REST API

Una REST API o API RESTful, es una interfaz de programación de aplicaciones (API por sus siglas en inglés) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful.

Las REST API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Las REST API permiten interactuar con una computadora o un sistema para obtener datos o ejecutar una función, de manera que el sistema comprenda la solicitud y la cumpla. Funcionan como mediadores entre los usuarios o clientes y los recursos o servicios web que quieren obtener. Con ellas, las empresas pueden compartir recursos e información mientras conservan la seguridad, el control y la autenticación, lo cual les permite determinar el contenido al que puede acceder cada usuario.

Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta). Por ejemplo, el diseño de una REST API de servicio meteorológico podría requerir que el usuario escribiera un código postal y que el productor diera una respuesta en dos partes: la primera sería la temperatura máxima y la segunda la mínima.

¿Qué es REST?

REST (transferencia de estado representacional) es una arquitectura de software que impone condiciones sobre cómo debe funcionar una REST API. En un principio, REST se creó como una guía para administrar la comunicación en una red compleja como Internet. Los servicios web que implementan una arquitectura de REST se llaman servicios web RESTful, los términos REST API y API RESTful también se utilizan para referirse a estos servicios.

Cuando el cliente envía una solicitud a través de una API de RESTful transfiere una representación del estado del recurso requerido a quien lo haya solicitado. La información se entrega por medio del protocolo HTTP en uno de estos formatos: JSON (JavaScript Object Notation), HTML, XLT, texto sin formato, entre otros. JSON es el más popular porque es fácil para los humanos leer y escribir, y para las máquinas analizar y generar.

REST en sí mismo no es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura. Los desarrolladores de las REST API pueden implementarlo de distintas maneras, sin embargo, para que una API se considere de RESTful, debe cumplir los siguientes criterios:

- Arquitectura cliente-servidor compuesta de clientes, servidores y recursos, con la gestión de solicitudes a través de HTTP.
- Comunicación entre el cliente y el servidor sin estado, lo cual implica que no se almacena la información del cliente entre las solicitudes GET y que cada una de ellas es independiente y está desconectada del resto.
- Datos que pueden almacenarse en caché y optimizan las interacciones entre el cliente y el servidor.
- Una interfaz uniforme entre los elementos, para que la información se transfiera de forma estandarizada. Para ello deben cumplirse las siguientes condiciones:
    - Los recursos solicitados deben ser identificables e independientes de las representaciones enviadas al cliente.
    - El cliente debe poder manipular los recursos a través de la representación que recibe, ya que esta contiene suficiente información.
    - Los mensajes deben ser autodescriptivos, es decir, que al enviarse al cliente contienen la información necesaria para describir cómo debe procesarla.
    - Debe contener hipertexto o hipermedios, lo cual significa que cuando el cliente acceda a algún recurso, debe poder utilizar hipervínculos para buscar las demás acciones que se encuentren disponibles en ese momento.
- Un sistema en capas que organiza en jerarquías invisibles para el cliente cada uno de los servidores (los encargados de la seguridad, del equilibrio de carga, etc.) que participan en la recuperación de la información solicitada.
- Código disponible según se solicite (opcional), es decir, la capacidad para enviar códigos ejecutables del servidor al cliente cuando se requiera, lo cual amplía las funciones del cliente.

Si bien la API de REST debe cumplir estos criterios, resulta más fácil de usar que otros protocolos definidos como SOAP (protocolo simple de acceso a objetos), el cual tiene

requisitos específicos, como la mensajería XML, la seguridad y el cumplimiento integrados de las operaciones. Por el contrario, REST es un conjunto de pautas que pueden implementarse según sea necesario.

Funcionamiento de una REST API

La función básica de una REST API es la misma que navegar por Internet, el cliente se pone en contacto con el servidor mediante la REST API cuando requiere un recurso (ver figura 1). Los desarrolladores explican cómo el cliente debe utilizarla en la documentación de la aplicación del servidor. A continuación, se indican los pasos generales para cualquier llamada a la REST API:

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 la 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. Finalmente devuelve una respuesta al cliente. Esta respuesta contiene información que dice al cliente si la solicitud se procesó de manera correcta, y también incluye lo que el cliente haya solicitado.

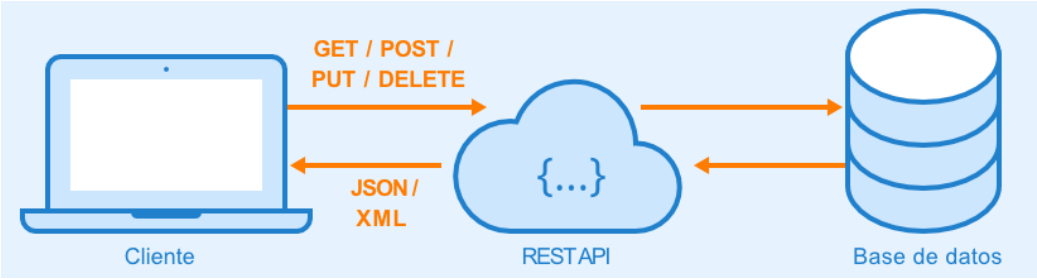

Figura 1. Esquema de funcionamiento simplificado de una REST API

Solicitud del cliente

Como se mencionó anteriormente, una REST API funciona a través del protocolo HTTP, por lo tanto para que el cliente pueda realizar una solicitud debe cumplir con la

estructura de llamadas definidas por el protocolo. Los encabezados y los parámetros de una solicitud HTTP son importantes porque contienen información de los metadatos, la autorización, el identificador único de recursos (URI), el almacenamiento en caché, las cookies y otros elementos de la solicitud. A continuación, se describen los componentes principales de las solicitudes API RESTful:

