UD10. Servicios web y aplicaciones híbridas

DESARROLLO WEB EN ENTORNO SERVIDOR

Funcionalidades adicionales

Para aumentar la funcionalidad de nuestra aplicación, podemos:

A. Usar alguna librería/componente que ha hecho alguien.
   -
   -
   -
   -

B. Acceder a aplicaciones de terceros.
   -
   -
   -

Funcionalidades adicionales

Para aumentar la funcionalidad de nuestra aplicación, podemos:

A. Usar alguna librería/componente que ha hecho alguien.
   - Se ejecutan en local.
   - Tenemos acceso a todo el código.
   - Pueden ser de pago o gratuitas.
   - Capacidades limitadas.

B. Acceder a aplicaciones de terceros.
   -
   -
   -

Funcionalidades adicionales

Para aumentar la funcionalidad de nuestra aplicación, podemos:

A. Usar alguna librería/componente que ha hecho alguien.
   - Se ejecutan en local.
   - Tenemos acceso a todo el código.
   - Pueden ser de pago o gratuitas.
   - Capacidades limitadas.

B. Acceder a aplicaciones de terceros.
   - Utilizando tecnologías web (HTML, HTTP, JSON,…)
   - Pueden ser de pago o gratuitas.
   - Sin limitaciones.

Funcionalidades adicionales

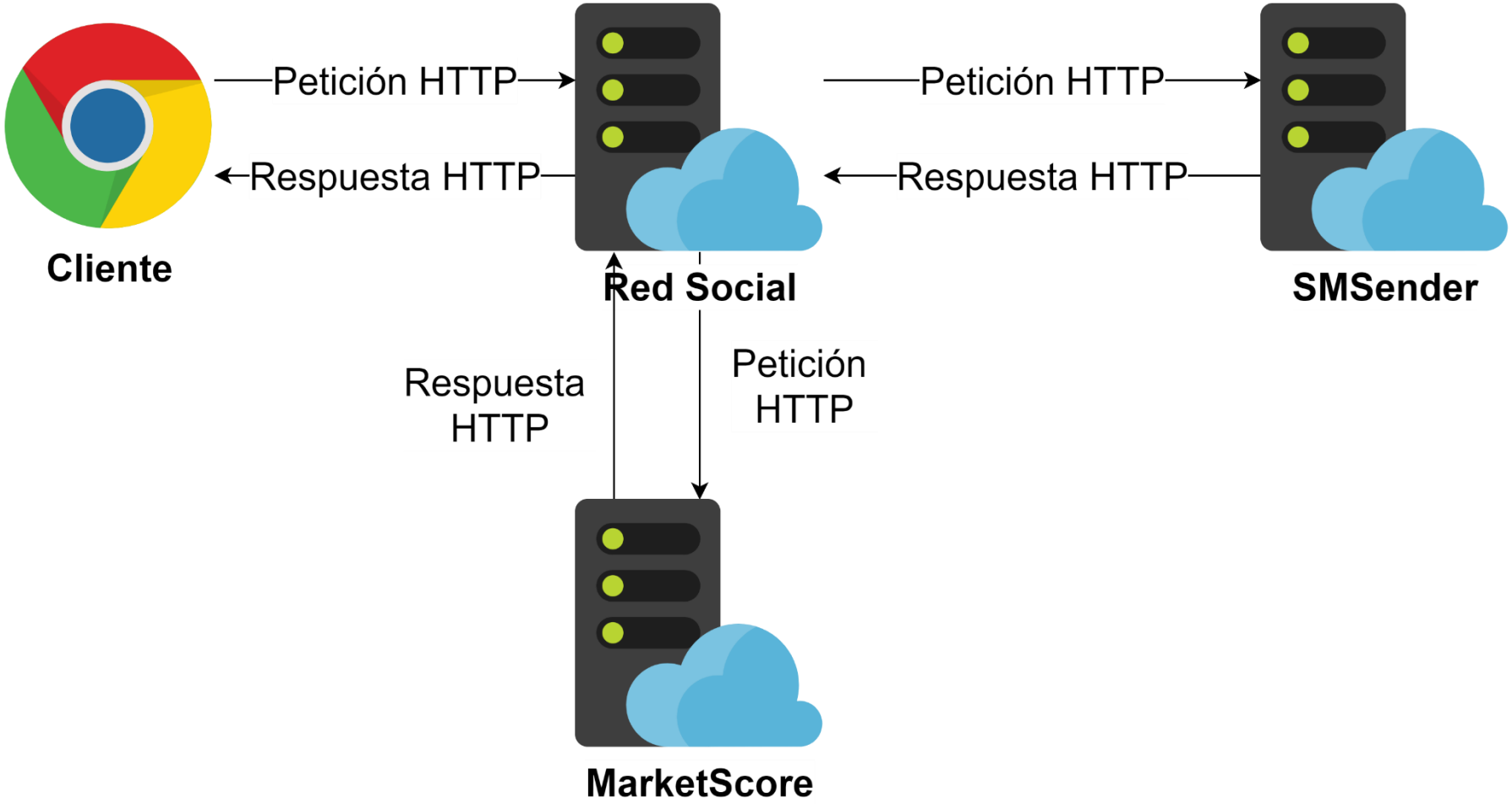
Para aumentar la funcionalidad de nuestra aplicación, podemos:

