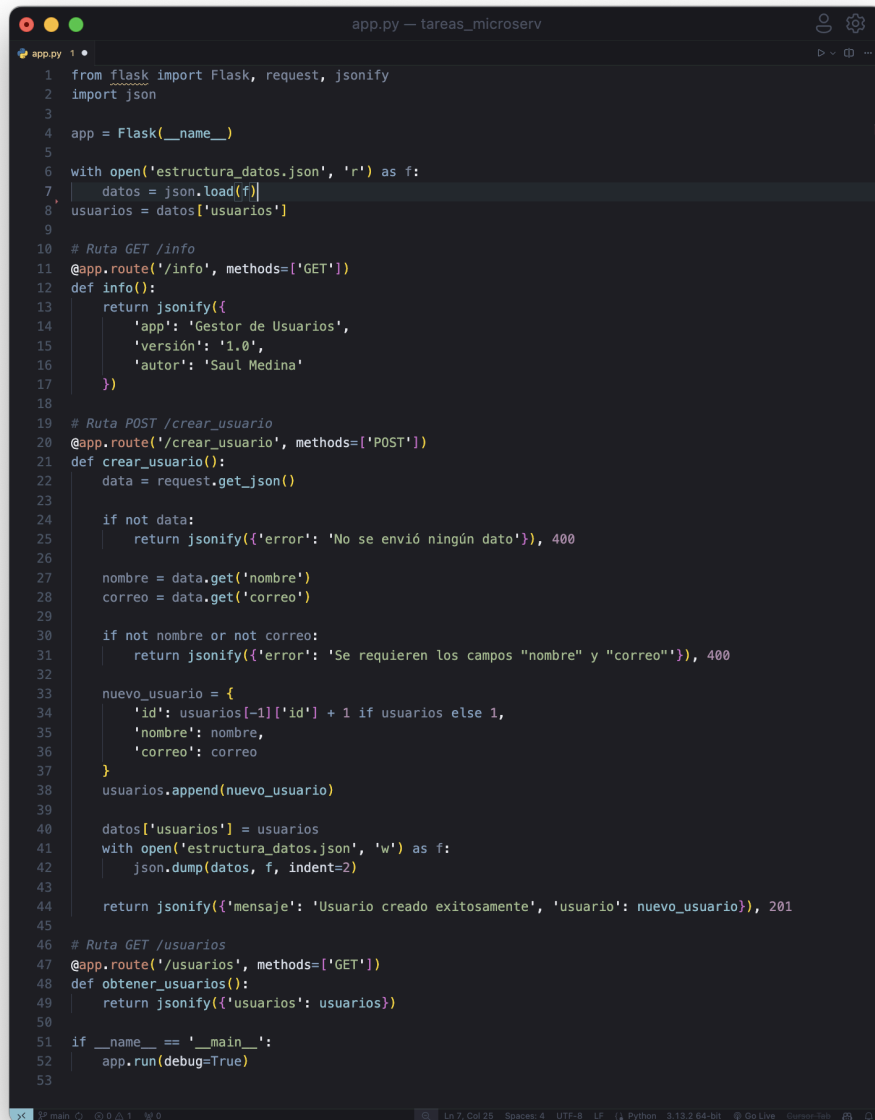


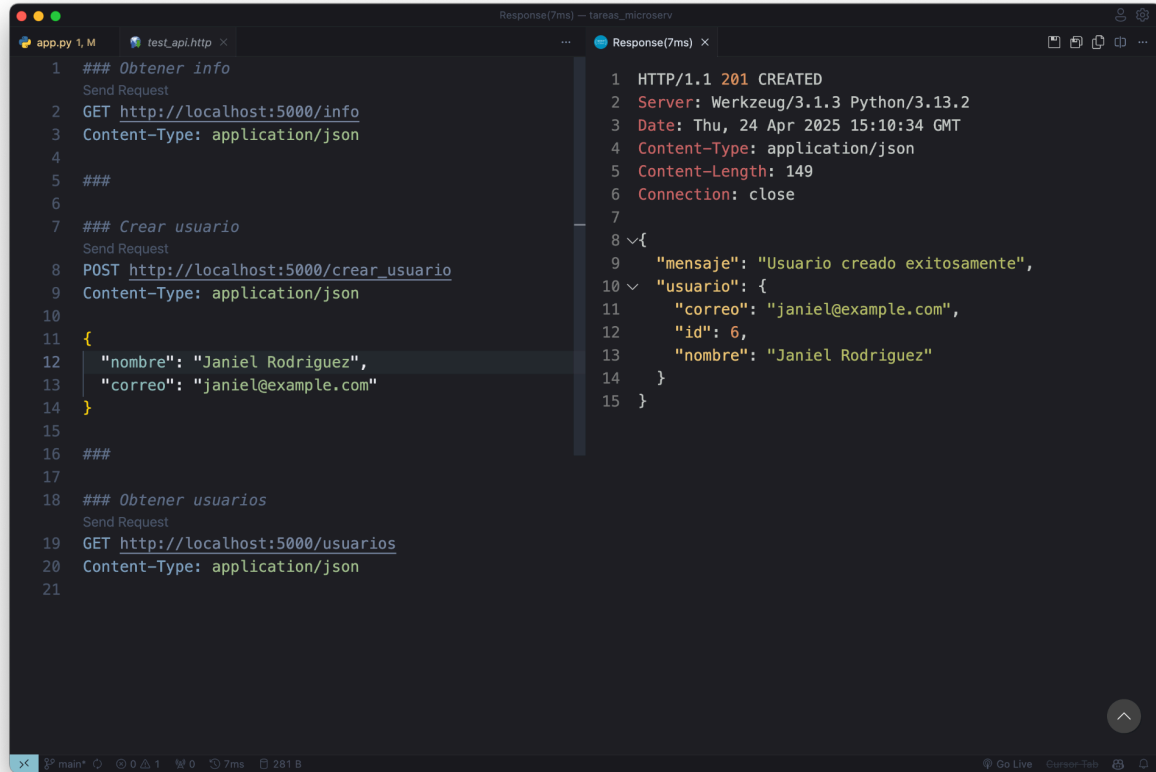
API

A screenshot of a code editor window titled 'app.py - tareas_microserv'. The code is written in Python and defines a Flask application. It imports Flask, request, jsonify, and json. It initializes the app and loads data from 'estructura_datos.json'. There are three main routes: a GET route for '/info' that returns app metadata, a POST route for '/crear_usuario' that creates a new user and returns a success message, and a GET route for '/usuarios' that returns the list of all users. The code also includes a main block to run the app in debug mode.

```
1 from flask import Flask, request, jsonify
2 import json
3
4 app = Flask(__name__)
5
6 with open('estructura_datos.json', 'r') as f:
7     datos = json.load(f)
8     usuarios = datos['usuarios']
9
10 # Ruta GET /info
11 @app.route('/info', methods=['GET'])
12 def info():
13     return jsonify({
14         'app': 'Gestor de Usuarios',
15         'versión': '1.0',
16         'autor': 'Saul Medina'
17     })
18
19 # Ruta POST /crear_usuario
20 @app.route('/crear_usuario', methods=['POST'])
21 def crear_usuario():
22     data = request.get_json()
23
24     if not data:
25         return jsonify({'error': 'No se envió ningún dato'}), 400
26
27     nombre = data.get('nombre')
28     correo = data.get('correo')
29
30     if not nombre or not correo:
31         return jsonify({'error': 'Se requieren los campos "nombre" y "correo"'}), 400
32
33     nuevo_usuario = {
34         'id': usuarios[-1]['id'] + 1 if usuarios else 1,
35         'nombre': nombre,
36         'correo': correo
37     }
38     usuarios.append(nuevo_usuario)
39
40     datos['usuarios'] = usuarios
41     with open('estructura_datos.json', 'w') as f:
42         json.dump(datos, f, indent=2)
43
44     return jsonify({'mensaje': 'Usuario creado exitosamente', 'usuario': nuevo_usuario}), 201
45
46 # Ruta GET /usuarios
47 @app.route('/usuarios', methods=['GET'])
48 def obtener_usuarios():
49     return jsonify({'usuarios': usuarios})
50
51 if __name__ == '__main__':
52     app.run(debug=True)
53
```

Explicación: En este flask api tenemos varios endpoints. Tenemos un endpoint que pide información para crear un usuario. Este endpoint requiere informacion en formato json y devuelve información en formato json. Cuando se crea lo guarda en un array y devuelve mensaje. El otro endpoint devuelve la lista de los usuarios registrados en el array.

