

Proyecto “uso de librería en Python”: Flask

Alumnos: Alan Orejón, Gonzalo Farías.



Flask es una librería minimalista escrita en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código.

¿Por qué usar Flask?

- Incluye un servidor web de desarrollo.
- Tiene un depurador y soporte integrado para pruebas unitarias.
- Posee un buen manejo de rutas.
- Sirve para construir servicios web como por ejemplo: APIs REST.
- Es Open Source.
- Posee una documentación completa.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Con sólo 6 líneas de código podemos crear una aplicación

Proyecto: API de un sistema de notas de Alumnos

Realizamos una aplicación backend que responde a los protocolos HTTP básicos:

- GET (Obtener)
- POST (Insertar)
- DELETE (Eliminar)
- PUT (Actualizar)

En conclusión disponemos de distintos endpoints que nos permitirá manejar la base de datos de alumnos y sus notas y poder realizar consultas a esta, obteniendo así los datos ya listos para ser trabajados en un frontend.

Realizamos una correcta implementación de las rutas de nuestra aplicación y prevención de errores a través de códigos de estado de respuesta HTTP.

GET

Retorna el JSON de un alumno a partir de una ID especificada

```
@app.route('/alumnos/<int:id>', methods=['GET'])
def listar_alumno(id):
    for alumno in alumnos:
        if alumno['id']==id:
            return jsonify(alumno)
    return alumno_no_encontrado("El alumno no fue encontrado")
```



```
1  {
2    "apellido": "Peque",
3    "dni": 4512325,
4    "id": 3,
5    "nombre": "Roberto",
6    "notas": [
7      8,
8      6,
9      6
10   ]
11 }
```

Retorna los alumnos ordenados de manera ascendente por apellido.

```
@app.route('/alumnos/asc', methods=['GET'])
def obtener_alumnos_apellido_ascendente():
    ordenados = sorted(alumnos, key=lambda x: x['apellido'])
    return jsonify(ordenados)
```



```
1  {
2    "apellido": "Cabrera",
3    "dni": 4123355,
4    "id": 5,
5    "nombre": "Sejo",
6    "notas": [
7      1,
8      1,
9      3
10   ]
11 },
12 {
13   "apellido": "Gaston",
14   "dni": 547455,
15   "id": 6,
16   "nombre": "Gaston",
17   "notas": [
18     1,
19     1,
20     3
21   ]
22 }
```

POST

Crea un alumno en la base de datos y retorna el JSON del mismo.

```
@app.route('/alumnos', methods=['POST'])
def crear_alumno():
    global contador
    contador+=1
    try:
        if request.json['nombre']!="" and request.json['apellido']!="" and request.json['dni']!="":
            nuevo_alumno= {
                "id": contador,
                "nombre": request.json['nombre'],
                "notas": request.json['notas'],
                "dni": request.json['dni'],
                "apellido": request.json['apellido']
            }
            alumnos.append(nuevo_alumno)
            return request.json
        else:
            return error_al_crear_alumno("Faltan datos")
    except:
        return error_al_crear_alumno("Faltan atributos")
```

The screenshot shows a REST client interface with a POST request to `http://localhost:5000/alumnos`. The request body is a JSON object: `{ "nombre": "Olivia", "apellido": "Olivera", "notas": [10, 7, 5], "dni": 423432 }`. The response body is also a JSON object: `{ "apellido": "Olivera", "dni": 423432, "nombre": "Olivia", "notas": [10, 7, 5] }`.

POST `http://localhost:5000/alumnos`

Params Authorization Headers (9) Body Pre-request Script

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

```
1 {
2   "nombre": "Olivia",
3   "apellido": "Olivera",
4   "notas": [10, 7, 5],
5   "dni": 423432
6 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "apellido": "Olivera",
3   "dni": 423432,
4   "nombre": "Olivia",
5   "notas": [
6     10,
7     7,
8     5
9   ]
10 }
```

PUT

Actualiza las notas de un alumno a partir de su ID.

```
@app.route('/alumnos/<int:id>', methods=['PUT'])
def editar_nota(id):
    alumno = [alumno for alumno in alumnos if alumno['id'] == id]
    alumno[0]['notas'] = request.json['notas']
    return jsonify(alumno[0])
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:5000/alumnos/11
- Body:** The body tab is selected, showing a JSON object:

```
{  "notas": [10, 7, 10]}
```
- Response:** The response tab is selected, showing a JSON object:

```
{  "apellido": "Olivera",  "dni": 423432,  "id": 11,  "nombre": "Olivia",  "notas": [    10,    7,    10  ]}
```

DELETE

Elimina un alumno de la base de datos a partir de una ID y retorna el mismo

```
@app.route('/alumnos/<int:id>', methods=['DELETE'])
def quitar_alumno(id):
    for alumno in alumnos:
        if alumno['id']==id:
            alumnos.remove(alumno)
            return jsonify(alumno)
    return alumno_no_encontrado("El alumno no fue encontrado")
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize

```
1  [
2      "apellido": "Olivera",
3      "dni": 423432,
4      "id": 11,
5      "nombre": "Olivia",
6      "notas": [
7          10,
8          7,
9          10
10     ]
11 ]
```

promocionennos

