API REST

Bryan Ivan Montiel Ortega

Noviembre 2023

Índice

¿Qué es REST API?

Una API, o interfaz de programación de aplicaciones, es un conjunto de reglas que definen cómo las aplicaciones o los dispositivos pueden conectarse y comunicarse entre sí. Una API REST es una API que cumple los principios de diseño del estilo de arquitectura REST o transferencia de estado representacional. Por este motivo, las API REST a veces se conocen como API RESTful.

La `'arquitectura REST'`, que fue definida por primera vez en 2000 por el Dr. Roy Fielding, científico de sistemas, en su tesis doctoral, proporciona un nivel relativamente alto de flexibilidad y libertad a los desarrolladores. Esta flexibilidad es solo una de las razones por las que han surgido las API REST como método común para conectar componentes y aplicaciones en una arquitectura de microservicios.

> *La `'arquitectura de microservicios'` es un enfoque arquitectónico nativo de la nube en el que una sola aplicación está compuesta por muchos componentes más pequeños, o servicios, acoplados de forma laxa e independientemente desplegables.*

Las API

Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. 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 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.

En otras palabras, las API le 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.

Imagínelas como si fueran los 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.

Otra ventaja de las API es que usted no necesita saber cómo se recibe el recurso ni de dónde proviene.

¿Qué es REST?

REST no es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura. Los desarrolladores de las API pueden implementarlo de distintas maneras.

Cuando el cliente envía una solicitud a través de una API de RESTful, esta transfiere una representación del estado del recurso requerido a quien lo haya solicitado o al extremo. La información se entrega por medio de HTTP en uno de estos formatos: JSON (JavaScript Object Notation), HTML, XLT, Python, PHP

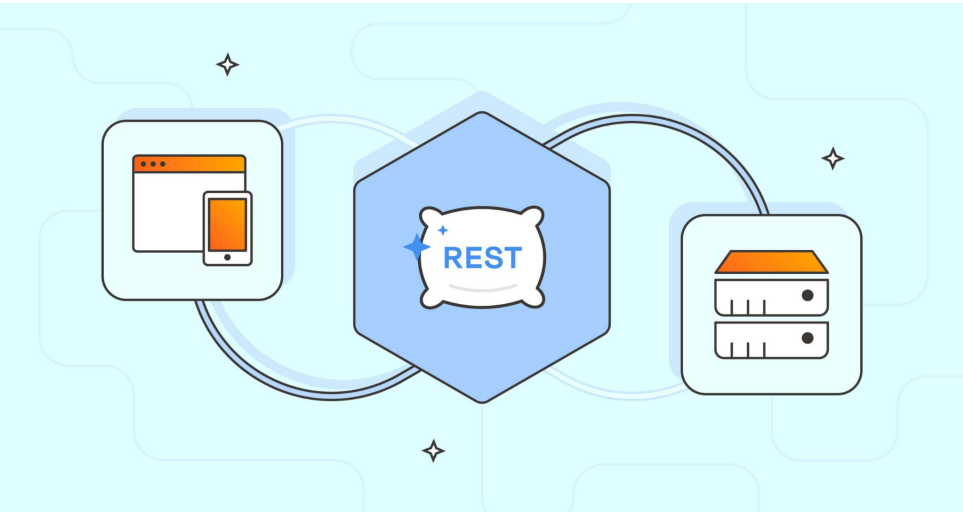
o texto sin formato. JSON es el lenguaje de programación más popular, ya que tanto las máquinas como las personas lo pueden comprender y no depende de ningún lenguaje, a pesar de que su nombre indique lo contrario.

También es necesario tener en cuenta otros aspectos. Los encabezados y los parámetros también son importantes en los métodos HTTP de una solicitud HTTP de la API de RESTful, ya que contienen información de identificación importante con respecto a los metadatos, la autorización, el identificador uniforme de recursos (URI), el almacenamiento en caché, las cookies y otros elementos de la solicitud. Hay encabezados de solicitud y de respuesta, pero cada uno tiene sus propios códigos de estado e información de conexión HTTP.