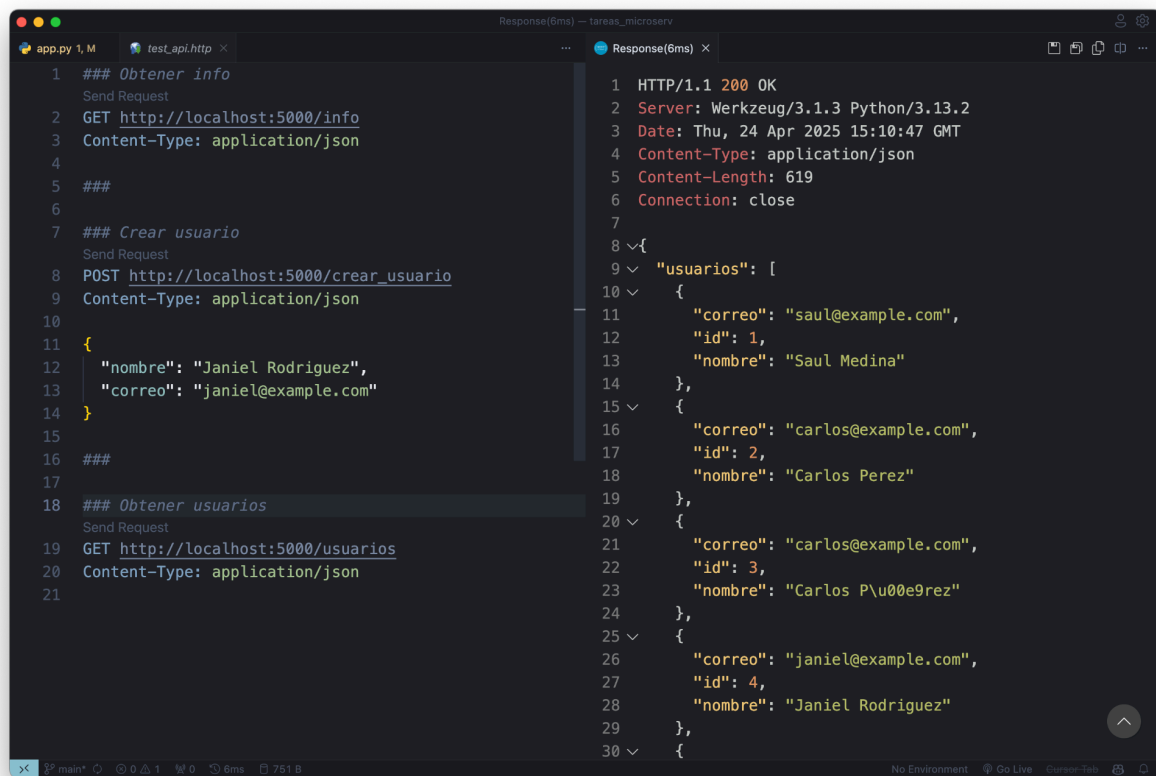
Test



```
1  ### Obtener info
2  Send Request
3  GET http://localhost:5000/info
4  Content-Type: application/json
5
6  ###
7  ### Crear usuario
8  Send Request
9  POST http://localhost:5000/crear_usuario
10 Content-Type: application/json
11 {
12   "nombre": "Janiel Rodriguez",
13   "correo": "janiel@example.com"
14 }
15
16 ###
17
18 ### Obtener usuarios
19 Send Request
20 GET http://localhost:5000/usuarios
21 Content-Type: application/json
```

```
1  HTTP/1.1 201 CREATED
2  Server: Werkzeug/3.1.3 Python/3.13.2
3  Date: Thu, 24 Apr 2025 15:10:34 GMT
4  Content-Type: application/json
5  Content-Length: 149
6  Connection: close
7
8  {
9    "mensaje": "Usuario creado exitosamente",
10   "usuario": {
11     "correo": "janiel@example.com",
12     "id": 6,
13     "nombre": "Janiel Rodriguez"
14   }
15 }
```

Explicación: Aquí tenemos un ejemplo del usuario enviando información al api para crear el usuario. El api recibe la información, crea el usuario y devuelve la información del usuario creado.



```
1  ### Obtener info
2  Send Request
3  GET http://localhost:5000/info
4  Content-Type: application/json
5
6  ###
7  ### Crear usuario
8  Send Request
9  POST http://localhost:5000/crear_usuario
10 Content-Type: application/json
11 {
12   "nombre": "Janiel Rodriguez",
13   "correo": "janiel@example.com"
14 }
15
16 ###
17
18 ### Obtener usuarios
19 Send Request
20 GET http://localhost:5000/usuarios
21 Content-Type: application/json
```

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Thu, 24 Apr 2025 15:10:47 GMT
4 Content-Type: application/json
5 Content-Length: 619
6 Connection: close
7
8 {
9   "usuarios": [
10    {
11      "correo": "saul@example.com",
12      "id": 1,
13      "nombre": "Saul Medina"
14    },
15    {
16      "correo": "carlos@example.com",
17      "id": 2,
18      "nombre": "Carlos Perez"
19    },
20    {
21      "correo": "carlos@example.com",
22      "id": 3,
23      "nombre": "Carlos P\u00e9rez"
24    },
25    {
26      "correo": "janiel@example.com",
27      "id": 4,
28      "nombre": "Janiel Rodriguez"
29    },
30    {
```

Explicación: Cuando ejecutamos este endpoint el api nos da una lista de todos los usuarios registrados en el array en formato json. Podemos ver que luego de añadir un usuario se refleja en la lista si la volvemos a solicitar.