¿Qué es una API?

API (Application Programming Interface) es un mecanismo de Software que permite a dos componentes de Software a comunicarse entre sí mediante un conjunto de definiciones y protocolos.

La comunicación entre una API y otro Software se lleva a cabo por medio de solicitudes y respuestas. La arquitectura de una API está basada en la arquitectura cliente-servidor, la aplicación que envía la solicitud pasa a ser el cliente, y quien envía la respuesta se llama servidor.

Tipos de API

Existen 4 clasificaciones de API's, y estas dependen del motivo de la creación, es decir, su propósito de funcionamiento.

API de SOAP

Estas API utilizan el protocolo simple de acceso a objetos. El cliente y el servidor intercambian mensajes mediante XML. Se trata de una API menos flexible que era más popular en el pasado.

API de RPC

Estas API se denominan llamadas a procedimientos remotos. El cliente completa una función (o procedimiento) en el servidor, y el servidor devuelve el resultado al cliente.

API de WebSocket

La API de WebSocket es otro desarrollo moderno de la API web que utiliza objetos JSON para transmitir datos. La API de WebSocket admite la comunicación bidireccional entre las aplicaciones cliente y el servidor. El servidor puede enviar mensajes de devolución de llamada a los clientes conectados, por lo que es más eficiente que la API de REST.

API de REST

Estas son las API más populares y flexibles que se encuentran en la web actualmente. El cliente envía las solicitudes al servidor como datos. El servidor utiliza esta entrada del cliente para iniciar funciones internas y devuelve los datos de salida al cliente. Veamos las API de REST con más detalle a continuación.

Las API Rest son las más utilizadas comúnmente, REST significa transferencia de estado representacional. REST define un conjunto de funciones como lo son GET, POST, PUT y DELETE, parecido a un CRUD en bases de datos, GET hace referencia a obtener recursos, POST a enviarlos, PUT a actualizarlos y DELETE a eliminar recursos, esta son las acciones que los clientes pueden realizar dentro de una API, el protocolo de comunicación entre el cliente y el servidor es el HTTP, por este medio se intercambian los datos entre cliente/servidor.

Una de las principales características de las API REST es que no tienen estado, esto significa que los servidores no guardan los datos del cliente, entre las solicitudes. Las solicitudes de los clientes al servidor son similares a las URL que se escriben en el

navegador para llegar a un sitio web, sin embargo, a diferencia, la respuesta del servidor son datos simples.

Características de una API REST

Existen criterios específicos para determinar que una API es REST, algunos de los más relevantes son:

- Debe de cumplir con la arquitectura cliente-servidor.
- Las ejecuciones de la API no deben de considerar el estado del cliente, el estado de peticiones anteriores o algún indicador almacenado que haga variar su comportamiento. La comunicación debe ser sin estado (stateless).
- Ha de estar orientado a recursos, usando las operaciones estándar de los verbos HTTP.
- Hace uso de la URL como identificador único de los recursos
.
- Debe ser hipermedia: cuando se consulte un recurso, este debe contener links o hipervínculos de acciones o recursos que lo complementen.

Verbos HTTP utilizados en la comunicación entre API's

Son aquellos verbos propios de HTTP que fueron tomados para definir operaciones muy puntuales y específicas sobre los recursos de la API.

Los más utilizados son:

1. GET: listado de recursos. Detalle de un solo recurso.
2. POST: creación de un recurso.PUT: modificación total de un recurso.
3. PATCH: modificación parcial de un recurso.

4. DELETE: eliminación de un recurso. En muchas ocasiones es un soft delete, es decir, no se elimina definitivamente un recurso sino que únicamente es marcado como eliminado o desactivado.

5. URL orientada a recursos: la definición de las URL de los endpoint de la API están orientadas a recursos, es decir, a entidades que tienen coherencia dentro del contexto de la API. Por ejemplo, en una API para un sistema que administra libros sería fácil encontrar entidades como libros, autores, editoriales, colecciones, etc. Estas entidades las veremos reflejadas en URL orientadas a recursos que las representen, por ejemplo:

- /api-libros/v0/autores: identifica los recursos autores
- /api-libros/v0/autores/{id-autor}: identifica un recurso autor
- /api-libros/v0/autores/{id-autor}/libros: identifica los libros de un autor en específico
- /api-libros/v0/libros: identifica los recursos libros
- /api-libros/v0/editoriales: identifica los recursos editoriales
- /api-libros/v0/editoriales/{id-editorial}/libros: identifica los libros de una editorial

HHTP Status

Algunos de los status de respuestas propios del protocolo HTTP que fueron tomadas para informar sobre el resultado de la operación solicitada. Los más comunes en API REST son:

1. 200 - OK
2. 201 – Created
3. 204 - No Content
4. 400 - Bad Request
5. 401 – Unauthorized
6. 403 – Forbidden
7. 404 - Not Found
8. 500 - Internal Server Error

¿Cuáles son los beneficios de las API de REST?