¿Cómo funciona la API REST’

La API REST se comunica a través de solicitudes HTTP y completa las siguientes funciones: crear, leer, actualizar y eliminar datos. También se conocen como operaciones CRUD. REST proporciona la información sobre los recursos solicitados y utiliza cuatro métodos para describir qué hacer con un recurso:

- POST : creación de un recurso;
- GET : obtener un recurso;
- PUT — actualizar un recurso;
- DELETE : eliminación de un recurso.



Recurso

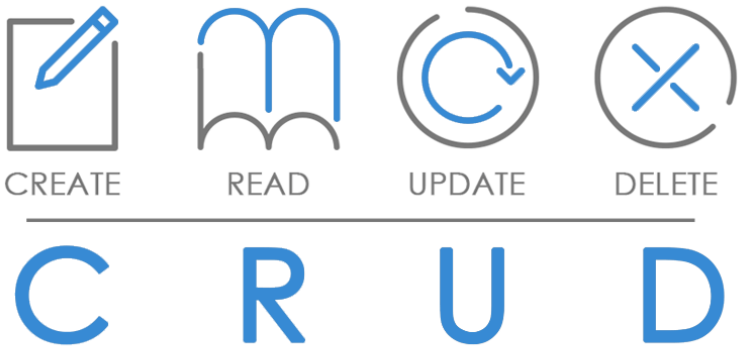
Un recurso es un concepto crítico en REST API, una abstracción de información. Puede ser cualquier información: documento, imagen, servicio temporal.

El estado del recurso en un momento dado se conoce como representación del recurso, que consta de datos, `'metadatos'` que describen los datos y enlaces hipermedia para ayudar a los clientes a pasar al siguiente estado.

La información puede ser entregada al cliente en varios formatos: JSON, HTML, XLT, Python o texto plano. El más popular y usado es JSON porque es legible por humanos y máquinas, y es independiente del lenguaje.

Para acceder a un recurso, un cliente debe realizar una solicitud. Después de recibirlo, el servidor generará una respuesta con datos codificados sobre un recurso.

La estructura de la solicitud incluye cuatro componentes principales: el método HTTP (CRUD que mencionamos anteriormente), puntos finales, encabezados y cuerpo.



El `'método HTTP'` describe lo que se debe hacer con el recurso. Justo arriba, mencionamos cuatro métodos disponibles: POST, GET, PUT, DELETE.

El `'punto final'` contiene un URI: identificador uniforme de recursos, que indica cómo y dónde se puede encontrar el recurso. Una URL o ubicación uniforme de recursos es el tipo de URI más común y representa una dirección web completa.

