API de REST o API RESTful

¿Qué es una API?

Una API, o Interfaz de Programación de Aplicaciones, es una aplicación con la que nos podemos comunicar para obtener información y realizar ciertas acciones.

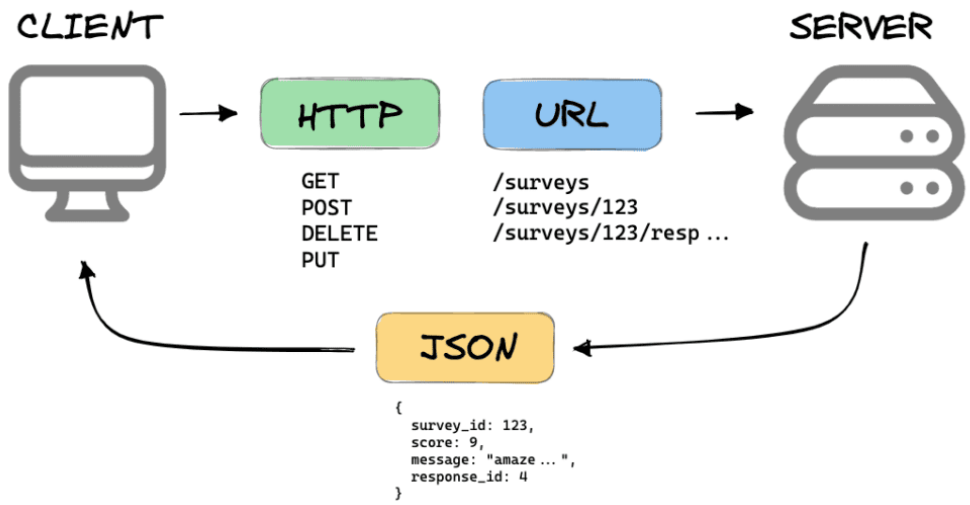
Las API simplifican y aceleran el desarrollo de software y aplicaciones permitiendo a los desarrolladores integrar datos, servicios y capacidades de otras aplicaciones, en lugar de desarrollarlas desde cero. Permiten compartir solo la información necesaria, manteniendo ocultos otros detalles internos del sistema, lo que ayuda a la seguridad del sistema.

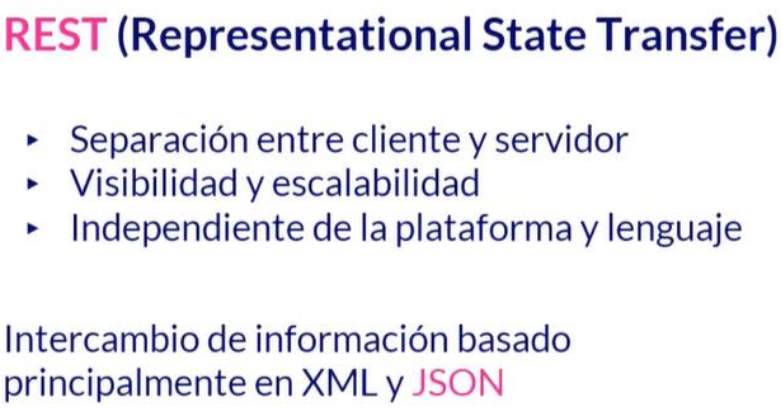
La API puede ser web o no. La API web, se utilizan para permitir la transferencia de datos y funcionalidades a través de Internet mediante el protocolo HTTP.

Una **API de REST** (o API RESTful) es una **interfaz de programación de aplicaciones (API) que sigue los principios de diseño del estilo de la arquitectura REST**. REST significa transferencia de estado representacional y consiste en un conjunto de reglas y recomendaciones para diseñar una API web. Utilizan solicitudes HTTP Verbs como GET, PUT, POST, DELETE,... para interactuar con los recursos.

En el nivel más básico, una API permite que una aplicación o servicio acceda a un recurso dentro de otra aplicación o servicio. La aplicación o servicio que accede a los recursos es el cliente, y la aplicación o servicio que contiene el recurso es el servidor.

REST hace que los datos estén disponibles como recursos que se representan por un URI único. Los clientes solicitan un recurso proporcionando su URI.