Las API de REST ofrecen cuatro beneficios principales.

1. Integración: Las API se utilizan para integrar nuevas aplicaciones con los sistemas de software existentes. Esto aumenta la velocidad de desarrollo, ya que no hay que escribir cada funcionalidad desde cero. Puede utilizar las API para aprovechar el código existente.

2. Innovación: Sectores enteros pueden cambiar con la llegada de una nueva aplicación. Las empresas deben responder con rapidez y respaldar la rápida implementación de servicios innovadores. Para ello, pueden hacer cambios en la API sin tener que reescribir todo el código.

3. Ampliación: Las API presentan una oportunidad única para que las empresas satisfagan las necesidades de sus clientes en diferentes plataformas. Por ejemplo, la API de mapas permite la integración de información de los mapas en sitios web, Android, iOS, etc. Cualquier empresa puede dar un acceso similar a sus bases de datos internas mediante el uso de API gratuitas o de pago.

4. Facilidad de mantenimiento: La API actúa como una puerta de enlace entre dos sistemas. Cada sistema está obligado a hacer cambios internos para que la API no se vea afectada. De este modo, cualquier cambio futuro que haga una de las partes en el código no afectará a la otra.

¿Cómo proteger una API REST?

Todas las API deben protegerse mediante una autenticación y una supervisión adecuadas. Las dos maneras principales de proteger las API de REST son las siguientes:

1. Tokens de autenticación: Se utilizan para autorizar a los usuarios a hacer la llamada a la API. Los tokens de autenticación comprueban que los usuarios son quienes dicen ser y que tienen los derechos de acceso para esa llamada concreta a la API. Por ejemplo, cuando inicia sesión en el servidor de correo electrónico, el cliente de correo electrónico utiliza tokens de autenticación para un acceso seguro.

2. Claves de API: Las claves de API verifican el programa o la aplicación que hace la llamada a la API. Identifican la aplicación y se aseguran de que tiene los derechos de acceso necesarios para hacer la llamada a la API en cuestión. Las claves de API no son tan seguras como los tokens, pero permiten supervisar la API para recopilar datos sobre su uso. Es posible que haya notado una larga cadena de caracteres y números en la URL de su navegador cuando visita diferentes sitios web. Esta cadena es una clave de la API que el sitio web utiliza para hacer llamadas internas a la API.

¿Cómo crear una API?

Se requiere la debida diligencia y esfuerzo para crear una API con la que otros desarrolladores quieran trabajar y en la que confíen. Los siguientes son los cinco pasos necesarios para un diseño de API de alta calidad:

1. Planificación de la API: Las especificaciones de la API, como OpenAPI, proporcionan el esquema para el diseño de su API. Es mejor pensar en los diferentes casos de uso por adelantado y asegurarse de que la API cumple con los estándares de desarrollo actuales.

2. Creación de la API: Los diseñadores de API crean prototipos de API mediante código reutilizable. Una vez probado el prototipo, los desarrolladores pueden personalizarlo a las especificaciones internas.

3. Prueba de la API: Las pruebas de la API son las mismas que las del software y deben hacerse para evitar errores y defectos. Las herramientas de pruebas de la API pueden utilizarse para reforzar la prueba de la API contra los ciberataques.

4. Documentación de la API: Aunque las API son explicativas, la documentación de estas sirve de guía para mejorar su uso. Las API bien documentadas que ofrecen una gama de funciones y casos de uso tienden a ser más populares en una arquitectura orientada a servicios.

5. Comercialización de la API: Así como Amazon es un mercado en línea para la venta minorista, los mercados de API existen para que los desarrolladores compren y vendan otras API. Publicar su API puede permitirle monetizarla.