SERVICIOS REST

¿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 microservicios.

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'. En el campo de las APIs, REST (Representational State Transfer- Transferencia de Estado Representacional) es, a día de hoy, el alfa y omega del desarrollo de servicios de aplicaciones.

En la actualidad no existe proyecto o aplicación que no disponga de una API REST para la creación de servicios profesionales a partir de ese software. Twitter, YouTube, los sistemas de identificación con Facebook… hay cientos de empresas que generan negocio gracias a REST y las APIs REST. Sin ellas, todo el crecimiento en horizontal sería prácticamente imposible. Esto es así porque REST es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet.

Buscando una definición sencilla, 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. A veces es preferible una solución más sencilla de manipulación de datos como REST.

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.

CARACTERÍSTICAS DE LAS API 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 más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- **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.
- **Sistema de capas:** arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.
- **Uso de hipermedios:** hipermedia es un término acuñado por Ted Nelson en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Para cualquier API REST es obligatorio disponer del principio HATEOAS (Hypermedia As The Engine Of Application State – Hipermedia Como Motor del Estado de la Aplicación) para ser una verdadera API REST. Este principio es el que define que cada vez que se hace una petición al servidor y éste devuelve una respuesta, parte de la información que contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente.

Devolución de una petición a una API REST según el principio HATEOAS (enlaza a un tutorial explicativo del concepto de hipermedia en API REST con un ejemplo práctico de una petición a una base de datos de automóviles):

```
{     "id": 78,
    "nombre": "Juan",
    "apellido": "García",
    "coches": [
           {
        "coche": "http://miservidor/concesionario/api/v1/clientes/78/coches/1033"
           },
           {
         "coche": "http://miservidor/concesionario/api/v1/clientes/78/coches/3889"
           }
    ]
}
```

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 (enlace externo a ibm.com), 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 (enlace externo a ibm.com) 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.

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.

Fuente:

https://www.ibm.com/mx-es/topics/rest-apis

https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/