¿Qué es una API REST?

Una API, o *interfaz de programación de aplicaciones* es un conjunto de reglas que definen la forma en que aplicaciones o dispositivos pueden conectarse y comunicarse entre sí. Una API REST es una API que se ajusta a los principios de diseño de REST, o el estilo de arquitectura de *transferencia de estado representacional*. Por esta razón, las API REST a veces se denominan API RESTful.

Definido por primera vez en 2000 por el científico informático Dr. Roy Fielding en su tesis doctoral, REST proporciona un nivel relativamente alto de flexibilidad y libertad para los desarrolladores. Esta flexibilidad es solo una de las razones por las que las API REST han surgido como un método común para conectar componentes y aplicaciones en una arquitectura de micro servicios.

Principios de diseño de REST

En el nivel más básico, una API es un mecanismo que permite que una aplicación o servicio acceda a un recurso dentro de otra aplicación o servicio. La aplicación o servicio que realiza el acceso se llama cliente y la aplicación o servicio que contiene el recurso se llama servidor.

Algunas API, como SOAP o XML-RPC, imponen un marco estricto a los desarrolladores. Pero las API REST se pueden desarrollar utilizando prácticamente cualquier lenguaje de programación y admiten una variedad de formatos de datos. El único requisito es que se alineen con los siguientes seis principios de diseño de REST, también conocidos como restricciones de arquitectura:

1. **Interfaz uniforme**. Todas las solicitudes de API para el mismo recurso deben tener el mismo aspecto, sin importar de dónde provenga la solicitud. La API REST debe garantizar que el mismo dato, como el nombre o la dirección de correo electrónico de un usuario, pertenezca a un solo identificador uniforme de recursos (URI). Los recursos no deben ser demasiado grandes, pero

deben contener toda la información que el cliente necesite.

2. **Desacoplamiento cliente-servidor**. En el diseño de API REST, las aplicaciones de cliente y servidor deben ser completamente independientes entre sí. La única información que debe conocer la aplicación del cliente es el URI del recurso solicitado; no puede interactuar con la aplicación del servidor de ninguna otra manera. De manera similar, una aplicación de servidor no debería modificar la aplicación del cliente además de pasarla a los datos solicitados a través de HTTP.

3. **Sin estado**. Las API REST no tienen estado, lo que significa que cada solicitud debe incluir toda la información necesaria para procesarla. En otras palabras, las API REST no requieren ninguna sesión del lado del servidor. Las aplicaciones de servidor no pueden almacenar ningún dato relacionado con la solicitud de un cliente.

4. **Capacidad de caché**. Cuando sea posible, los recursos deben almacenarse en la memoria caché en el lado del cliente o del servidor. Las respuestas del servidor también deben contener información sobre si se permite el almacenamiento en caché para el recurso entregado. El objetivo es mejorar el rendimiento en el lado del cliente, mientras aumenta la escalabilidad en el lado del servidor.

5. **Arquitectura del sistema en capas**. En las API REST, las llamadas y respuestas pasan por diferentes capas. Como regla general, no asuma que las aplicaciones cliente y servidor se conectan directamente entre sí. Puede haber varios intermediarios diferentes en el bucle de comunicación. Las API REST deben diseñarse de modo que ni el cliente ni el servidor puedan saber si se comunican con la aplicación final o con un intermediario.

6. **Código bajo demanda (opcional)**. Las API REST generalmente envían recursos estáticos, pero en ciertos casos, las respuestas también pueden contener código ejecutable (como los subprogramas de Java). En estos casos, el código solo debe ejecutarse bajo demanda.

Cómo funcionan las API REST

Las API REST se comunican mediante solicitudes HTTP para realizar funciones de bases de datos estándar como crear, leer, actualizar y eliminar registros (también conocidas como CRUD) dentro de un recurso. Por ejemplo, una API REST usaría una solicitud GET para recuperar un registro, una solicitud POST para crearlo, una solicitud PUT para actualizar un registro y una solicitud DELETE para eliminarlo. Todos los métodos HTTP se pueden utilizar en llamadas a API. Una API REST bien diseñada es similar a un sitio web que se ejecuta en un navegador web con funcionalidad HTTP incorporada.

El estado de un recurso en un instante particular, o la indicación de fecha y hora, se conoce como representación del recurso. Esta información se puede entregar a un cliente en prácticamente cualquier formato, incluido JavaScript Object Notation (JSON), HTML, XLT, Python, PHP o texto sin formato. JSON es popular porque es legible tanto por humanos como por máquinas, y es independiente del lenguaje de programación.

Los encabezados y parámetros de las solicitudes también son importantes en las llamadas a la API REST porque incluyen información de identificación importante, como metadatos, autorizaciones, identificadores uniformes de recursos (URI), almacenamiento en caché, cookies y más. Los encabezados de solicitud y de respuesta, junto con los códigos de estado HTTP convencionales, se utilizan dentro de API REST bien diseñadas.

Mejores prácticas de API REST

Aunque la flexibilidad es una gran ventaja del diseño de API REST, esa misma flexibilidad permite el diseño de una API que no funciona o tiene un rendimiento deficiente. Por esta razón, los desarrolladores profesionales comparten las mejores prácticas en las especificaciones de la API REST.

La Especificación OpenAPI (OAS) establece una interfaz para describir una API de una forma que permite que cualquier desarrollador o aplicación pueda descubrir y comprender completamente sus parámetros y funcionalidades: puntos finales disponibles, operaciones permitidas en cada punto final, parámetros de operación, métodos de autenticación y otra información. La última versión, OAS3 incluye herramientas prácticas como, por ejemplo, OpenAPI Generator, para generar apéndices de clientes y servidor de API en distintos lenguajes de programación.

La protección de una API REST también comienza con las mejores prácticas de la industria, como el uso de algoritmos hash para la seguridad de las contraseñas y HTTPS para la transmisión segura de datos. Un marco de autorización como OAuth 2.0 puede ayudar a limitar los privilegios de las aplicaciones de terceros. Al usar una indicación de fecha y hora en el encabezado HTTP, una API también puede rechazar cualquier solicitud que llegue después de un determinado período de tiempo. La validación de parámetros y el JSON Web Token son otras formas de garantizar que solo los clientes autorizados puedan acceder a la API.