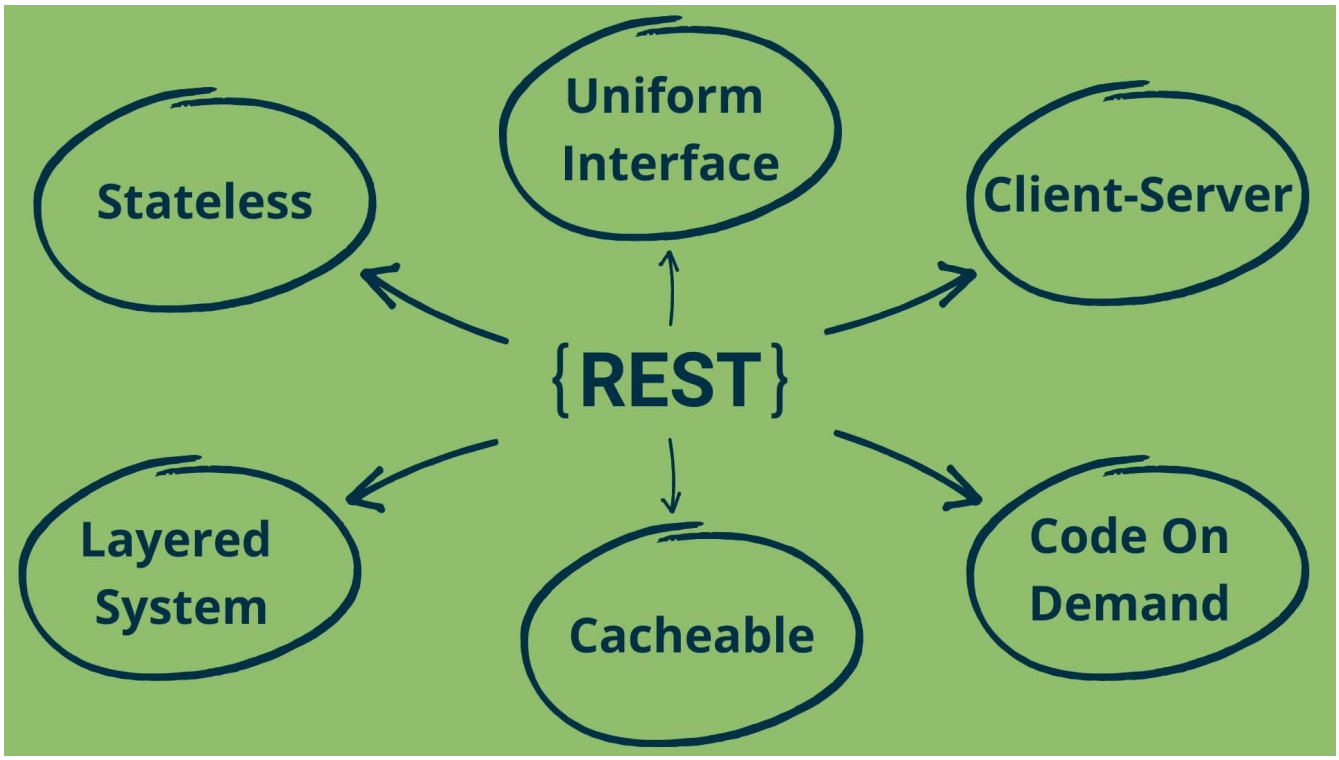
Los `'encabezados'` contienen los datos relacionados con el cliente y el servidor. Los encabezados incluyen datos de autenticación: clave API, nombre, dirección IP que pertenece a la computadora en la que está instalado el servidor y también la información sobre el formato de respuesta.

El `'cuerpo'` se usa para enviar información adicional al servidor, como los datos que desea agregar.

Criterios para que una API se considere RESTful

Algunas API, como SOAP o XML-RPC, imponen una infraestructura estricta a los desarrolladores. No obstante, las API REST se pueden desarrollar utilizando prácticamente cualquier lenguaje de

programación y dar soporte a una amplia variedad de formatos de datos. El único requisito es que cumplan los siguientes seis principios de diseño de REST, también conocidos como restricciones de arquitectura.



REST es un conjunto de pautas que pueden implementarse según sea necesario. Por esta razón, las API de REST son más rápidas y ligeras, cuentan con mayor capacidad de ajuste y, por ende, resultan ideales para el Internet de las cosas (IoT) y el desarrollo de aplicaciones para dispositivos móviles.

1 Interfaz uniforme

Todas las solicitudes de API para el mismo recurso deben ser iguales, independientemente de la procedencia de la solicitud. La API REST debe asegurarse de que un mismo dato, por ejemplo, el nombre o la dirección de correo electrónico de un usuario, pertenezca a un único identificador de recurso uniforme (URI). Los recursos no deben ser demasiado grandes, pero deben contener cada dato que el cliente pueda necesitar.

2 Desacoplamiento del cliente-servidor

En el diseño de la API REST, las aplicaciones de cliente y servidor deben ser completamente independientes entre sí. La única información que la aplicación de cliente debe conocer es el URI del recurso solicitado; no puede interactuar con la aplicación de servidor de ninguna otra forma. De forma

similar, una aplicación de servidor no debe modificar la aplicación de cliente sino para pasarle los datos solicitados a través de HTTP.