A. Usar alguna librería/componente que ha hecho alguien.
   - Se ejecutan en local.
   - Tenemos acceso a todo el código.
   - Pueden ser de pago o gratuitas.
   - Capacidades limitadas.

B. Acceder a aplicaciones de terceros.
   - Utilizando tecnologías web (HTML, HTTP, JSON,…)
   - Pueden ser de pago o gratuitas.
   - Sin limitaciones.

Servicio web

Servicio web

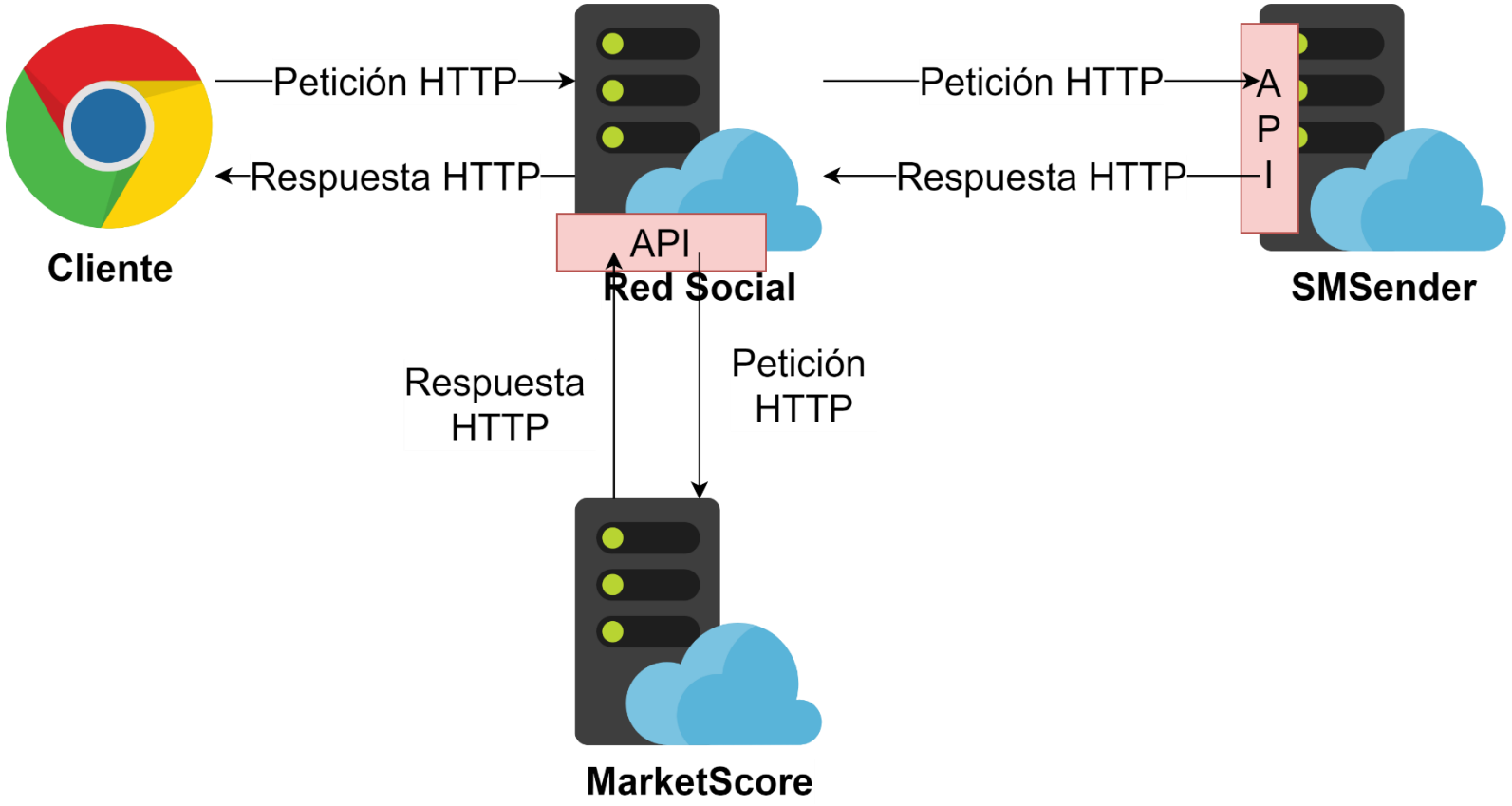


Servicio web - API

Para que una aplicación ofrezca un servicio web a otras aplicaciones:

- Debe definir qué peticiones atiende.
- …y qué va a responder.

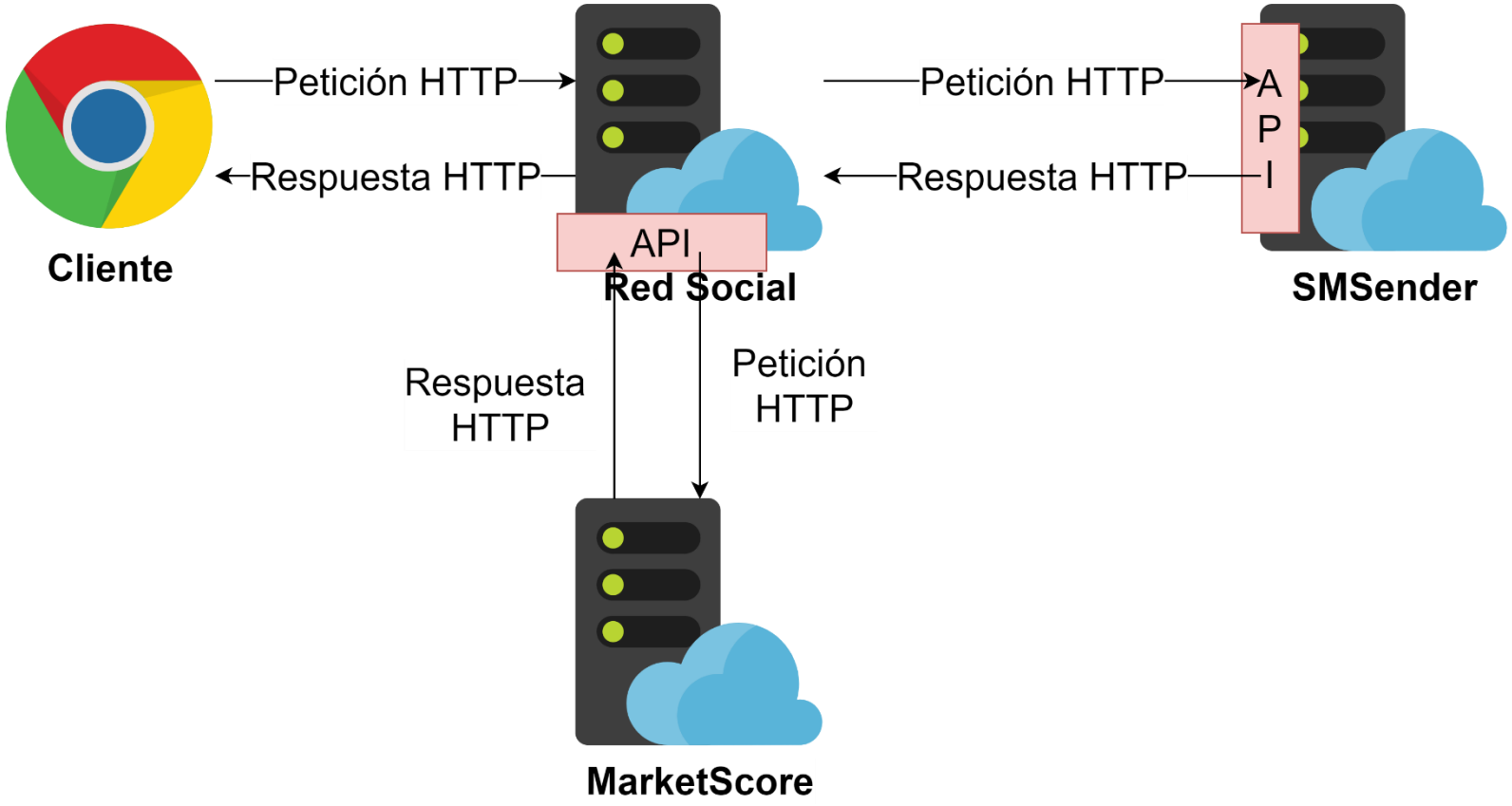
Si hacemos esto, estaremos creando una API (*Application Programming Interface)*



