Licenciatura en

Tecnologías de la Información

DevOps

Equipo: Daemontech

Integrantes:

- Joaquín Hernandez
- Mauricio Siri
- Rodolfo Alegre

- **Tutores: MATHÍAS RODRÍGUEZ AGRO**

Fecha de entrega: 13/09/24

Mini proyecto en Python

UTEC
Universidad Tecnológica

Paso 1: Instalación de dependencias

Primero, instalamos Flask en nuestro entorno de desarrollo. Si no tenemos Flask instalado, ejecutamos el siguiente comando en tu terminal:

```
pip install flask
```

Paso 2: Estructura del proyecto

Creamos una estructura de directorios para el proyecto:

```
pokemon_api/
│
├── app.py  # Archivo principal de la aplicación
├── README.md  # Archivo de instrucciones para el usuario
└── requirements.txt  # Dependencias del proyecto
```

Paso 3: Código de la API en app.py

```python
from flask import Flask, jsonify, request

app = Flask(__name__)

# Simulando una base de datos en memoria
pokemons = []
current_id = 1

# Obtener todos los Pokémon
@app.route('/pokemons', methods=['GET'])
def get_pokemons():
    return jsonify(pokemons), 200

# Crear un nuevo Pokémon
@app.route('/pokemons', methods=['POST'])
def create_pokemon():
    global current_id
    data = request.get_json()

    # Crear un nuevo Pokémon
    pokemon = {
        'id': current_id,
        'nombre': data['nombre'],
        'imagen': data['imagen'],
        'caracteristicas': {
            'peso': data['caracteristicas']['peso'],
            'altura': data['caracteristicas']['altura'],
            'fuerza': data['caracteristicas']['fuerza'],
            'edad': data['caracteristicas']['edad']
```

```
        },
        'habilidades': data['habilidades'],
        'tipo': data['tipo'],
        'habitat': data['habitat']
    }
    pokemons.append(pokemon)
    current_id += 1
    return jsonify(pokemon), 201

# Obtener un Pokémon específico por ID
@app.route('/pokemons/<int:id>', methods=['GET'])
def get_pokemon(id):
    pokemon = next((p for p in pokemons if p['id'] == id), None)
    if pokemon is None:
        return jsonify({'message': 'Pokémon no encontrado'}), 404
    return jsonify(pokemon), 200

# Actualizar la información de un Pokémon por ID
@app.route('/pokemons/<int:id>', methods=['PUT'])
def update_pokemon(id):
    data = request.get_json()
    pokemon = next((p for p in pokemons if p['id'] == id), None)
    if pokemon is None:
        return jsonify({'message': 'Pokémon no encontrado'}), 404

    # Actualizar los campos
    pokemon['nombre'] = data.get('nombre', pokemon['nombre'])
    pokemon['imagen'] = data.get('imagen', pokemon['imagen'])
    pokemon['caracteristicas']['peso'] = data['caracteristicas'].get('peso',
pokemon['caracteristicas']['peso'])
    pokemon['caracteristicas']['altura'] = data['caracteristicas'].get('altura',
pokemon['caracteristicas']['altura'])
    pokemon['caracteristicas']['fuerza'] = data['caracteristicas'].get('fuerza',
pokemon['caracteristicas']['fuerza'])
    pokemon['caracteristicas']['edad'] = data['caracteristicas'].get('edad',
pokemon['caracteristicas']['edad'])
    pokemon['habilidades'] = data.get('habilidades', pokemon['habilidades'])
    pokemon['tipo'] = data.get('tipo', pokemon['tipo'])
    pokemon['habitat'] = data.get('habitat', pokemon['habitat'])

    return jsonify(pokemon), 200

# Eliminar un Pokémon por ID
@app.route('/pokemons/<int:id>', methods=['DELETE'])
def delete_pokemon(id):
    global pokemons
    pokemons = [p for p in pokemons if p['id'] != id]
    return jsonify({'message': 'Pokémon eliminado'}), 200

if __name__ == '__main__':
    app.run(debug=True)
```

Paso 4:

Todas las solicitudes estarán disponibles en `http://127.0.0.1:5000`, que es el **localhost** donde se ejecuta la API.

