API REST

Ing. Luis Guillermo Molero Suárez

Una API REST (también conocida como API RESTful) es una interfaz de programación de aplicaciones (API o API web) que se ajusta a las limitaciones del estilo arquitectónico REST y permite la interacción con los servicios web RESTful. REST significa transferencia de estado representacional y fue creado por el científico informático Roy Fielding.

Una API es un conjunto de definiciones y protocolos para crear e integrar software de aplicación. A veces se lo conoce como un contrato entre un proveedor de información y un usuario de información, que establece el contenido requerido por el consumidor (la llamada) y el contenido requerido por el productor (la respuesta). Por ejemplo, el diseño de la API para un servicio meteorológico podría especificar que el usuario proporcione un código postal y que el productor responda con una respuesta de 2 partes, la primera es la temperatura alta y la segunda la baja.

En otras palabras, si desea interactuar con una computadora o sistema para recuperar información o realizar una función, una API lo ayuda a comunicar lo que desea a ese sistema para que pueda comprender y cumplir la solicitud. Puede pensar en una API como un mediador entre los usuarios o clientes y los recursos o servicios web que desean obtener. También es una forma de que una organización comparta recursos e información mientras mantiene la seguridad, el control y la autenticación, lo que determina quién tiene acceso a qué.

Otra ventaja de una API es que no es necesario que conozca los detalles del almacenamiento en caché: cómo se recupera su recurso o de dónde proviene. Más información sobre las API

REST

REST es un conjunto de restricciones arquitectónicas, no un protocolo ni un estándar. Los desarrolladores de API pueden implementar REST de diversas formas.

Cuando se realiza una solicitud de cliente a través de una API RESTful, transfiere una representación del estado del recurso al solicitante o al punto final. Esta información, o representación, se entrega en uno de varios formatos a través de HTTP: JSON (notación de objetos Javascript), HTML, XLT, Python, PHP o texto sin formato. JSON es el lenguaje de programación más popular en general porque, a pesar de su nombre, es independiente del lenguaje y es legible tanto por humanos como por máquinas.

Algo más a tener en cuenta: los encabezados y los parámetros también son importantes en los métodos HTTP de una solicitud HTTP de la API RESTful, ya que contienen información de identificación importante en cuanto a los metadatos de la solicitud, la autorización, el identificador uniforme de recursos (URI), el almacenamiento en caché, las cookies y más. Hay encabezados de solicitud y encabezados de respuesta, cada uno con su propia información de conexión HTTP y códigos de estado.

Para que una API se considere RESTful, debe cumplir con estos criterios:

- Una arquitectura cliente-servidor compuesta por clientes, servidores y recursos, con solicitudes gestionadas a través de HTTP.
- Comunicación cliente-servidor sin estado, lo que significa que no se almacena información del cliente entre las solicitudes de obtención y cada solicitud es independiente y no está conectada.
- Datos almacenables en caché que agilizan las interacciones cliente-servidor.
- Una interfaz uniforme entre los componentes para que la información se transfiera de forma estándar. Esto requiere que:
    - Los recursos solicitados son identificables y separados de las

representaciones enviadas al cliente.

- Los recursos pueden ser manipulados por el cliente a través de la representación que reciben porque la representación contiene suficiente información para hacerlo.
- Los mensajes autodescriptivos devueltos al cliente tienen suficiente información para describir cómo el cliente debe procesarlos.
- el hipertexto / hipermedia está disponible, lo que significa que después de acceder a un recurso, el cliente debería poder utilizar hipervínculos para encontrar todas las demás acciones disponibles actualmente que puede realizar.

- Un sistema en capas que organiza cada tipo de servidor (los responsables de la seguridad, balanceo de carga, etc.) implicó la recuperación de la información solicitada en jerarquías, invisibles para el cliente.
- Código a pedido (opcional): la capacidad de enviar código ejecutable desde el servidor al cliente cuando se solicita, extendiendo la funcionalidad del cliente.

Aunque la API REST tiene estos criterios para cumplir, todavía se considera más fácil de usar que un protocolo prescrito como SOAP (Protocolo simple de acceso a objetos), que tiene requisitos específicos como mensajería XML y seguridad incorporada y cumplimiento de transacciones que lo hacen más lento y más pesado.

Por el contrario, REST es un conjunto de pautas que se pueden implementar según sea necesario, lo que hace que las API REST sean más rápidas y livianas, con una mayor escalabilidad, perfecto para Internet de las cosas (IoT) y el desarrollo de aplicaciones móviles.

¿Cómo utilizar las API REST?

Con las API REST, generalmente enviamos solicitudes HTTP (más sobre HTTP aquí) como GET, PUT o POST (y más). Exploremos la API de The Movie Database. Esta API está basada en JSON, lo que significa que la respuesta se devuelve en formato JSON. Si necesita un repaso sobre JSON, puede leer más aquí. Si desea realizar solicitudes usted mismo a esta API, asegúrese de obtener un token de API (algunas API requieren autenticación porque quieren identificar al cliente).

Supongamos que queremos encontrar las últimas películas. Podemos usar el punto final para las últimas películas. La URL (donde está alojado el recurso) es: https://api.themoviedb.org/3/movie/latest. Según el "contrato", necesitamos un parámetro obligatorio (la clave API) y un parámetro opcional para el código de idioma (el idioma al que queremos que se traduzcan los resultados). Si no se proporciona ningún código, el texto no se traducirá del idioma original).