Servicio web - consumir VS ofrecer API

Lo normal es que una aplicación haga uso de algún servicio web/consuma una API.

...pocas veces ofrece un servicio web/ofrece una API.

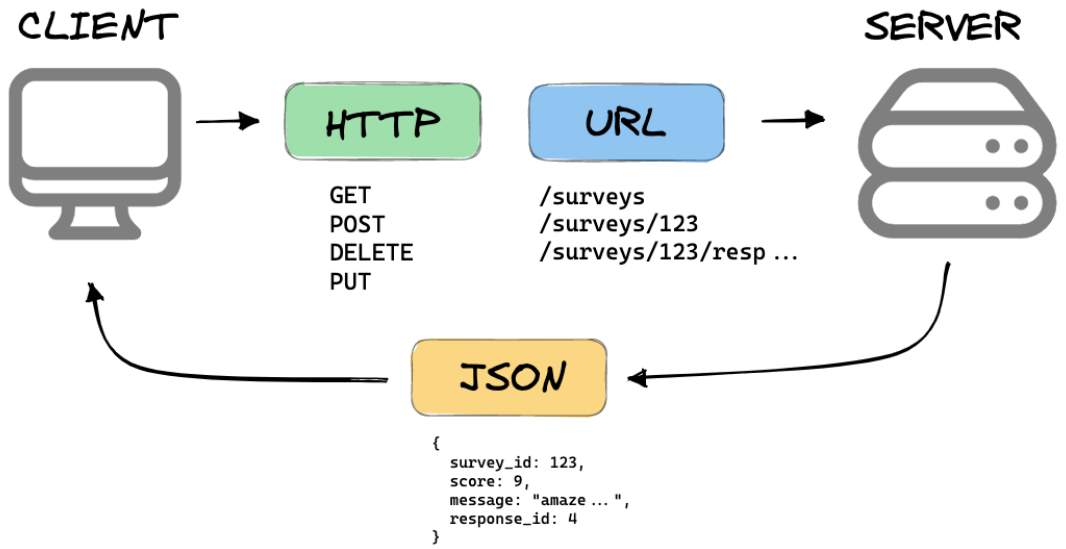


Cómo funciona una API

API REST

En un mundo ideal crearíamos una API REST *(REpresentational State Transfer):*

- La URL representa un recurso.
- Se hace una petición al recurso con un método HTTP.
- El método determina una acción.



OJO: no es ningún estándar oficial.

