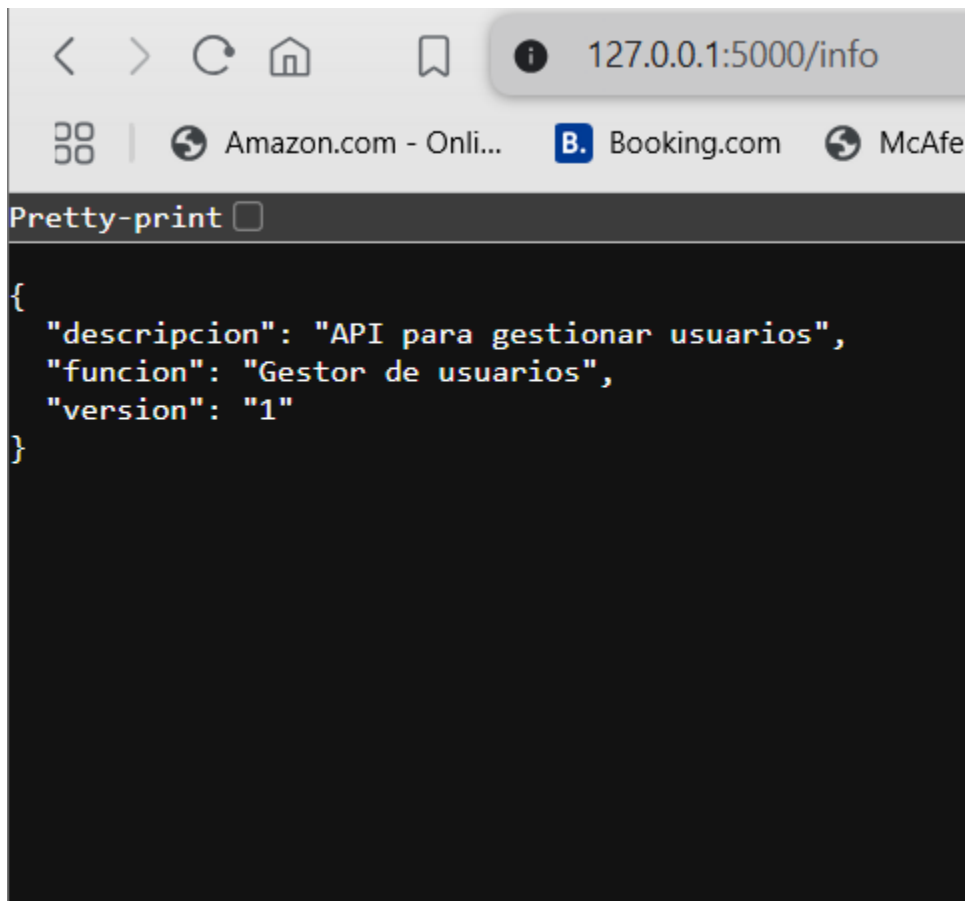
### Actividad

- **Ejercicio práctico:**
  - Crear un servidor Flask con las siguientes rutas:
  - GET /info: Devuelve información general sobre el sistema en formato JSON.
  - POST /crear\_usuario: Recibe datos de un usuario (nombre y correo) y valida que ambos datos están presentes.
  - GET /usuarios: Devuelve una lista de usuarios almacenados en memoria (diccionario o lista).
  - Asegurarse de manejar errores adecuadamente y devolver mensajes claros.
- **Propuesta de estructura de datos:**
  - Diseñar la estructura de datos para una aplicación que gestione productos y usuarios.
  - Subir la propuesta escrita en formato JSON.

Código:

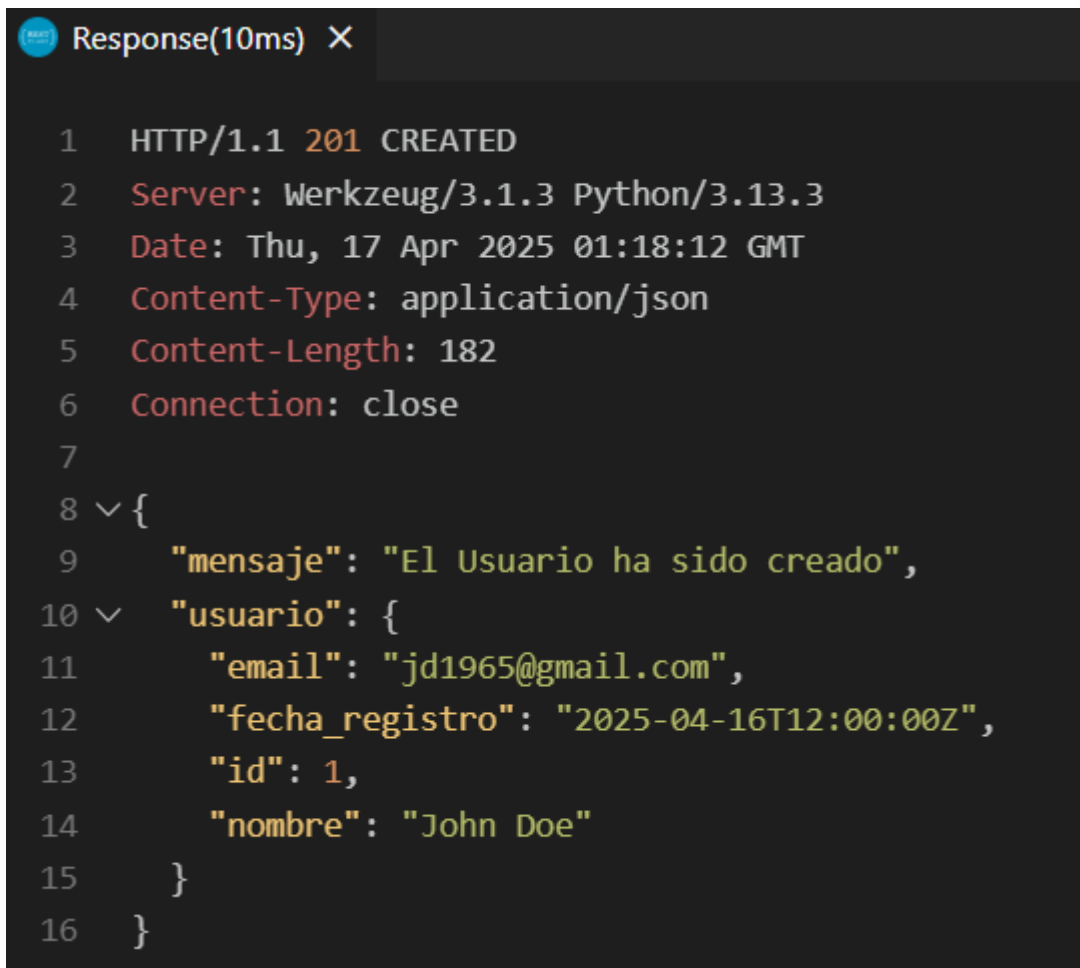
```
1  from flask import Flask, jsonify, request
2
3  app = Flask(__name__)
4
5  usuarios = [] # Lista para almacenar los usuarios
6
7  # Ruta GET /info
8  @app.route("/info", methods=["GET"])
9  def info():
10     return jsonify({
11         "funcion": "Gestor de usuarios",
12         "version": "1",
13         "descripcion": "API para gestionar usuarios"
14     })
15
16 # Ruta POST /createUser
17 @app.route("/createUser", methods=["POST"])
18 def crear_usuario():
19     data = request.get_json()
20     if "nombre" not in data or "email" not in data: # Verifica que la info incluya un nombre y un email
21         return jsonify({"error": "Se requieren 'nombre' y 'email'."}), 400
22
23     usuario = {
24         "id": len(usuarios) + 1, # Genera un id basado en la cantidad de usuarios en memoria
25         "nombre": data["nombre"],
26         "email": data["email"],
27         "fecha_registro": "2025-04-16T12:00:00Z" # Fecha de registro fija como ejemplo
28     }
29     usuarios.append(usuario)
30     return jsonify({"mensaje": "El Usuario ha sido creado", "usuario": usuario}), 201
31
32 # Ruta GET /usuarios
33 @app.route("/usuarios", methods=["GET"])
34 def obtener_usuarios():
35     return jsonify({"usuarios": usuarios})
36
37 if __name__ == "__main__":
38     app.run(debug=True)
```