- **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 la ruta específica hacia el recurso, es decir, un URL. Un URL es similar a la dirección de un sitio web que se ingresa al navegador para visitar cualquier página web.
- **Método.** Un método de HTTP informa al servidor lo que debe hacer con el recurso. Los cuatro métodos de HTTP más comunes son GET, POST, PUT y DELETE. GET sirve para consultar recursos que están ubicados en el URL especificado. POST envía datos al servidor y el cliente debe incluir la representación de los datos con la solicitud, podría considerarse equivalente a la sentencia INSERT en una base de datos. PUT actualiza los recursos existentes, a diferencia de POST este método sería equivalente al UPDATE. DELETE elimina el recurso. Para los últimos tres métodos, el cliente puede cambiar el estado del servidor, y 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.
- **Carga útil.** En el cuerpo de una solicitud HTTP se pueden incluir datos. Por lo general, en el contexto de las REST API se la utiliza para que los métodos POST, PUT y otros métodos HTTP puedan insertar o modificar datos en el servidor.

Autenticación de clientes

Los clientes de los servicios RESTful deben demostrar su identidad al servidor para establecer confianza. Para ello se debe deben autenticar las solicitudes antes de enviar una respuesta. La autenticación es el proceso de verificar una identidad. A continuación se listan algunos métodos de autenticación utilizados:

- **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.
- **OAuth.** Este método 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. El token puede estar definido con un alcance y duración específicos.

Respuesta del servidor

Los principios de REST requieren que la respuesta del servidor contenga tres componentes principales: la línea de estado, el cuerpo del mensaje y encabezados.

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 del URL. Algunos códigos de estado comunes son: 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).

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 Juan, el servidor devuelve una representación JSON como la siguiente: '{"nombre":"Juan", "edad":30}'.

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.

Documentación de una REST API

Para usar e integrar una REST API, los servidores proporcionan una documentación. Esta incluye información detallada sobre los puntos de consulta, métodos, recursos, protocolos de autenticación, parámetros y encabezados disponibles de una API, así como ejemplos de solicitudes y respuestas comunes (ver ejemplos en las figuras 2 y 3).

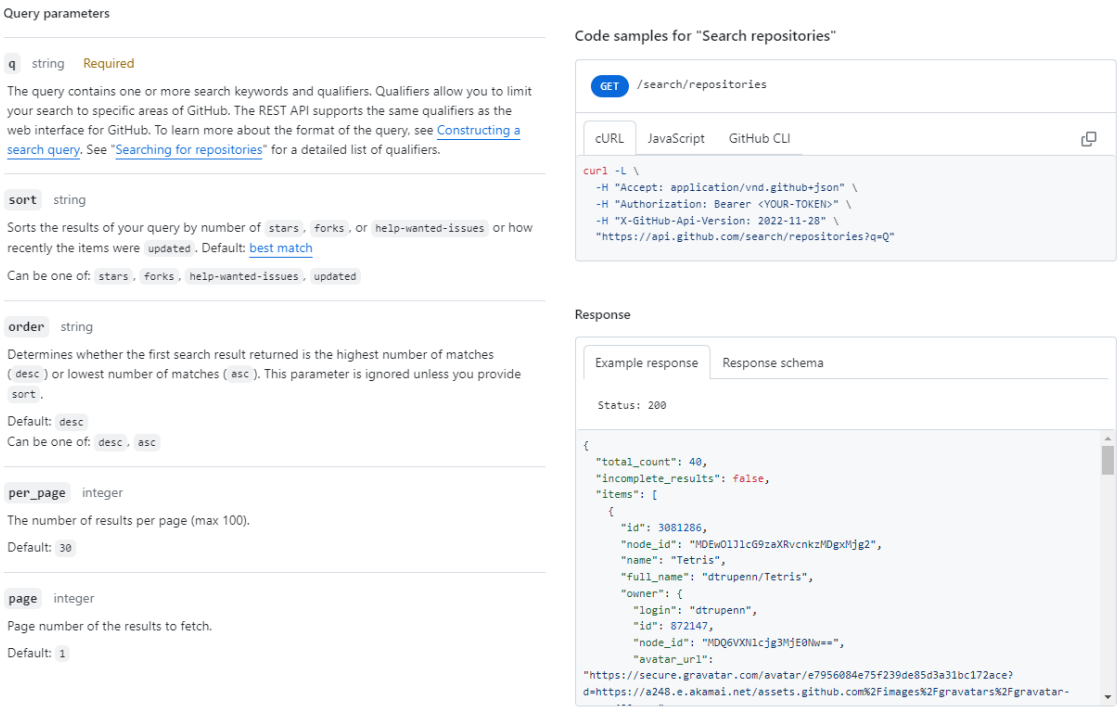

Figura 2. Documentación REST API de Github.