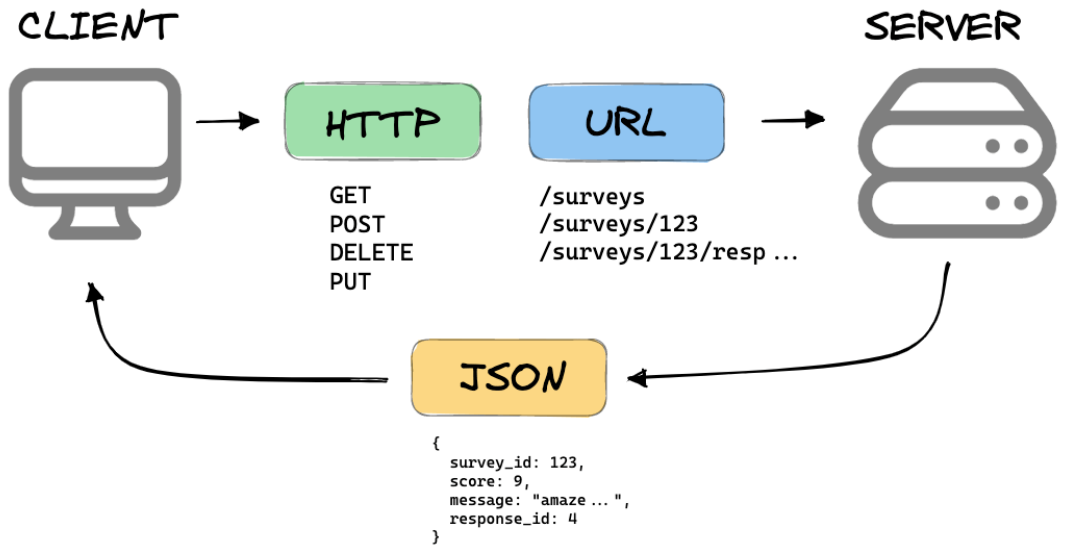
API REST

Ejemplo:

Método	Uso
GET /objetct/id	Obtiene el recurso
POST /object	Crea el recurso
PUT /object/id	Modifica el recurso
DELETE /object/id	Elimina el recurso

API REST: principios

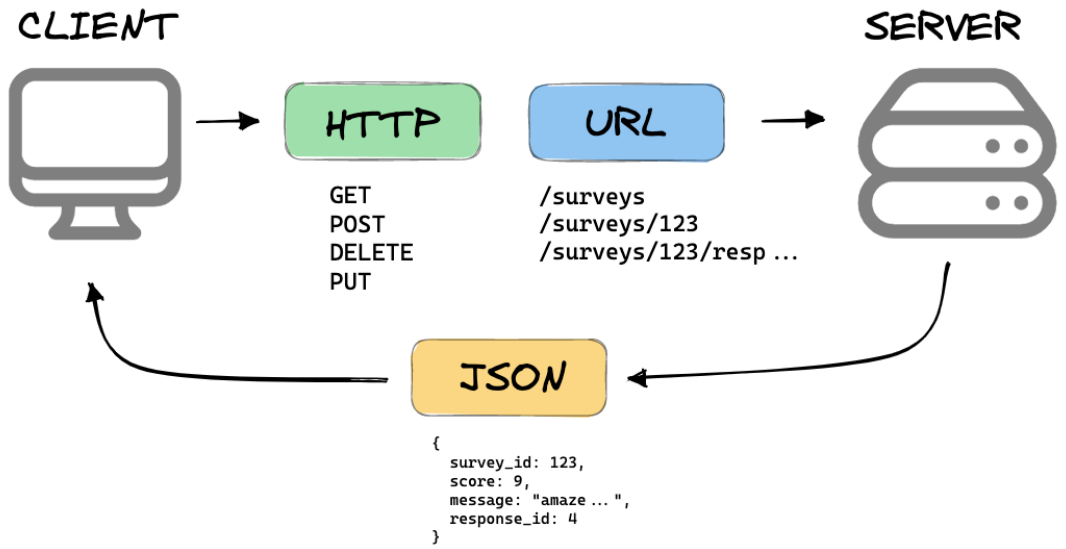
1. Hay que identificar los recursos de la aplicación.
2. Crear una URL para cada recurso, que sea clara y sin ambigüedades.
3. Identificar las operaciones que sean pertinentes:
   - Crear (POST).
   - Leer (GET).
   - Actualizar (PUT).
   - Borrar (DELETE).
4. Proteger la API, si procede.