3 Sin estado

Las API REST son API sin estado, lo que significa que cada solicitud debe incluir toda la información necesaria para procesarla. Es decir, las API REST no requieren ninguna sesión del lado del servidor. Las aplicaciones de servidor no pueden almacenar datos relacionados con una solicitud de cliente.

4 Capacidad de almacenamiento en memoria caché

Siempre que sea posible, los recursos deben poder almacenarse en la memoria caché en el lado del cliente o el servidor. Las respuestas de servidor también deben contener información sobre si el almacenamiento en memoria caché está permitido para el recurso entregado. El objetivo es mejorar el rendimiento en el lado del cliente, a la vez que se aumenta la escalabilidad en el lado del servidor.

5 Arquitectura del sistema en capas

En las API REST, las llamadas y las respuestas pasan por diferentes capas. Como regla general, no suponga que las aplicaciones de cliente y servidor se conectan directamente entre sí. Puede que haya varios intermediarios diferentes en el bucle de comunicación. Las API REST deben diseñarse para que ni el cliente ni el servidor puedan reconocer si se comunican con la aplicación final o con un intermediario.

6 Código bajo demanda (opcional)

Las API REST normalmente envían recursos estáticos, pero en algunos casos, las respuestas también pueden contener código ejecutable (por ejemplo, applets Java). En estos casos, el código solo debe ejecutarse bajo demanda.

Historia

Antes de REST, la mayoría de los desarrolladores tenían que usar SOAP para integrar las API. SOAP era conocido por ser complejo de construir, usar y depurar. Afortunadamente, un grupo de desarrolladores, liderados por Roy Fielding, crearon REST, cambiando el panorama de las API para siempre.

Esta es la cronología histórica de las API de REST:

- Antes de REST: los desarrolladores usaban SOAP para integrar las API escribiendo a mano un documento XML con una llamada a procedimiento remoto (RPC) en el cuerpo. A continuación, los desarrolladores especificarían el punto de conexión y enviarían su folder SOAP a ese punto de conexión.

- 2000: Un grupo de desarrolladores, entre los que se encontraba Roy Fielding, decidió crear un estándar para que cualquier servidor pueda comunicarse con cualquier otro servidor. Definió las restricciones para las API REST. Dado que estas reglas son universales, es más sencillo para los desarrolladores integrar el software necesario.
- 2002: En 2002, eBay construyó su API REST, expandiendo su mercado a cualquier sitio que pudiera acceder a su API. Como resultado, llamó la atención de Amazon, otro gigante del comercio electrónico, que anunció su API en 2002.
- 2004-2006: En 2004, Flickr lanzó su API RESTful, que permite a los blogueros incrustar fácilmente imágenes en sus sitios, así como en sus feeds de redes sociales. Luego, Facebook y Twitter lanzaron sus API dos años después, cuando se dieron cuenta de que una gran cantidad de desarrolladores estaban raspando los sitios y creando API "Frankenstein".
- 2006-Ahora: Hoy en día, los desarrolladores han adoptado completamente las API RESTful, usándolas para agregar funcionalidad dentro de sus sitios web y aplicaciones. Postman simplifica el proceso de creación de una API y agiliza la colaboración para que pueda crear API más rápido.

¿Para qué se utilizan las API REST?

Una de las principales ventajas de REST es que proporcionan mucha flexibilidad, lo que le permite hacer más con esta API en particular. A continuación se enumeran ejemplos de para qué son útiles las API REST:

Aplicaciones en la nube

Las API de REST son útiles en las aplicaciones en la nube porque sus llamadas no tienen estado. Si algo falla, los componentes sin estado pueden volver a implementarse y escalarse sin problemas para adaptarse a los cambios de carga. El intercambio de documentos, el almacenamiento, las finanzas y la contabilidad, la gestión de las relaciones con los clientes (CRM), el control de inventario y la recopilación de información son algunos de los trabajos que se realizan con las aplicaciones basadas en la nube.

Servicios en la nube

REST también es útil en los servicios en la nube, ya que tendría que controlar cómo se descodifica la dirección URL para enlazar a un servicio a través de una API. Dicho esto, la computación en la nube y los microservicios sin duda harán que el diseño de API RESTful sea la regla del futuro.