Output del GET/info

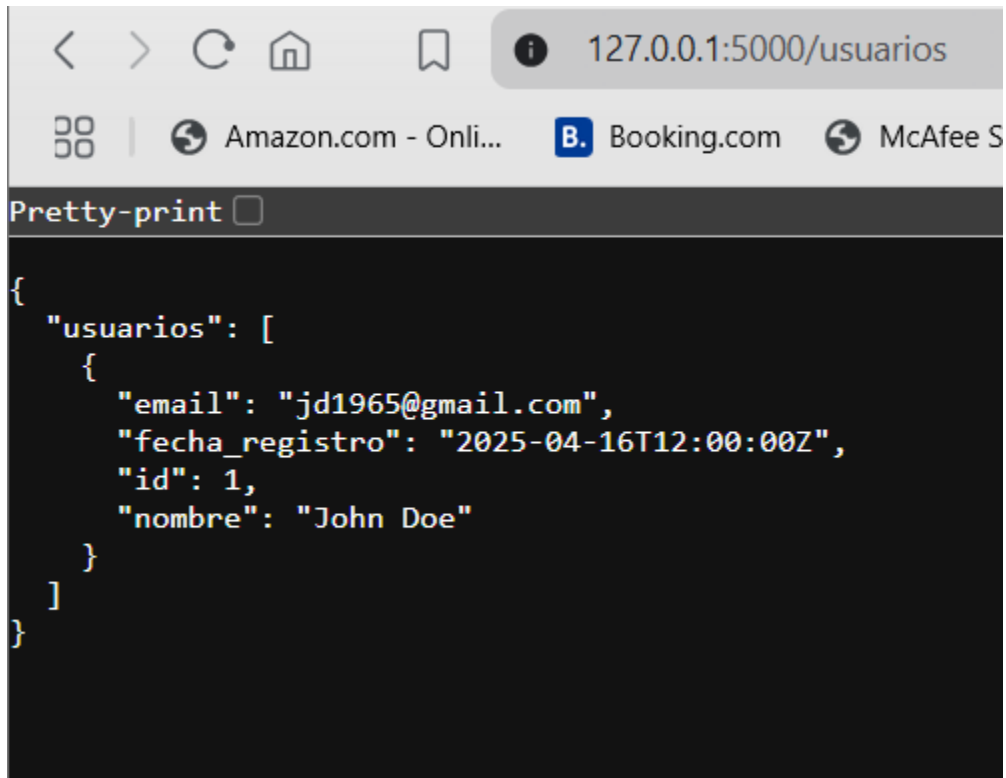


```
{
  "descripcion": "API para gestionar usuarios",
  "funcion": "Gestor de usuarios",
  "version": "1"
}
```

Output del Post:

A screenshot of a web browser's developer console. At the top, there's a tab labeled 'Response(10ms)' with a close button 'X'. The console shows an HTTP response with status 201 and reason 'CREATED'. The response headers are: 'Server: Werkzeug/3.1.3 Python/3.13.3', 'Date: Thu, 17 Apr 2025 01:18:12 GMT', 'Content-Type: application/json', 'Content-Length: 182', and 'Connection: close'. The response body is a JSON object. The first line of the JSON is '"mensaje": "El Usuario ha sido creado",' and the second line is '"usuario": {' followed by an indented object containing '"email": "jd1965@gmail.com",' '"fecha\_registro": "2025-04-16T12:00:00Z",' '"id": 1,' and '"nombre": "John Doe"'. The JSON object is closed with '}' and the entire response body is closed with '}'.

Output del GET/usuarios



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/usuarios`. Below the address bar, there are tabs for `Amazon.com - Onli...`, `Booking.com`, and `McAfee S`. A `Pretty-print` button is visible. The main content area displays the following JSON output:

```
{
  "usuarios": [
    {
      "email": "jd1965@gmail.com",
      "fecha_registro": "2025-04-16T12:00:00Z",
      "id": 1,
      "nombre": "John Doe"
    }
  ]
}
```

Propuesta de estructura de usuarios en json basada en la actividad:

```
{
  "usuarios": [
    {
      "id": 1,
      "nombre": "John Doe",
      "email": "jd1965@gmail.com",
      "fecha_registro": "2025-04-16T10:00:00"
    },
    {
      "id": 2,
      "nombre": "Jane Smith",
      "email": "janesmyt@gmail.com",
      "fecha_registro": "2025-04-16T11:00:00Z"
    },
    {
      "id": 3,
      "nombre": "Michael Jordan",
      "email": "mj23@outlook.com",
      "fecha_registro": "2025-04-15T17:00:00Z"
    }
  ]
}
```