API REST: en la realidad

Como no es un estándar…

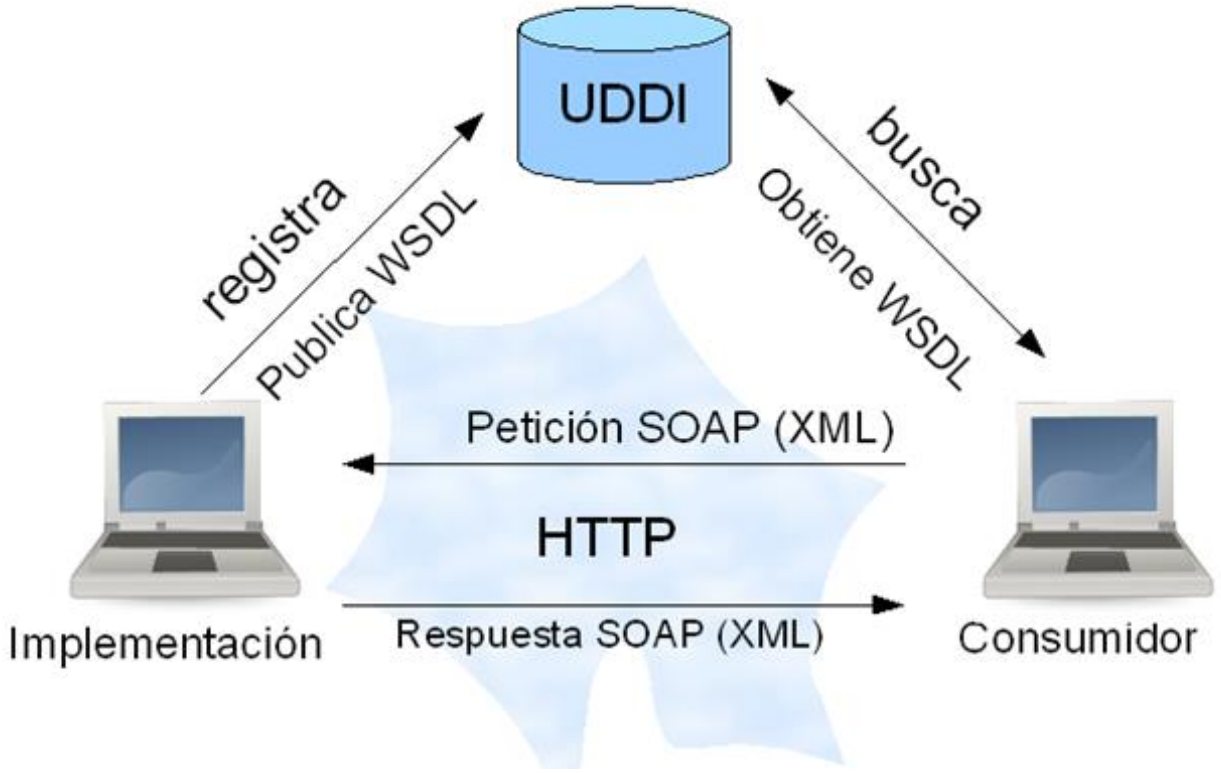
- A veces resulta un modelo pesado. En la práctica → no se sigue completamente.
- No hay obligación de devolver nada en concreto. En la práctica → JSON.



Alternativa

Arquitectura de *web services* del W3C:

- Lenguajes.
- Arquitectura.
- Protocolo.



Como trabajar con una API

Depurar API

El navegador no está preparado para probar APIs.

