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 desarrrollo.
- 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.

```python
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

```python
@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")
```


```json
{
    "apellido": "Peque",
    "dni": 4512325,
    "id": 3,
    "nombre": "Roberto",
    "notas": [
        8,
        6,
        6
    ]
}
```


Retorna los alumnos ordenados de manera ascendente por apellido.

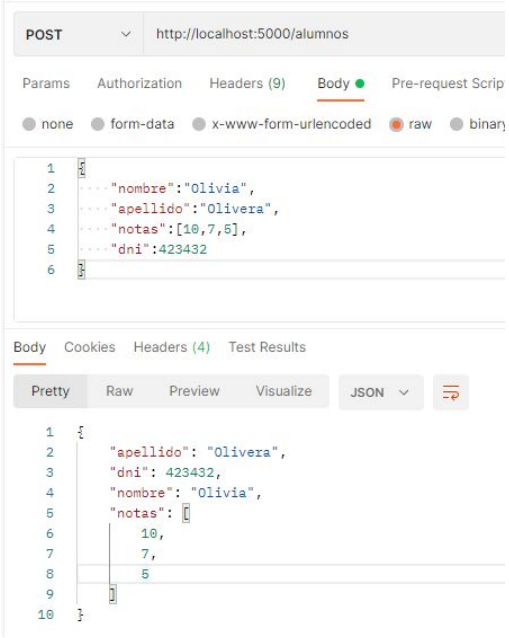
```python
@app.route('/alumnos/asc',methods=['GET'])
def obtener_alumnos_apellido_ascendente():
    ordenados = sorted(alumnos,key=lambda x: x['apellido'])
    return jsonify(ordenados)
```


```json
[
    {
        "apellido": "Cabrera",
        "dni": 4123355,
        "id": 5,
        "nombre": "Sejo",
        "notas": [
            1,
            1,
            3
        ]
    },
    {
        "apellido": "Gaston",
        "dni": 547455,
        "id": 6,
```

POST

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

```python
@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")
```



PUT

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

```python
@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])
```

DELETE

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

```python
@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")
```

promocionennos