

Integración APIs

Jorge Cañete Martín

Primer paso

Instalamos Node.js para poder lanzar servidores locales a partir de ficheros JSON para la API A (puerto 3000), la API B (puerto 3001) y el token (puerto 3002). Tendremos 3 ficheros JSON el cual contendrán los dos contenidos dados por el ejercicio y por último el token.

Inicialmente mostraremos el ejemplo con localhost, para cambiar el nombre de dominio a <https://api.sistemaA.com/facturas> y <https://api.sistemaB.com/bills> instalaríamos **nginx** para poder realizar proxys. Posteriormente en el archivo de configuración de nginx añadiríamos un proxy del puerto 3000 a <https://api.sistemaA.com/facturas> y del puerto 3001 a <https://api.sistemaB.com/bills>.

Segundo paso

Creación de un script llamado Servers.py el cual lanzará 3 cmd simultáneamente en el que cada uno estará lanzando un servidor localhost para poder tenerlos activos para poder ejecutar métodos HTTP REST sobre ellos. Aquí podemos ver una parte de este código:

```
Servers.py X
API > Servers.py > ...
32
33 if __name__ == "__main__":
34     if not verificar_node():
35         print("No se puede iniciar json-server porque Node.js no está disponible.")
36     else:
37         # Rutas absolutas al ejecutable json-server
38         json_server_path = 'C:\\Users\\Jorge\\AppData\\Roaming\\npm\\json-server.cmd'
39
40         # Comandos para iniciar json-server con diferentes configuraciones
41         command1 = [json_server_path, '--watch', 'RespuestaA.json']
42         command2 = [json_server_path, '--watch', 'RespuestaB.json', '--port', '3001']
43         command3 = [json_server_path, '--watch', 'token.json', '--port', '3002']
44
45         # Directorio de trabajo donde están los archivos JSON
46         cwd = 'C:\\Users\\Jorge\\Desktop\\a'
47
48         # Iniciar los procesos de json-server
49         process1 = iniciar_json_server(command1, cwd=cwd)
50         process2 = iniciar_json_server(command2, cwd=cwd)
51         process3 = iniciar_json_server(command3, cwd=cwd)
```

Cada uno de los servidores sera lanzado con los comandos:

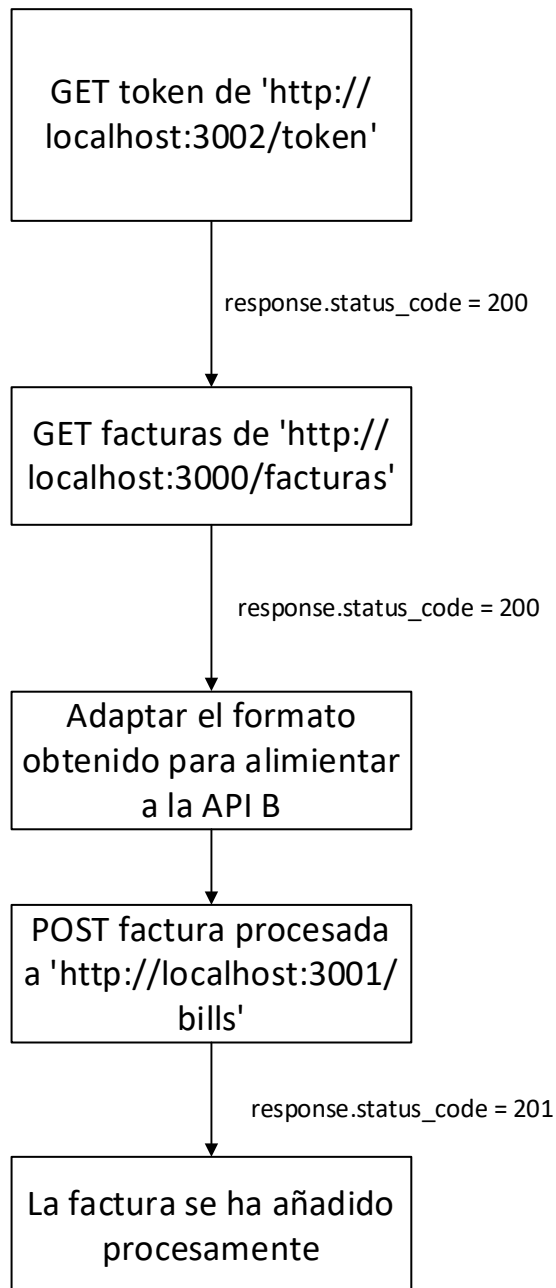
- `json-server --watch RespuestaA.json`
- `json-server --watch RespuestaB.json --port 3001`
- `json-server --watch Token.json --port 3002`