Uso de la web

Dado que REST no está vinculado a la tecnología del lado cliente, se puede acceder a estas API desde un proyecto web del lado cliente, una aplicación iOS, un dispositivo IoT o un Windows Phone. Puede crear la infraestructura para su organización sin preocuparse por estar atascado en una pila particular del lado del cliente.

Las ventajas de utilizar las API REST

REST es preferible a SOAP por varias razones. Estas son algunas de las ventajas que tienen las API REST:

- Escalabilidad: Debido a la separación entre cliente y servidor, el producto puede ser escalado por los equipos de desarrollo sin mucha dificultad.
- Flexibilidad y portabilidad: Dado que las API de estilo REST requieren que los datos de una de las solicitudes se envíen correctamente, es posible realizar una migración de un servidor a otro. También es posible realizar cambios en la base de datos en cualquier momento.
- Independencia: Con la separación entre cliente y servidor, el protocolo facilita que los desarrollos en un proyecto se lleven a cabo de forma independiente. Las API REST también se adaptan a la sintaxis y la plataforma de trabajo, lo que ofrece oportunidades para probar varios entornos a la vez durante el desarrollo.
- Peso ligero: Las API REST son ligeras y rápidas, ya que utilizan el estándar HTTP que admite múltiples formatos, incluidos JSON, XML y HTML. Esta característica lo hace ideal para proyectos de aplicaciones móviles, dispositivos IoT y mucho más.

Desafíos del uso de las API REST

Junto con las limitaciones de diseño y arquitectura, las personas tendrán que lidiar con algunos desafíos al usar las API REST. Estos desafíos pueden incluir:

Consenso de punto de conexión REST

No importa cómo formatees tus URL, pero la coherencia en toda tu API es crucial. Desafortunadamente, el número de combinaciones aumenta aún más con operaciones más complejas. Como resultado, la coherencia puede ser difícil de lograr en bases de código grandes con muchos desarrolladores.

Control de versiones de la API de REST

El control de versiones de la API es la práctica de crear varias versiones de una API para adaptarse a los cambios o actualizaciones sin interrumpir a los consumidores. Para evitar problemas de compatibilidad, las API suelen tener versiones. Sin embargo, los puntos de conexión antiguos permanecen activos, lo que conduce a un aumento de la carga de trabajo, ya que se mantienen varias API.

Autenticación de la API REST

La autenticación de la API variará en función del contexto de su uso. Algunas aplicaciones de terceros se consideran usuarios que han iniciado sesión con derechos y permisos específicos. Otros usuarios registrados pueden utilizar otras aplicaciones de terceros en las que solo pueden acceder a sus datos,

como buscar correos electrónicos o documentos. Podría haber más de 20 enfoques de autorización diferentes en uso, lo que aumenta drásticamente la dificultad de llegar a realizar su primera llamada a la API. Con tanta fricción desde el principio, los desarrolladores a veces terminan alejándose.

Seguridad de la API REST

A pesar de que las API RESTful proporcionan una forma más sencilla de acceder y manipular la aplicación, pueden surgir problemas de seguridad. Por ejemplo, un cliente puede enviar miles de solicitudes cada segundo y bloquear su servidor. Otros desafíos de seguridad de la API REST incluyen:

- Falta de autenticación adecuada
- Ausencia de limitación de velocidad y estrangulamiento
- Error al cifrar los datos de la carga útil
- Implementación incorrecta de HTTPS
- Claves de API débiles que se ven comprometidas fácilmente

Múltiples solicitudes y datos innecesarios

Una respuesta puede contener más datos de los que necesita o requerir más solicitudes para acceder a todos los datos.

Mejores prácticas de API REST

Aunque la flexibilidad es una gran ventaja del diseño de API REST, esa misma flexibilidad puede hacer que se diseñe una API inservible o que no funcione correctamente. Por este motivo, los desarrolladores profesionales comparten mejores prácticas en las especificaciones de la API REST.

