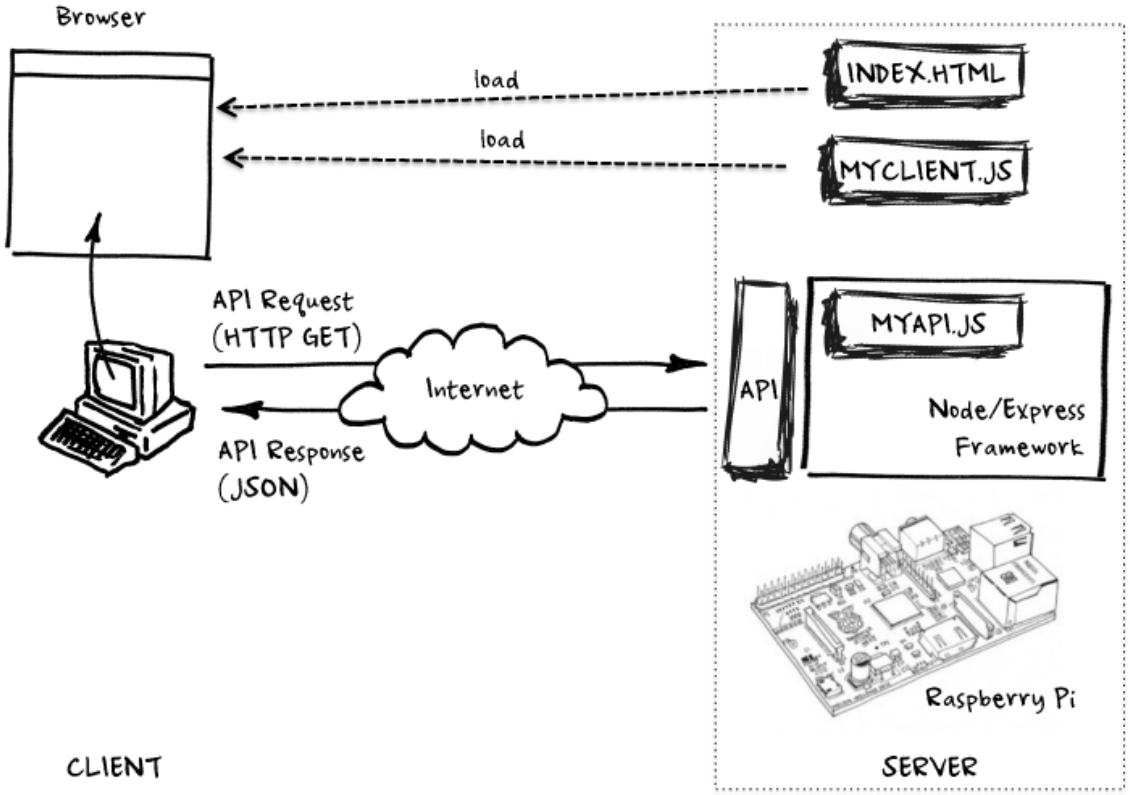
REST API

API

Se conoce como API (del inglés: “Application Programming Interface”), o Interfaz de programación de Aplicaciones al conjunto de rutinas, funciones y procedimientos (métodos) que permite utilizar recursos de un software por otro, sirviendo como una capa de abstracción o intermediario. – Wikipedia

REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad.



Importante

REST (Representational State Transfer) es una arquitectura que se ejecuta sobre HTTP.

RESTful hace referencia a un servicio web que implementa la arquitectura REST.

Características de REST

- *Protocolo cliente/servidor sin estado:* cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como protocolo cliente-caché-servidor sin

estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.

- Las operaciones (métodos) más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro:

    - POST (crear)
    - GET (leer y consultar)
    - PUT (editar)
    - DELETE (eliminar).
    - Otros métodos que se utilizan en RESTful API son OPTIONS y HEAD. Este último se emplea para pasar parámetros de validación, autorización y tipo de procesamiento, entre otras funciones.

- Los objetos en REST siempre se manipulan a partir de la URI. Es la URI y ningún otro elemento, el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

- *Interfaz uniforme:* para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.

- S*istema de capas:* arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.

Ventajas que ofrece REST para el desarrollo

1. Separación entre el cliente y el servidor: el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.

2. Visibilidad, fiabilidad y escalabilidad. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

3. La API REST siempre es independiente del tipo de plataformas o lenguajes: la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que

ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.

Consideraciones

Otro componente de un RESTful API es el "HTTP Status Code", que le informa al cliente o consumidor del API que debe hacer con la respuesta recibida. Estos son una referencia universal de resultado, es decir, al momento de diseñar un RESTful API toma en cuenta utilizar el "Status Code" de forma correcta.

Por ejemplo, el código 200 significa OK, que la consulta ha recibido respuesta desde el servidor o proveedor del API. Los código de estado más utilizados son:

200 OK

201 Created (Creado)

304 Not Modified (No modificado)

400 Bad Request (Error de consulta)

401 Unauthorized (No autorizado)

403 Forbidden (Prohibido)

404 Not Found (No encontrado)

422 Unprocessable Entity (Entidad no procesable)

500 Internal Server Error (Error Interno de Servidor)

Ejemplo de respuesta:

```
Status Code: 200 OK
Access-Control-Allow-Methods: PUT, GET, POST, DELETE, OPTIONS
Connection: Keep-Alive
Content-Length: 186
Content-Type: application/json
Date: Mon, 24 May 2016 15:15:24 GMT
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
access-control-allow-credentials: true
```

Nota de color

REST cambió por completo la ingeniería de software a partir del 2000. Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por *Roy Fielding*, el padre de la especificación HTTP y uno los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su disertación 'Architectural Styles and the Design of Network-based Software Architectures'.