APIS Y SERVICIOS

MODELO CLIENTE SERVIDOR

Cliente-Servidor es uno de los estilos arquitectónicos distribuidos más conocidos, el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumidos por el Cliente.

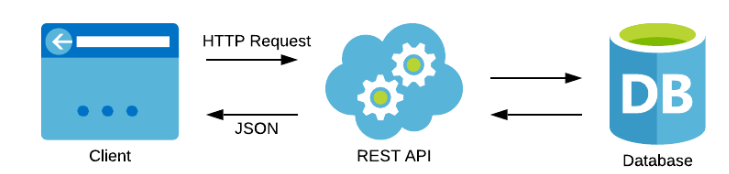
En una arquitectura Cliente-Servidor existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado.

API

Application Program Interface o simplemente API es un contrato entre una porción de software y otra, donde la información se comparte de manera estructurada a través de request (petición) y response (respuesta). Una API es un mensajero, toma un request de un software para llevarlo a otro y regresa un response.

REST API

REST significa Representational State Transfer. Este es un estilo de arquitectura para diseñar aplicaciones en red que se basan en un protocolo cliente-servidor, usualmente es HTTP. Esto permite a una aplicación web realizar una petición HTTP a un servidor y obtener un response en cierto formato que en la mayoría de los casos es un JSON, una representación de un objeto de JavaScript.



La mayoría de los lenguajes de programación pueden realizar peticiones HTTP, por lo que se puede implementar no solo con JavaScript.

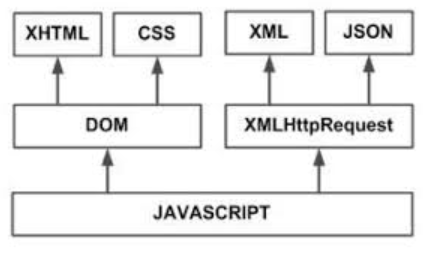
AJAX

JavaScript Asíncrono + XML (AJAX) nos permite crear aplicaciones web más rápidas y con mejor respuesta al usuario. Con AJAX podemos actualizar continuamente una parte de una página web sin necesidad de volver a cargar la página completa.

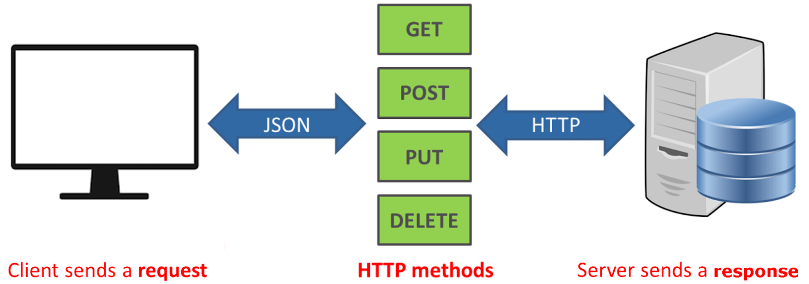
Se trata de un nuevo modo para utilizar varias tecnologías en conjunto: HTML/XHTML, CSS, DOM, XML, JavaScript, XSLT y XMLHttpRequest.



FORMATOS DE INFORMACION



METODOS HTTP



Cuando realizamos una petición HTTP debemos indicar el método, este define el tipo de acción que deseamos realizar para un recurso determinado. Los métodos más usados:

GET: obtener datos

POST: enviar información

PUT: actualizar datos

DELETE: elimina un recurso



Otros métodos menos usados:

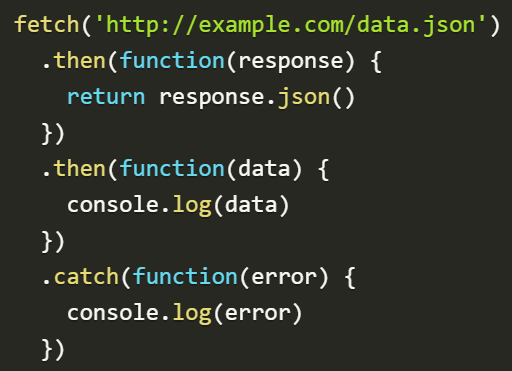
HEAD: Es similar al método GET, la diferencia es que HEAD no regresa el body del request

OPTIONS: obtener todos los métodos que un recurso soporta

PATCH: realizar modificaciones parciales

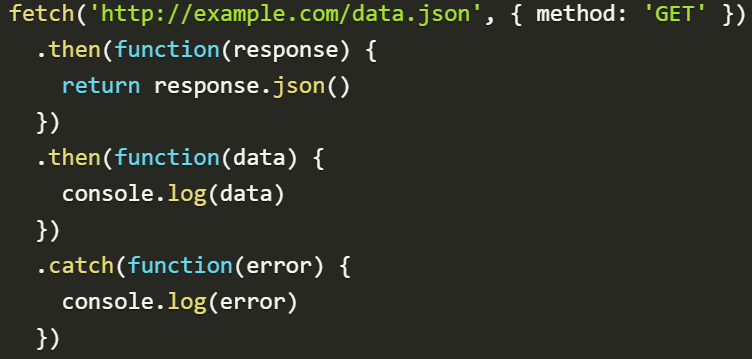
FETCH

El método fetch() nos permite realizar peticiones HTTP de manera asíncrona.



En este ejemplo se está haciendo una petición tipo GET a la URL http://example.com/data.json lo que nos retorna una promesa con un objeto Response. Esta respuesta contiene información como los headers, la url a la que se hizo la petición y el body que contiene la información que estamos solicitando. Para extraer este body usamos el método json().

El método fetch() acepta un segundo argumento opcional para definir configuraciones del request.



uando se requiera enviar un JSON mediante una petición POST se puede usar la propiedad body en este segundo argumento, junto con Content-Type en los headers.

