Proyecto FFE FIDESOL

¿Qué son las API RESTful?

La API RESTful es una interfaz que dos sistemas de computación utilizan para intercambiar información de manera segura a través de Internet. La mayoría de las aplicaciones para empresas deben comunicarse con otras aplicaciones internas o de terceros para llevar a cabo varias tareas. Las API RESTful admiten este intercambio de información porque siguen estándares de comunicación de software seguros, confiables y eficientes.

¿Qué es una API?

Una interfaz de programa de aplicación (API) define las reglas que se deben seguir para comunicarse con otros sistemas de software. Los desarrolladores exponen o crean API para que otras aplicaciones puedan comunicarse con sus aplicaciones mediante programación. Se puede pensar en una API web como una puerta de enlace entre los clientes y los recursos de la Web.

¿Qué es REST?

La transferencia de estado representacional (REST) es una arquitectura de software que impone condiciones sobre cómo debe funcionar una API. Puede implementarla y modificarla fácilmente, lo que brinda visibilidad y portabilidad entre plataformas a cualquier sistema de API. REST se creó como una guía para administrar la comunicación en una red compleja como Internet. Es posible utilizar una arquitectura basada en REST para admitir comunicaciones confiables y de alto rendimiento a escala.

Principios de diseño de REST

La aplicación o servicio que accede a los recursos es el cliente, y la aplicación o servicio que contiene el recurso es el servidor. Pero pueden desarrollar API REST utilizando prácticamente cualquier lenguaje de programación y admitir una variedad de formatos de datos. El único requisito es que se alineen con los siguientes seis principios de diseño REST, también conocidos como *restricciones arquitectónicas*.

Restricciones arquitectónicas	Descripción
Interfaz uniforme	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 de recursos uniforme (URI). Los recursos no deben ser demasiado grandes, pero deben contener toda la información que el cliente pueda necesitar.
Desacoplamiento cliente-servidor	Las aplicaciones cliente y servidor deben ser completamente independientes entre sí. La única información que debe conocer la aplicación cliente es el URI del recurso solicitado; no puede interactuar con la aplicación servidor de ninguna otra forma. Del mismo modo, una aplicación servidor no debería modificar la aplicación cliente más allá de pasarle los datos solicitados a través de HTTP.
Sin estado	Las API REST no requieren ninguna sesión del lado del servidor. Las aplicaciones de servidor no pueden almacenar ningún dato relacionado con una solicitud de cliente. Lo que significa que cada solicitud debe incluir toda la información necesaria para procesarla

Capacidad de almacenamiento en caché	El objetivo es mejorar el rendimiento en el lado del cliente y, al mismo tiempo, aumentar la escalabilidad en el lado del servidor. Las respuestas del servidor también deben contener información sobre si se permite el almacenamiento en caché para el recurso entregado.
Arquitectura de sistema en capas	Las API REST deben diseñarse de tal manera que ni el cliente ni el servidor puedan saber si se comunican con la aplicación final o con un intermediario. Las llamadas y respuestas pasan por diferentes capas
Código bajo demanda (opcional)	Las API REST suelen enviar recursos estáticos, pero, en algunos casos, las respuestas también pueden contener código ejecutable (como applets de Java). En estos casos, el código solo debe ejecutarse bajo demanda.

Cómo funcionan las API REST

La función básica de una API RESTful es la misma que navegar por Internet. Cuando requiere un recurso, el cliente se pone en contacto con el servidor mediante la API. Los desarrolladores de API explican cómo el cliente debe utilizar la API REST en la documentación de la API de la aplicación del servidor.

Pasos generales para cualquier llamada a la API REST:

1. El cliente envía una solicitud al servidor. El cliente sigue la documentación de la API para dar formato a la solicitud de una manera que el servidor comprenda.
2. El servidor autentica al cliente y confirma que este tiene el derecho de hacer dicha solicitud.
3. El servidor recibe la solicitud y la procesa internamente.

4. Luego, devuelve una respuesta al cliente. Esta respuesta contiene información que dice al cliente si la solicitud se procesó de manera correcta. La respuesta también incluye cualquier información que el cliente haya solicitado.

Beneficios que ofrecen las API RESTful

Escalabilidad	REST optimiza las interacciones entre el cliente y el servidor. La tecnología sin estado elimina la carga del servidor porque este no debe retener la información de solicitudes pasadas del cliente. El almacenamiento en caché bien administrado elimina de forma parcial o total algunas interacciones entre el cliente y el servidor.
Flexibilidad	Los servicios web RESTful admiten una separación total entre el cliente y el servidor. Simplifican y desacoplan varios componentes del servidor, de manera que cada parte pueda evolucionar de manera independiente. Los cambios de la plataforma o la tecnología en la aplicación del servidor no afectan la aplicación del cliente. La capacidad de ordenar en capas las funciones de la aplicación aumenta la flexibilidad aún más.
Independencia	Las API REST son independientes de la tecnología que se utiliza. Puede escribir aplicaciones del lado del cliente y del servidor en diversos lenguajes de programación, sin afectar el diseño de la API. También puede cambiar la tecnología subyacente en cualquiera de los lados sin que se vea afectada la comunicación

Formato JSON

JSON es un formato para la especificación de datos usado de manera habitual en las aplicaciones y sitios web. Su principal utilidad se da en el intercambio de datos entre sistemas informáticos. Se dice que es ligero porque los datos no suelen ocupar mucho, como sí ocurre con otros lenguajes de intercambio de datos.

Características::

- Su facilidad de escritura, así como la lectura
- Ser especialmente ligero, es decir, con poco peso en bytes
- Es fácil de analizar y de generar
- Es compatible con la mayoría de los lenguajes de programación

No obstante, al usar la notación de los literales de objeto JavaScript, el código resulta extremadamente familiar para los desarrolladores web.

Funciones:

Transferencia de datos en aplicaciones y servicios web	Un uso común de JSON es obtener datos de un servidor web a través de una solicitud HTTP. Este intercambio de datos puede ser entre dos servidores completos o entre un servidor y un cliente.
Configuración y almacenamiento de datos en aplicaciones	Los formatos JSON también se utilizan para almacenar configuraciones de las aplicaciones, ya sean introducidas en el código o como soporte al almacenamiento de las configuraciones introducidas por el usuario. Además, muchos sistemas de bases de datos están adoptando este formato como notación para el almacenamiento de los datos.
Intercambio de datos entre diferentes lenguajes de programación	Por ejemplo, un servidor puede estar escrito en PHP y el cliente web que consume los datos del backend PHP puede estar escrito en Javascript, Sin embargo, ambos pueden enviar y recibir datos en formato JSON.

Ventajas

Facilidad de uso y lectura	JSON es fácil de leer y escribir. Su estructura simple y clara facilita la comprensión de los datos a simple vista.
Eficiencia en la transmisión de datos	JSON es eficiente en la transmisión de datos, lo que lo hace ideal para la comunicación entre el cliente y el servidor en aplicaciones web
Compatibilidad con numerosos lenguajes de programación	JSON es compatible con una amplia gama de lenguajes de programación, lo que lo convierte en una opción versátil para el intercambio de datos.