HTTP

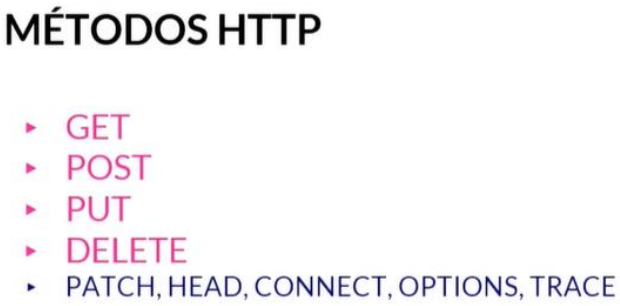
Cuando trabajamos con una API, ya sea creándola o comunicándonos con ella, será a través del protocolo HTTP.

Es un protocolo cliente/servidor sin estado, es decir, cada petición HTTP tiene toda la información necesaria para su procesamiento: ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre ellos.

Petición (Request) HTTP

Cuando nosotros accedemos a una página web desde el navegador o cliente HTTP, estamos realizando una petición o request HTTP a esa web.

Constará de cuatro partes: método, cabeceras, versión del protocolo y el cuerpo del mensaje (body).



<u>Respuesta (Response) HTTP</u>

Siempre que lanzamos una petición, obtendremos una respuesta por parte de la web o API (al menos que el servicio no esté disponible).

Al igual que en la petición, la respuesta se compone de varias partes y aquí veremos las más relevantes que son el código de estado, cabeceras y la respuesta.

Los códigos de estado (status code) son representados por un número y su labor es informativa. Desde el 100 hasta el 599, se dividen según su tipo de información en rangos de 100:

- Del 100 hasta 199 se encuentran las respuestas informativas.
- Del 200 hasta el 299 las respuestas en las que todo ha ido bien.
- Las redirecciones se encuentran desde el 300 hasta el 399.
- Los errores por parte del cliente van del 400 hasta el 499.
- Los errores por parte del servidor van desde el 500 hasta el 599.

Cómo funcionan las API REST

Las API REST se comunican a través de solicitudes HTTP para realizar funciones estándar de bases de datos, como crear, leer, actualizar y eliminar registros (también conocido como CRUD) dentro de un recurso.

Por ejemplo, una API REST usaría:

- una solicitud GET para recuperar un registro
- una solicitud POST crea un nuevo registro
- una solicitud PUT actualiza un registro
- una solicitud DELETE para eliminar.

Ejemplo de llamada al API

Se implementan como peticiones HTTP, en las que:

- La URL representa el recurso:
  http://www.mislibros.es/api/libros/1

- El método (HTTP Verbs) representa la operación:
  GET http://www.mislibros.es/api/libros/1

- El código de estado HTTP representa el resultado:

  200 OK HTTP/1.1

  404 NOT FOUND HTTP/1.1

En el caso de **creación de recursos** la URL estará “abierta” (ya el recurso todavía no existe y por tanto no tiene id) y el método debe ser POST.

Respuestas a la creación de recursos posibles:

403 (acceso prohibido)

400 (petición incorrecta, p.ej. falta un campo o su valor no es válido)

500 (error del lado del servidor, p.ej. se ha caído la BD)

201 (recurso creado correctamente)

Si **actualizamos** un recurso, usamos el método PUT (cambia TODOS los datos). PATCH es otro método estándar HTTP pensado para cambiar solo ciertos datos. Hay frameworks de programación REST que todavía no lo soportan.

Los resultados posibles:

Errores ya vistos con POST

201 (Recurso creado, cuando le pasamos el id deseado al servidor)

200 (Recurso modificado correctamente)

Si **eliminamos** un recurso, usamos el método DELETE

Algunos resultados posibles:

200 OK

404 Not found

500 Server error

Tras ejecutar el DELETE con éxito, las siguientes peticiones GET a la URL del recurso deberían devolver 404

Formato de salida

En función de la petición nuestra API podría devolver uno u otro formato. Utilizaremos JSON ya que es sencillo y simple

Nos fijaremos en el ACCEPT HEADER

GET /v1/geocode HTTP/1.1

Host: api.geocod.io

Accept: application/json

Práctica API Película

Crear proyecto

Instalar Doctrine

Crear la Base de Datos y comprobación

Crear Entidad y comprobación

Crear tabla en BD (migración) y comprobación

Modificar Entidad y tablas y comprobación

Crear Controlador

Instalar extensión VS Code: thunderclient

Cambios en el Controller

Probar los métodos HTTP: GET, POST, PUT y DELETE

Incluye capturas dónde se muestren las pruebas en Thunder client.