La especificación OpenAPI (OAS) establece una interfaz para describir una API de manera que permita que cualquier desarrollador o aplicación la descubra y comprenda completamente sus parámetros y prestaciones: puntos finales disponibles, operaciones permitidas en cada punto final, parámetros de funcionamiento, métodos de autenticación y otra información. La última versión, OAS3, 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 empieza siguiendo las *'mejores prácticas'* del sector, por ejemplo, el uso de '`algoritmos hash`' para garantizar la seguridad de contraseñas y HTTPS para proteger la transmisión de datos. Una infraestructura de autorización como OAuth 2.0 puede ayudar a limitar los privilegios de las aplicaciones de terceros. Utilizando una indicación de fecha y hora en la cabecera HTTP, una API también puede rechazar cualquier solicitud que llegue después de un determinado periodo de tiempo. La validación de parámetros y los tokens web de JSON son otras formas de garantizar que solo los clientes autorizados puedan acceder a la API.

La especificación OpenAPI

La especificación OpenAPI (también conocida como OAS) es un estándar de descripción de API que define una interfaz de programación de aplicaciones (API) de HTTP de forma independiente del lenguaje de programación. Esto permite a los desarrolladores describir y documentar sus API de una manera clara y coherente. La especificación OpenAPI se utiliza comúnmente para crear documentación de API, generar código de cliente y servidor, y validar la conformidad de la API con el estándar.

La especificación OpenAPI es mantenida por la Iniciativa OpenAPI, que es un grupo de empresas y organizaciones que trabajan juntas para mejorar la interoperabilidad de las API. La especificación OpenAPI es compatible con muchos lenguajes de programación y marcos de trabajo, lo que la hace muy versátil y fácil de usar.

¿Qué es OAuth 2.0?

OAuth 2.0, que significa “Open Authorization” (autorización abierta), es un estándar diseñado para permitir que un sitio web o una aplicación accedan a recursos alojados por otras aplicaciones web en nombre de un usuario. Sustituyó a OAuth 1.0 en 2012 y ahora es el estándar de facto de la industria para la autorización en línea. OAuth 2.0 proporciona acceso consentido y restringe las acciones que la aplicación del cliente puede realizar en los recursos en nombre del usuario, sin compartir nunca las credenciales del usuario.

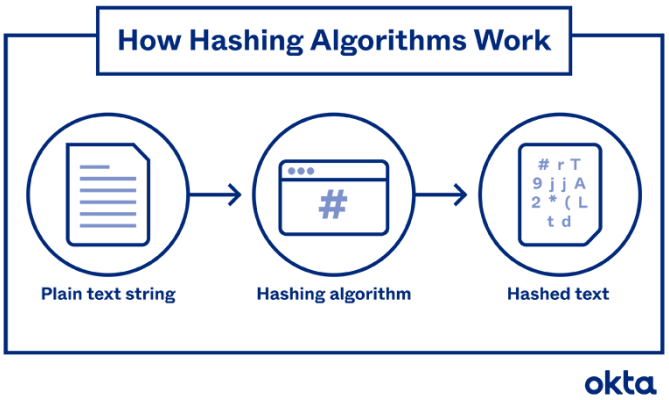
Aunque la web es la principal plataforma para OAuth 2, la especificación también describe cómo manejar este tipo de acceso delegado a otros tipos de clientes (aplicaciones basadas en el navegador, aplicaciones web del lado del servidor, aplicaciones nativas/móviles, dispositivos conectados, etc.).

Algoritmos Hash

Un algoritmo hash es una función matemática que cifra los datos y los hace ilegibles.

Los algoritmos hash son programas unidireccionales, por lo que el texto no puede ser descifrado por nadie más. Y ese es el punto. El hashing protege los datos en reposo, por lo que incluso si alguien obtiene acceso a su servidor, los elementos almacenados allí permanecen ilegibles.