Podemos utilizar una herramienta como cURL para realizar la solicitud:

```
curl

https://api.themoviedb.org/3/movie/latest?api_key=<<
api_key>>
```

Y obtendremos una respuesta (con código de estado 200) como:


```
  "adult": false,
  "backdrop_path": null,
  "belongs_to_collection": null,
  "budget": 0,
  "genres": [],
  "homepage": "",
  "id": 774645,
  "imdb_id": null,
  "original_language": "ru",
  "original_title": "Не моя история",
  "overview": "Five of the six heroes of the film came to
Kazan from different parts of the earth to go beyond their
fear and for the first time openly declare their HIV status
by running a marathon in a special uniform. For Deanna, who
flew in from Australia with her 70-year-old mother, talking
about her diagnosis is a daily job throughout her life. She
is an example for the American Sean, who prepared for the
\"coming out\" long and carefully. Seam is sure that his
parents in an Indonesian village will hardly ever know about
his participation in the Open Faces team in distant Kazan.
Zhandos, a young doctor from Kazakhstan, only worries about
how the news will affect his mother.",
  "popularity": 0,
  "poster_path": null,
  "production_companies": [],
  "production_countries": [],
  "release_date": "",
  "revenue": 0,
  "runtime": 21,
  "spoken_languages": [
    {
      "english_name": "Russian",
      "iso_639_1": "ru",
      "name": "Русский"
    }
  ],
  "status": "Released",
  "tagline": "",
  "title": "Not My Story",
  "video": false,
  "vote_average": 0,
  "vote_count": 0
}
```

Tenga en cuenta que la respuesta anterior puede cambiar dependiendo de cuándo llame a la API. El código de estado 200 indica que fue una solicitud exitosa, una que arrojó la respuesta deseada (de encontrar las últimas películas). Si la solicitud falla por alguna razón, es posible que no obtengamos el resultado deseado y esto será indicado por el código de estado. Estos códigos de estado son útiles para los mecanismos de verificación de errores. Puede leer más sobre los códigos de estado HTTP aquí.

¡Y así es como interactuamos y usamos una API REST! El proceso es similar para otras solicitudes como POST y PUT, donde nosotros como cliente también podemos adjuntar datos para enviar al servidor.

Por otra parte, una solicitud se envía de cliente a servidor en forma de URL web como solicitud HTTP GET o POST o PUT o DELETE. Después de eso, vuelve una respuesta del servidor en forma de recurso que puede ser cualquier cosa como HTML, XML, Image o JSON. Pero ahora JSON es el formato más popular que se utiliza en los servicios web.

En HTTP hay cinco métodos que se utilizan comúnmente en una arquitectura basada en REST, es decir, POST, GET, PUT, PATCH y DELETE. Estos corresponden a las operaciones de creación, lectura, actualización y eliminación (o CRUD) respectivamente. Hay otros métodos que se utilizan con menos frecuencia como OPTIONS y HEAD.

1. GET: el método HTTP GET se utiliza para leer (o recuperar) una representación de un recurso. En la ruta segura, GET devuelve una representación en XML o JSON y un código de respuesta HTTP de 200 (OK). En caso de error, la mayoría de las veces devuelve un 404 (NO ENCONTRADO) o 400 (PETICIÓN MALA).
2. POST: El verbo POST se utiliza con mayor frecuencia para crear nuevos recursos. En particular, se utiliza para crear recursos subordinados. Es decir, subordinado a algún otro recurso (por ejemplo, padre). En la creación exitosa, devuelva el estado HTTP 201, devolviendo un encabezado de ubicación con un enlace al recurso

recién creado con el estado HTTP 201.

NOTA: POST no es seguro ni idempotente.

1. PUT: Se utiliza para actualizar las capacidades. Sin embargo, PUT también se puede utilizar para crear un recurso en el caso de que el cliente elija el ID del recurso en lugar del servidor. En otras palabras, si el PUT es para un URI que contiene el valor de un ID de recurso inexistente. En una actualización exitosa, devuelva 200 (o 204 si no devuelve ningún contenido en el cuerpo) de un PUT. Si usa PUT para crear, devuelva el estado HTTP 201 en una creación exitosa. PUT no es una operación segura, pero es idempotente.
2. PATCH: Se utiliza para modificar capacidades. La solicitud PATCH solo necesita contener los cambios en el recurso, no el recurso completo. Esto se parece a PUT, pero el cuerpo contiene un conjunto de instrucciones que describen cómo se debe modificar un recurso que reside actualmente en el servidor para producir una nueva versión. Esto significa que el cuerpo de PATCH no debe ser solo una parte modificada del recurso, sino en algún tipo de lenguaje de parche como JSON Patch o XML Patch. PATCH no es seguro ni idempotente.
3. ELIMINAR: Se utiliza para eliminar un recurso identificado por un URI. En caso de eliminación exitosa, devuelva el estado HTTP 200 (OK) junto con un cuerpo de respuesta.

Idempotencia: un método HTTP idempotente es un método HTTP que se puede llamar muchas veces sin resultados diferentes. No importaría si el método se llama solo una vez o diez veces. El resultado debería ser el mismo. Nuevamente, esto solo se aplica al resultado, no al recurso en sí. Ejemplo,