Ejemplos de URLs para cada operación:

1)Obtener todos los Pokémon (GET):

```
GET http://127.0.0.1:5000/pokemons
```

Luego agregamos datos a

2.- Obtener un Pokémon por ID (GET):
Supongamos que el ID del Pokémon es 1:

```
GET http://127.0.0.1:5000/pokemons/1
```

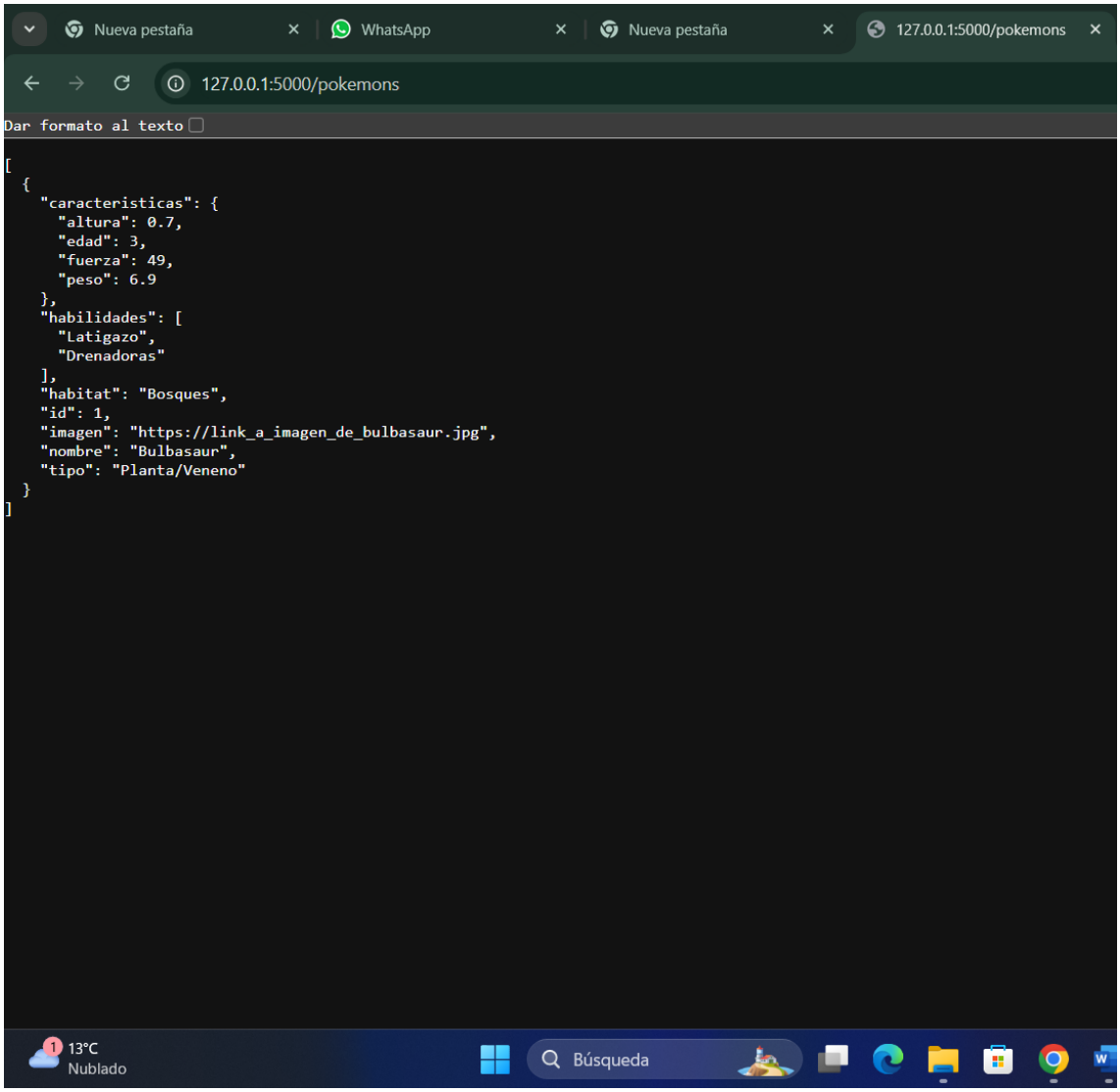
3.-Crear un nuevo Pokémon (POST):

- ***Se debe enviar la información del nuevo Pokémon en el cuerpo de la solicitud en formato JSON.***
- JSON.

