📩 Informe API RESTful 📩

¿Qué son las APIS?

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. Un ejemplo de esto sería el caso de un camarero en un restaurante; un cliente pide algo al camarero y este transmite la comanda a la cocina, esta le da al camarero una comida acorde a la información que le ha dado el cliente y este último le da la comida al cliente. Con este ejemplo se intenta expresar que la cocina (un software cualquiera) y el cliente (otro software cualquiera) no interactúan en ningún momento, si no que el que hace de intermediario es el camarero (la API).

¿Qué es REST?

REST (Transferencia de Estado Representacional) es una arquitectura de software que establece reglas para el funcionamiento de una API. Fue diseñada para facilitar la comunicación eficiente en redes complejas como Internet. Las API que siguen esta arquitectura se llaman API REST o RESTful, términos que suelen usarse como sinónimos. REST permite crear sistemas escalables, confiables y fáciles de implementar, con buena visibilidad y portabilidad entre plataformas.

Algunos de los principios del estilo arquitectónico de REST son:

- La arquitectura REST se basa en varios principios clave que permiten diseñar servicios web escalables, eficientes y fáciles de mantener. Uno de los pilares es **la interfaz uniforme**, que garantiza que todos los recursos se transfieran en un formato estándar conocido como "representación". Esta interfaz impone cuatro restricciones: los recursos deben ser identificables mediante un URI; los clientes deben recibir metadatos suficientes para modificar o eliminar los recursos; las respuestas deben ser autodescriptivas, indicando cómo deben ser utilizadas; y deben incluir hipervínculos que permitan descubrir otros recursos relacionados.

- Otro principio importante es el de **tecnología sin estado**, lo que significa que cada solicitud del cliente debe contener toda la información necesaria

para ser procesada, sin depender de solicitudes anteriores. Esto permite una mayor escalabilidad y simplicidad en el servidor, ya que no necesita recordar el estado de los clientes.

- REST también utiliza una **arquitectura por capas**, donde el cliente puede comunicarse con el servidor a través de intermediarios (como proxies, gateways o servidores adicionales). Estas capas pueden encargarse de funciones como seguridad, almacenamiento o procesamiento, y son transparentes para el cliente.

- El **almacenamiento en caché** es otra característica clave. Permite guardar ciertas respuestas en la memoria caché del cliente o de intermediarios para reducir la carga del servidor y mejorar los tiempos de respuesta. Esto es útil, por ejemplo, cuando se repiten recursos como imágenes o encabezados en distintas páginas web.

- REST permite **código bajo demanda**, lo que significa que el servidor puede enviar código ejecutable al cliente, como scripts, para mejorar su funcionalidad. Un ejemplo típico es la validación de formularios en tiempo real dentro del navegador.

¿Qué es 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. Por ejemplo, para generar nóminas mensuales, su sistema interno de cuentas debe compartir datos con el sistema bancario de su cliente para automatizar la facturación y comunicarse con una aplicación interna de planillas de horarios. Las API RESTful admiten este intercambio de información porque siguen estándares de comunicación de software seguros, confiables y eficientes.

Algunos beneficios que ofrecen las API RESTful

La arquitectura REST permite diseñar servicios web robustos, modulares y eficientes. Uno de sus principios clave es la interfaz uniforme, que asegura que los recursos se identifiquen mediante URI y se transfieran en formatos estándar llamados representaciones. Esta interfaz también proporciona

metadatos y enlaces a otros recursos, permitiendo a los clientes interactuar de manera clara y autónoma con el sistema.

REST funciona bajo una tecnología sin estado, lo que significa que cada solicitud del cliente es independiente y contiene toda la información necesaria para que el servidor la procese. Esto reduce la carga en el servidor y mejora la eficiencia. Además, el sistema por capas permite distribuir el servicio en distintos niveles (como seguridad, lógica de negocio o almacenamiento), lo que mejora la organización interna del sistema sin afectar al cliente.

Otra ventaja importante es el almacenamiento en caché, que permite reutilizar respuestas previamente almacenadas para reducir el tráfico entre el cliente y el servidor, acelerando las respuestas y mejorando el rendimiento.

REST también admite código bajo demanda, permitiendo que el servidor envíe scripts u otros fragmentos de código al cliente para ejecutar funciones específicas, como validaciones en formularios, lo que añade dinamismo a la experiencia del usuario.

Gracias a estas características, REST ofrece una alta escalabilidad, ya que minimiza la sobrecarga del servidor y permite manejar grandes volúmenes de solicitudes sin pérdida de rendimiento. Asimismo, proporciona gran flexibilidad, ya que separa completamente el cliente del servidor. Esto permite que cada parte evolucione o se actualice de forma independiente, como cambiar la base de datos sin modificar la lógica de la aplicación.

Finalmente, REST promueve la independencia tecnológica, permitiendo que los clientes y servidores se desarrollen en distintos lenguajes de programación y plataformas sin afectar la interoperabilidad. Esto facilita la integración entre sistemas diversos y su mantenimiento a largo plazo.

¿Cómo funcionan las APIS RESTful?

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. A continuación, se indican los 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.

Los detalles de la solicitud y la respuesta de la API REST varían un poco en función de cómo los desarrolladores de la API la hayan diseñado.

¿Qué contiene la solicitud del cliente de la API RESTful?

Las API RESTful requieren que las solicitudes contengan los siguientes componentes principales:

Identificador único de recursos

El servidor identifica cada recurso con identificadores únicos de recursos. En los servicios REST, el servidor por lo general identifica los recursos mediante el uso de un localizador uniforme de recursos (URL). El URL especifica la ruta hacia el recurso. Un URL es similar a la dirección de un sitio web que se ingresa al navegador para visitar cualquier página web. El URL también se denomina punto de conexión de la solicitud y especifica con claridad al servidor qué requiere el cliente.