Aquí vemos como se ven los 3 servidores locales. El token generado es aleatorio para posteriormente ser autorizado siguiendo los principios de OAuth 2.0.

localhost:3000/facturas	localhost:3001/bills	localhost:3002/token
<pre>{ "id": "123", "cliente": "Empresa XYZ", "monto": 1500.75, "fecha_emision": "2023-05-01", "estado": "pagada" }</pre>	<pre>{ "invoice_id": "456", "customer": "Company ABC", "amount_due": 2000.5, "date_issued": "2023-05-02", "status": "unpaid", "id": "0a3f" }</pre>	<pre>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwiaWF0IjoiSf1KxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c</pre>

Tercer paso

Ejecutar el script Token.py el cual realizará el flujo de información indicado por el ejercicio. El status code 200 significa que la información se ha obtenido correctamente y el 201 que se ha posteado correctamente.



```
client_id = 'tu_client_id'
client_secret = 'tu_client_secret'
token_url = 'http://localhost:3002/token'

def obtener_token():
    try:
        # Solicitar el token desde el endpoint simulado
        response = requests.get(token_url, timeout=30)

        # Verificar si la solicitud fue exitosa
        response.raise_for_status()

        # Obtener el token directamente de la respuesta
        token = response.text.strip()

        if not token:
            raise ValueError('No se recibió el token')

        print(f'Token obtenido: {token}')
        return token

    except requests.exceptions.RequestException as e:
        print(f'Error en la solicitud: {e}')
    except ValueError as e:
        print(f'Error en la respuesta del token: {e}')
```

```
# Parámetros de consulta
fecha_inicio = '2023-05-01'
fecha_fin = '2023-05-31'
url_api_a = 'http://localhost:3000/facturas'

# Solicitar las facturas de API A
response = requests.get(url_api_a, headers={
    'Authorization': f'Bearer {token}'
}, params={
    'fecha_inicio': fecha_inicio,
    'fecha_fin': fecha_fin
}, timeout=30)
```

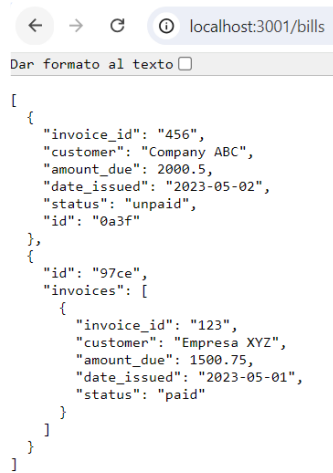
```
# Clave API de autenticación para API B
api_key = 'tu_api_key'
url_api_b = 'http://localhost:3001/bills'

# Enviar las facturas transformadas a API B
response = requests.post(url_api_b, headers={
    'x-api-key': api_key,
    'Content-Type': 'application/json'
}, json={
    'invoices': facturas_transformadas
}, timeout=30)

print(f'Respuesta de la API B: {response.text}')

if response.status_code == 201:
    print('Facturas enviadas exitosamente')
```

El resultado que queda tras haber lanzado Server.py y Token.py es la API B situada en 'http://localhost:3001/bills' ha pasado de tener el recibo unpaid que tenía inicialmente a tener también el recibo paid añadido actualmente.



```
[
  {
    "invoice_id": "456",
    "customer": "Company ABC",
    "amount_due": 2000.5,
    "date_issued": "2023-05-02",
    "status": "unpaid",
    "id": "0a3f"
  },
  {
    "id": "97ce",
    "invoices": [
      {
        "invoice_id": "123",
        "customer": "Empresa XYZ",
        "amount_due": 1500.75,
        "date_issued": "2023-05-01",
        "status": "paid"
      }
    ]
  }
]
```