POST http://127.0.0.1:5000/pokemons

Ejemplo de cuerpo (JSON):

```json
{
  "nombre": "Bulbasaur",
  "imagen": "https://link_a_imagen_de_bulbasaur.jpg",
  "caracteristicas": {
    "peso": 6.9,
    "altura": 0.7,
    "fuerza": 49,
    "edad": 3
  },
  "habilidades": ["Latigazo", "Drenadoras"],
  "tipo": "Planta/Veneno",
  "habitat": "Bosques"
}
```



4 actualizar un Pokémon por ID (PUT):

- Supongamos que queremos actualizar el Pokémon con ID 1:

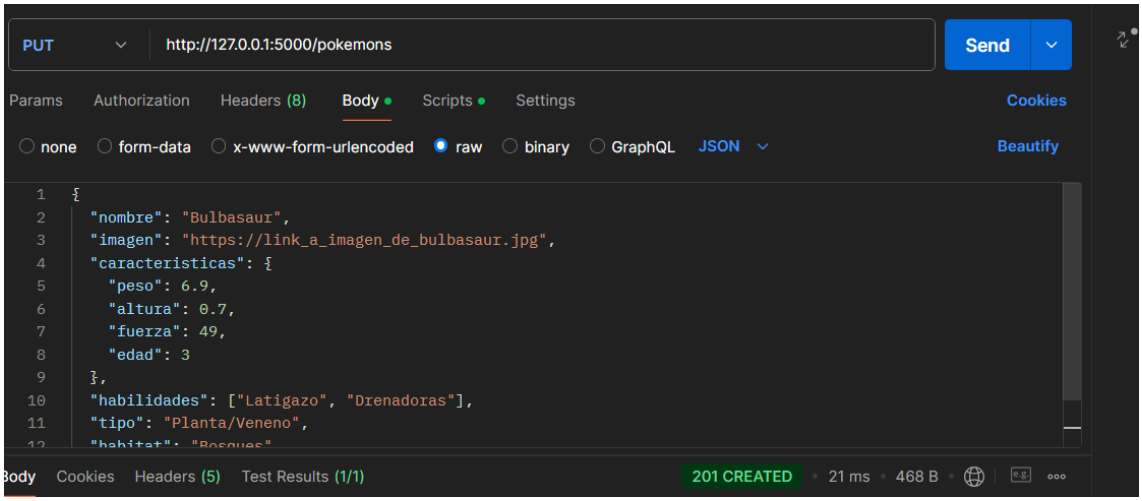
```
PUT http://127.0.0.1:5000/pokemons/1
```

Headers

- header con la clave Content-Type y el valor application/json para indicar que el cuerpo del request está en formato JSON.

Body

- Seleccionamos la opción raw en el cuerpo (Body) y el tipo JSON.
- Luego, pegamos los datos JSON con la información de Bulbasaur:

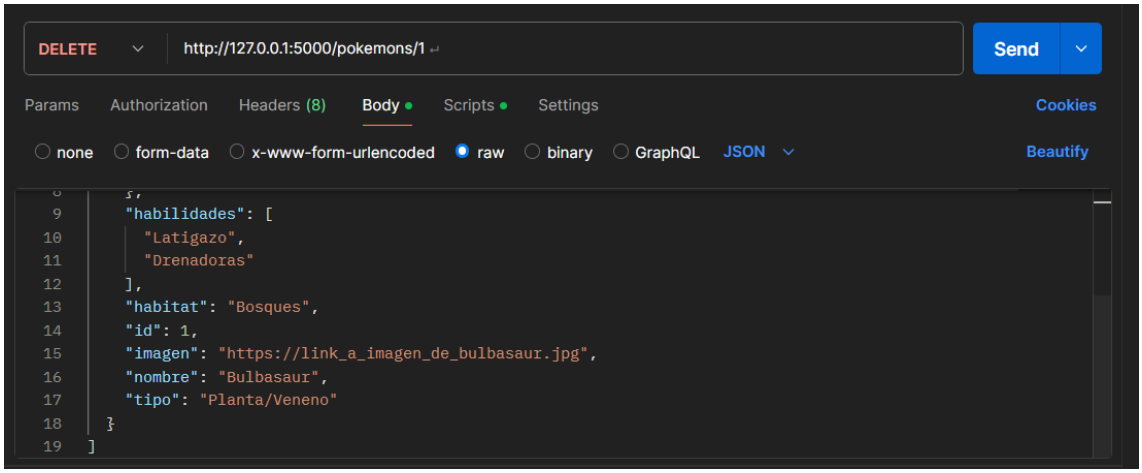


Aquí captura de la salida por terminal exitosa

5. La URL para la solicitud DELETE debe incluir el id del Pokémon que deseamos eliminar. Por ejemplo, si el Pokémon que deseas eliminar tiene el id 1:

url seria:

http://127.0.0.1:5000/pokemons/1



Si la solicitud es exitosa, deberías obtener una respuesta similar a esta:

```
{
  "mensaje": "Pokémon eliminado"
}
```