El hashing también puede ayudarte a demostrar que los datos no se ajustaron o alteraron después de que el autor terminó con ellos. Y algunas personas usan el hashing para ayudarles a dar sentido a grandes cantidades de datos



Ejemplos de API REST

Amazon S3

Con muchas empresas de primer nivel que ofrecen estos servicios, el uso de API REST para aplicaciones de inteligencia artificial, ciencia de datos y aprendizaje automático está en aumento. Los servicios de IA de AWS de Amazon permiten a los desarrolladores incorporar la funcionalidad de IA en sus aplicaciones para una interacción más adaptable e inteligente. Esto también puede ayudar a proteger el intercambio de datos entre sistemas mediante la detección de posibles vulnerabilidades de seguridad.

Twitter

Con 450 millones de usuarios activos mensuales, Twitter tiene un enorme alcance en el ámbito de las redes sociales. Para los desarrolladores, la API de Twitter ofrece una forma de integrar la funcionalidad de Twitter y promocionar sus aplicaciones a través de la plataforma.

Con la API de Twitter, los desarrolladores pueden agilizar el proceso de registro aprovechando el sistema de identificación de Twitter. La API también permite mostrar tweets a los usuarios en función de criterios como la ubicación o los hashtags de tendencia, así como un marketing eficaz utilizando los datos de Twitter.

Instagram

La API básica de visualización de Instagram proporciona a los desarrolladores acceso a datos de perfil, imágenes y videos en la plataforma. Esto permite a los desarrolladores crear aplicaciones que integran los datos de los usuarios de Instagram en sus propios productos. Además, Instagram ofrece una API Graph para cuentas profesionales, lo que permite a los usuarios administrar sus actividades en línea.

Plaid

El creciente mercado de productos SaaS está impulsando el crecimiento de las API REST en el sector FinTech, y Plaid es una de las principales empresas que lideran la "democratización de los datos" en los servicios financieros. Este enfoque hace que los datos sean accesibles para todos, independientemente de su capacidad técnica, lo que permite la creación de experiencias personalizadas que satisfagan las necesidades de los usuarios.

Bibliografía

1. Buenas practicas para el manejo de errores en tu REST API. (2019, noviembre 8). RicardoGeek. https://ricardogeek.com/buenas-practicas-para-el-manejo-de-errores-en-tu-rest-api/

2. Hashing algorithm overview: Types, methodologies & usage. (s/f). Okta.com. Recuperado el 10 de noviembre de 2023, de https://www.okta.com/identity-101/hashing-algorithms/

3. Izertis. (2017, octubre 6). Mejores prácticas para su API REST. Izertis.com; Izertis. https://www.izertis.com/es/-/blog/mejores-practicas-para-su-api-rest

4. Mejores prácticas en la construcción de una API Rest. (2021, enero 28). CodersLink. https://coderslink.com/talento/blog/mejores-practicas-en-la-construccion-de-una-api-rest/

5. OpenAPI-Specification: The OpenAPI Specification Repository. (s/f).

6. ¿Qué es la API REST y en qué se diferencia de otros tipos? (2022, enero 20). Appmaster.io; AppMaster. https://appmaster.io/es/blog/que-es-la-api-rest-y-en-que-se-diferencia-de-otros-tipos

7. ¿Qué es OAuth 2.0 y para qué sirve? (s/f). Auth0. Recuperado el 10 de noviembre de 2023, de https://auth0.com/es/intro-to-iam/what-is-oauth-2

8. ¿Qué es una API REST? (s/f). Ibm.com. Recuperado el 10 de noviembre de 2023, de https://www.ibm.com/es-es/topics/rest-apis

9. REST API. (s/f). Redhat.com. Recuperado el 10 de noviembre de 2023, de https://www.redhat.com/es/topics/api/what-is-a-rest-api

10. REST Architectural Constraints: The 6 guiding principles behind a truly RESTful API. (s/f). Saurabhmisra.dev. Recuperado el 10 de noviembre de 2023, de https://www.saurabhmisra.dev/rest-architectural-constraints/

11. What is a REST API? Examples, uses & challenges. (2023, junio 28). Postman Blog; Postman. https://blog.postman.com/rest-api-examples/