La documentación juega un papel crucial para garantizar el éxito de cualquier REST API. Una buena documentación mejora la experiencia del desarrollador de backend. Por ejemplo, la documentación interna facilita la colaboración entre equipos, reduce la duplicación de código y agiliza el proceso de incorporación de nuevos desarrolladores.

Por su parte, la documentación pública de API ayuda a los consumidores potenciales a comprender y experimentar con ella, lo que genera una mayor adopción de servicios externos.

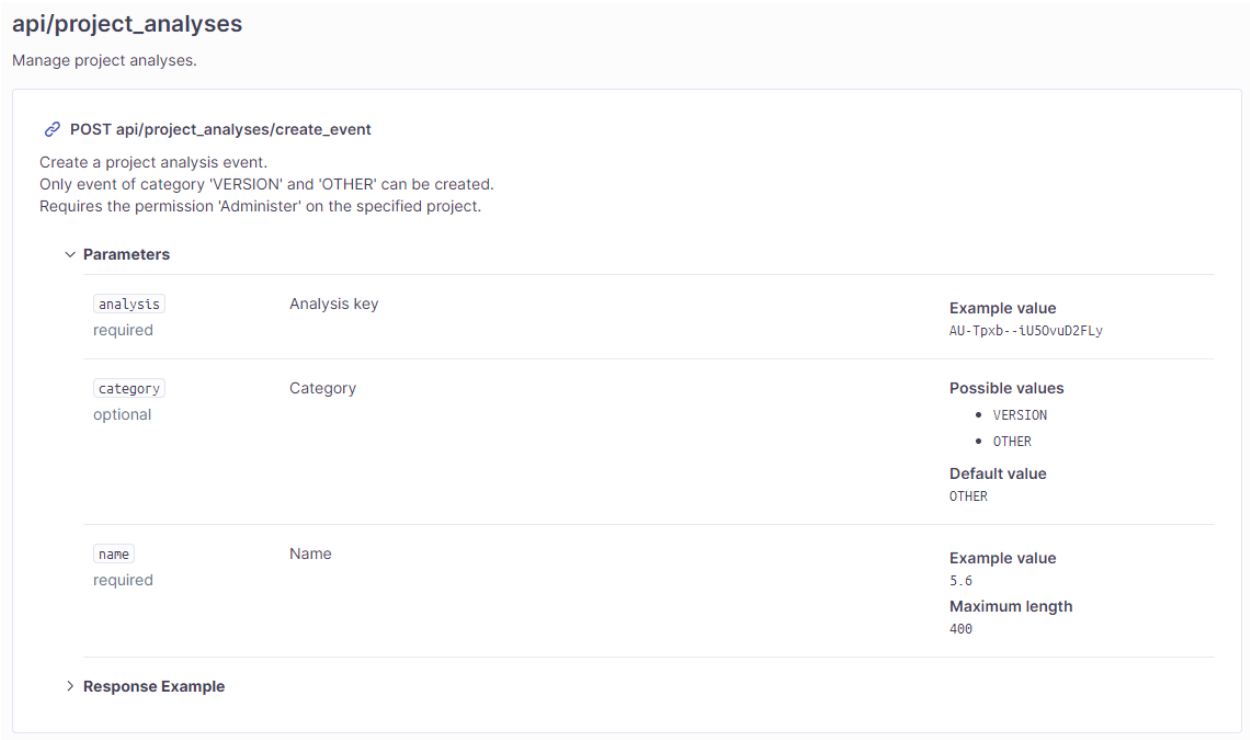

Figura 3. Documentación REST API de Sonar Cloud.

Beneficios que ofrecen las REST API

A continuación, se describen algunos de los beneficios que incluyen las API RESTful:

- **Escalabilidad.** Los sistemas que implementan REST API pueden escalar de forma eficiente porque 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. Todas estas características admiten la escalabilidad, sin provocar cuellos de botella en la comunicación que reduzcan el rendimiento.

- **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. Por ejemplo, los desarrolladores pueden efectuar cambios en la capa de la base de datos sin tener que volver a escribir la lógica de la aplicación.
- **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.

Referencias

- https://www.redhat.com/es/topics/api/what-is-a-rest-api
- https://aws.amazon.com/es/what-is/restful-api/
- https://www.postman.com/api-platform/api-documentation/
- https://docs.github.com/en/rest/search/search?apiVersion=2022-11-28#search-repositories
- https://sonarcloud.io/web_api/api/project_analyses?deprecated=false