Método

Los desarrolladores a menudo implementan API RESTful mediante el uso del protocolo de transferencia de hipertexto (HTTP). Un método de HTTP informa al servidor lo que debe hacer con el recurso. A continuación, se indican cuatro métodos de HTTP comunes:

- *GET:*
  Los clientes utilizan GET para acceder a los recursos que están ubicados en el URL especificado en el servidor. Pueden almacenar en caché las solicitudes GET y enviar parámetros en la solicitud de la API RESTful para indicar al servidor que filtre los datos antes de enviarlos.

- *POST:*
  Los clientes usan POST para enviar datos al servidor. Incluyen la representación de los datos con la solicitud. Enviar la misma solicitud POST varias veces produce el efecto secundario de crear el mismo recurso varias veces.

- *PUT:*
  Los clientes utilizan PUT para actualizar los recursos existentes en el servidor. A diferencia de POST, el envío de la misma solicitud PUT varias veces en un servicio web RESTful da el mismo resultado.

- *DELETE:*
  Los clientes utilizan la solicitud DELETE para eliminar el recurso. Una solicitud DELETE puede cambiar el estado del servidor. Sin embargo, si el usuario no cuenta con la autenticación adecuada, la solicitud fallará.

Encabezados de HTTP

Los encabezados de solicitudes son los metadatos que se intercambian entre el cliente y el servidor. Por ejemplo, el encabezado de la solicitud indica el formato de la solicitud y la respuesta, proporciona información sobre el estado de la solicitud, etc.

- *Datos:*
  Las solicitudes de la API REST pueden incluir datos para que los métodos POST, PUT y otros métodos HTTP funcionen de manera correcta.

- *Parámetros:*
  Las solicitudes de la API RESTful pueden incluir parámetros que brindan al servidor más detalles sobre lo que se debe hacer. A continuación, se indican algunos tipos de parámetros diferentes:
  - Los parámetros de ruta especifican los detalles del URL.
  - Los parámetros de consulta solicitan más información acerca del recurso.
  - Los parámetros de cookie autentican a los clientes con rapidez.

¿Qué son los métodos de autenticación de la API RESTful?

Un servicio web RESTful debe autenticar las solicitudes antes de poder enviar una respuesta. La autenticación es el proceso de verificar una identidad. Por ejemplo, puede demostrar su identidad mostrando una tarjeta de identificación o una licencia de conducir. De forma similar, los clientes de los servicios RESTful deben demostrar su identidad al servidor para establecer confianza.

La API RESTful tiene cuatro métodos comunes de autenticación:

Autenticación HTTP
HTTP define algunos esquemas de autenticación que se pueden utilizar directamente cuando se implementa la API REST. A continuación, se indican dos de estos esquemas:

- *Autenticación básica:*
  En la autenticación básica, el cliente envía el nombre y la contraseña del usuario en el encabezado de la solicitud. Los codifica con base64, que es una técnica de codificación que convierte el par en un conjunto de 64 caracteres para su transmisión segura.

- *Autenticación del portador:*
  El término autenticación del portador se refiere al proceso de brindar el control de acceso al portador del token. El token del portador suele ser una cadena de caracteres cifrada que genera el servidor como respuesta a una solicitud de inicio de sesión. El cliente envía el token en los encabezados de la solicitud para acceder a los recursos.

- *Claves de la API:*
  Las claves de la API son otra opción para la autenticación de la API REST. En este enfoque, el servidor asigna un valor único generado a un cliente por primera vez. Cada vez que el cliente intenta acceder a los recursos, utiliza la clave de API única para su verificación. Las claves de API son menos seguras debido a que el cliente debe transmitir la clave, lo que la vuelve vulnerable al robo de red.

- *OAuth:*
  OAuth combina contraseñas y tokens para el acceso de inicio de sesión de alta seguridad a cualquier sistema. El servidor primero solicita una

contraseña y luego solicita un token adicional para completar el proceso de autorización. Puede verificar el token en cualquier momento y, también, a lo largo del tiempo, con un alcance y duración específicos.

¿Qué contiene la respuesta del servidor de la API RESTful?

Los principios de REST requieren que la respuesta del servidor contenga los siguientes componentes principales:

Línea de estado
La línea de estado contiene un código de estado de tres dígitos que comunica si la solicitud se procesó de manera correcta o dio error. Por ejemplo, los códigos 2XX indican el procesamiento correcto, pero los códigos 4XX y 5XX indican errores. Los códigos 3XX indican la redirección de URL.

A continuación, se enumeran algunos códigos de estado comunes:

- 200: respuesta genérica de procesamiento correcto.
- 201: respuesta de procesamiento correcto del método POST.
- 400: respuesta incorrecta que el servidor no puede procesar.
- 404: recurso no encontrado.

Cuerpo del mensaje
El cuerpo de la respuesta contiene la representación del recurso. El servidor selecciona un formato de representación adecuado en función de lo que contienen los encabezados de la solicitud. Los clientes pueden solicitar información en los formatos XML o JSON, lo que define cómo se escriben los datos en texto sin formato. Por ejemplo, si el cliente solicita el nombre y la edad de una persona llamada John, el servidor devuelve una representación JSON como la siguiente:

'{"name":"John", "age":30}'

Encabezados
La respuesta también contiene encabezados o metadatos acerca de la respuesta. Estos brindan más contexto sobre la respuesta e incluyen información como el servidor, la codificación, la fecha y el tipo de contenido.