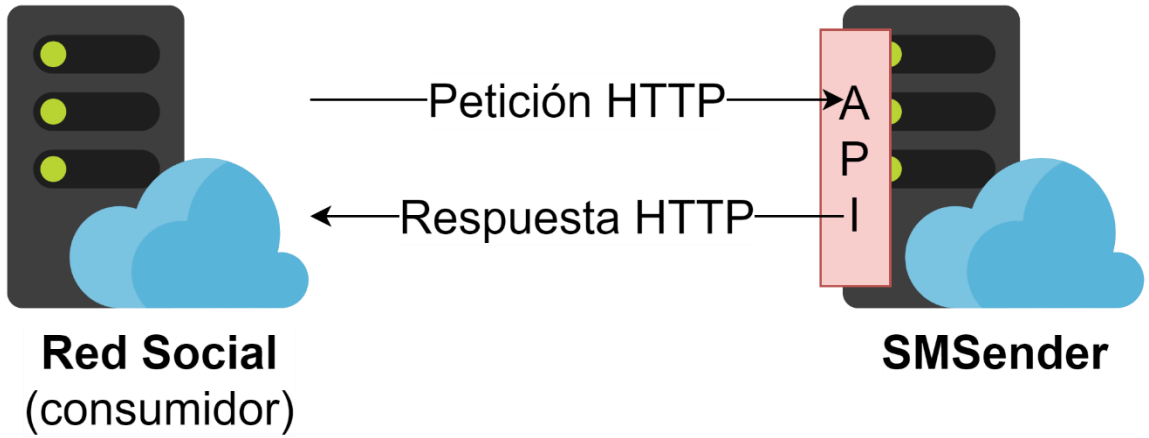
Necesitamos una herramienta para lanzar peticiones HTTP libremente.

Ejemplos:

Consumir una API

Implica hacer peticiones desde el servidor a otro servidor y analizar la respuesta.

- Ya conoces una manera: `file_get_contents` (solo hace peticiones GET).
- La manera de verdad: librería cURL (permite todo).



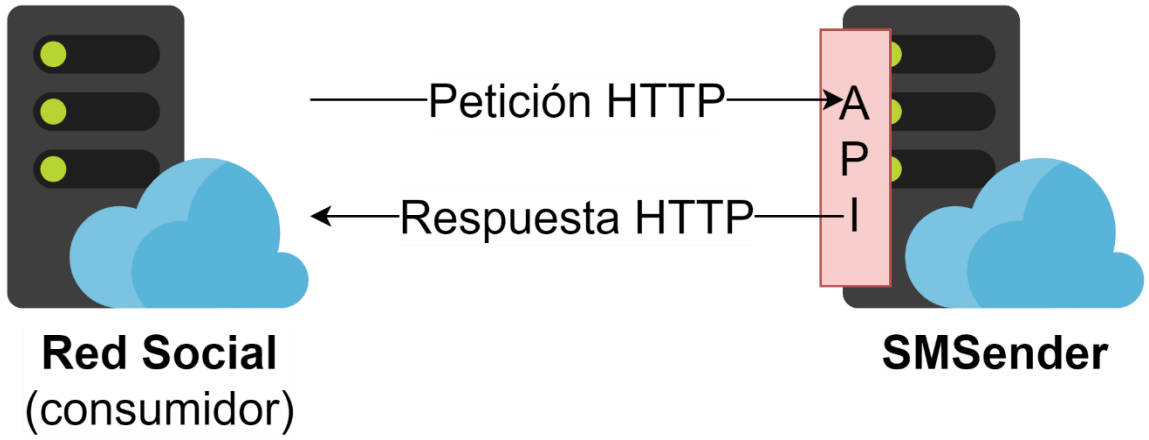
Consumir una API

Implica hacer peticiones desde el servidor a otro servidor y analizar la respuesta.

- Ya conoces una manera: `file_get_contents` (solo hace peticiones GET).
- La manera de verdad: librería cURL (permite todo).

…pero primero tienes que saber cómo hacer la petición leyendo la petición de la API.

- Ejemplo: https://publicapis.dev/category/animals



Ofrecer una API

Idealmente puedes seguir el esquema REST visto.

Siendo pragmáticos, haz algo coherente y sencillo (*worse is better*):

1. Cada controlador hace una operación (y solo una).
2. No hay plantillas.
3. Se devuelve JSON como resultado de hacer `json_encode`.
4. Piensa en si hace falta autenticar a los usuarios.